МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ОНЛАЙН-ГОЛОСУВАНЬ ТА ОПИТУВАНЬ З АНАЛІТИКОЮ РЕЗУЛЬТАТІВ У РЕАЛЬНОМУ ЧАСІ»**

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
В. К. Денисенко
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, д.е.н., доцент Полуектова Н. Р..
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, к.ф.-м.н., доцент Лісняк А. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" 08 " грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Денисенку Владиславу Констянтиновичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Проєктування вебзастосунку для онлайн-голосувань та опитувань з аналітикою результатів у реальному часі

керівник роботи Полуектова Наталія Робертівна, д.е.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 « грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи.
2. Проєктування вебсистеми.
3. Реалізація вебсистеми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.12.2026

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.02.2026	
2.	Збір вихідних даних.	23.03.2026	
3.	Обробка методичних та теоретичних	02.04.2025	
	джерел.		
4.	Розробка першого та другого розділу.	26.04.2026	
5.	Розробка третього розділу.	11.05.2026	
6.	Оформлення та нормоконтроль	04.06.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2026	

Студент _____ В. К. Денисенко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н. Р. Полуектова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проєктування вебзастосунку для онлайн-голосувань та опитувань з аналітикою результатів у реальному часі»: 74 с., 13 рис., 12 табл., 28 джерел, 2 додатків.

АНАЛІЗ ДАНИХ, ВЕБЗАСТОСУНОК, ВІЗУАЛІЗАЦІЯ ДАНИХ, ОНЛАЙН-ГОЛОСУВАННЯ, ОПИТУВАННЯ, TYPESCRIPT, NEXT.JS, REACT, TAILWIND CSS, SQLITE, SWR

Об’єктом дослідження є процеси збору, обробки та аналізу даних у системах онлайн-голосувань і опитувань.

Предметом дослідження виступають методи та засоби проєктування вебсистеми для проведення електронного голосування з підтримкою аналітики в реальному часі.

Метою роботи є проєктування вебсистеми для онлайн-голосувань та опитувань з можливістю обробки та візуалізації результатів у реальному часі.

Методи дослідження – аналіз наукових джерел і сучасних вебсервісів онлайн-опитувань, системний аналіз, методи проєктування програмних систем і баз даних, об’єктно-орієнтоване програмування, а також тестування програмного забезпечення.

У роботі проаналізовано сучасні системи онлайн-голосувань та опитувань, визначено їх переваги та недоліки. Спроєктовано архітектуру вебсистеми, структуру бази даних. Реалізовано ключові модулі системи: створення опитувань, участь у голосуванні, обробку голосів у реальному часі та аналітику результатів. Розроблено механізм візуалізації даних у вигляді графіків і діаграм, а також реалізовано базові засоби захисту даних і запобігання повторному голосуванню. Проведене тестування підтвердило коректність роботи системи, її стабільність та зручність використання.

SUMMARY

Bachelor's Thesis «Designing a Web Application for Online Voting and Surveys with Real-Time Results Analytics»: 74 pages, 13 figures, 12 tables, 28 sources, 2 appendices.

DATA ANALYSIS, DATABASE, SECURITY, WEB APPLICATION, VOTING, POLLING, SURVEY SYSTEM, TYPESCRIPT, NEXT.JS, REACT, TAILWIND CSS, SQLITE, SWR

The object of research is the processes of data collection, processing, and analysis in online voting and survey systems.

The subject of research is the methods and tools for designing a web system for electronic voting with real-time analytics support.

The aim of the work is to design a web system for online voting and surveys with the ability to process and visualize results in real time.

Research methods include analysis of scientific sources and modern online survey services, systems analysis, software and database design methods, client-server architecture design, object-oriented programming, and software testing.

The work analyzes modern online voting and survey systems and identifies their advantages and disadvantages. The architecture of the web system, database structure, and main UML diagrams are designed. The core system modules are implemented, including poll creation, voting participation, real-time vote processing, and result analytics. A data visualization mechanism using charts and graphs is developed, along with basic data protection and duplicate voting prevention mechanisms. Testing confirmed the correctness, stability, and usability of the system.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2

Реферат 4

Summary 5

Вступ..... 7

1 Теоретичні основи..... 10

1.1 Поняття та класифікація систем онлайн-голосувань і опитувань 10

1.2 Методи організації електронного голосування: безпека, анонімність, достовірність..... 12

1.3 Огляд сучасних рішень (Google Forms, Mentimeter, Kahoot, SurveyMonkey)..... 16

1.3.1 Google Forms..... 17

1.3.2 Mentimeter..... 18

1.3.3 Kahoot..... 19

1.3.4 SurveyMonkey..... 19

1.3.5 Порівняльний аналіз платформ 20

2 Проєктування вебсистеми 22

2.1 Архітектура системи: клієнт–серверна модель, модулі збору даних, аналітики та візуалізації 22

2.2 Вибір технологій 26

2.3 Моделювання даних: структура користувачів, опитувань, варіантів відповідей, результатів 28

2.4 UML-діаграми..... 32

3 Реалізація вебсистеми 40

3.1 Реалізація інтерфейсу користувача: створення опитувань, участь у голосуванні 40

3.2 Механізм збору та обробки голосів у реальному часі 43

3.3 Аналітика результатів: побудова графіків, діаграм, статистичних звітів…… 48

3.4 Забезпечення безпеки: шифрування даних, захист від повторного голосування..... 51

3.5 Тестування системи: функціональне, навантажувальне, юзабіліті ... 54

3.6 Інструкція користувача..... 55

Висновки 58

Перелік посилань..... 60

Додаток А Інтерфейс користувача 63

Додаток Б Лістинг реалізації механізму збору та обробки голосів у реальному часі 68

ВСТУП

Одним із ключових напрямів розвитку сучасних технологій аналізу великих даних є системи онлайн-голосувань та опитувань, які дозволяють швидко отримувати зворотний зв’язок, приймати колективні рішення та аналізувати громадську думку в режимі реального часу. Актуальність подібних рішень зумовлена зростаючою потребою в прозорих, доступних та інтерактивних інструментах комунікації між користувачами, організаціями, освітніми установами та державними структурами.

Потреба у розробці або удосконаленні вебсистем для голосування також пов’язана з обмеженнями існуючих платформ, які часто мають недостатній рівень гнучкості, обмежену аналітику або не забезпечують достатньої прозорості процесу обробки результатів. Крім того, значна частина комерційних рішень орієнтована на закриті екосистеми, що ускладнює інтеграцію з іншими інформаційними системами або адаптацію під специфічні вимоги користувачів. У цьому контексті розробка власної вебсистеми онлайн-голосувань з аналітикою результатів у реальному часі є обґрунтованою та практично значущою задачею.

Історично розвиток систем електронного голосування пройшов шлях від простих вебформ до складних розподілених систем із використанням технологій потокової обробки даних, WebSockets та хмарних обчислень. Сучасні платформи, такі як Google Forms, SurveyMonkey, Mentimeter та Kahoot!, продемонстрували ефективність онлайн-опитувань у бізнесі, освіті та соціологічних дослідженнях. Водночас їх функціональність часто є універсальною, що створює потребу у створенні спеціалізованих рішень, орієнтованих на підвищену інтерактивність та аналітику в реальному часі.

Метою роботи є проєктування вебсистеми для онлайн-голосувань та опитувань з можливістю обробки та візуалізації результатів у реальному часі. Досягнення поставленої мети передбачає вирішення таких завдань:

– проаналізувати сучасний стан розвитку вебсистем онлайн-голосувань та опитувань і існуючі аналогічні рішення;

– дослідити методи організації електронного голосування з урахуванням вимог безпеки, анонімності та достовірності даних;

– визначити функціональні та нефункціональні вимоги до вебсистеми;

– спроєктувати архітектуру вебсистеми та структуру бази даних;

– обрати та обґрунтувати сучасні технології для реалізації системи;

– розробити механізм обробки та візуалізації результатів голосування в реальному часі.

Об’єктом дослідження є процеси збору, обробки та аналізу даних у системах онлайн-голосувань і опитувань. Предметом дослідження виступають методи та засоби проєктування вебсистеми для проведення електронного голосування з підтримкою аналітики в реальному часі.

Методи дослідження – аналіз наукових джерел і сучасних вебсервісів онлайн-опитувань, системний аналіз, методи проєктування програмних систем і баз даних, клієнт-серверне проєктування, об’єктно-орієнтоване програмування, а також тестування програмного забезпечення.

Наукова новизна роботи полягає у комплексному поєднанні механізмів реального часу (real-time обробки даних), інтерактивної візуалізації результатів та підходів до забезпечення безпеки голосування в межах єдиної вебсистеми. Практична значущість полягає у можливості застосування розробленої системи в освітніх установах, соціологічних дослідженнях, корпоративному середовищі для прийняття управлінських рішень, а також у громадських ініціативах.

Основними проєктними рішеннями, що обґрунтовуються в роботі, є використання клієнт-серверної архітектури, застосування сучасних вебтехнологій для фронтенду та бекенду, а також використання технологій реального часу (зокрема WebSockets) для миттєвого оновлення результатів голосування. Окрему увагу приділено моделюванню даних та забезпеченню захисту інформації шляхом реалізації механізмів автентифікації та запобігання повторному голосуванню.

Результати дослідження можуть бути використані для створення сучасних інтерактивних платформ опитувань, інтеграції систем з освітніми LMS-рішеннями, корпоративними порталами та аналітичними системами. Крім того, запропоновані підходи можуть бути застосовані для подальших досліджень у сфері електронної демократії, цифрових сервісів участі громадян та систем прийняття рішень на основі даних.

Отже, робота спрямована на розробку сучасної, масштабованої та безпечної вебсистеми онлайн-голосувань, що відповідає актуальним вимогам інформаційного суспільства та тенденціям розвитку вебтехнологій.

1 ТЕОРЕТИЧНІ ОСНОВИ

1.1 Поняття та класифікація систем онлайн-голосувань і опитувань

Системи онлайн-голосувань та опитувань є сучасними цифровими інструментами, що забезпечують автоматизований збір, передачу, обробку та аналіз відповідей користувачів через мережу Інтернет. Вони належать до класу інформаційних систем електронної взаємодії та широко використовуються в освіті, бізнесі, соціологічних дослідженнях, маркетингу та процесах прийняття управлінських рішень. У науковій літературі такі системи розглядаються як складова електронної участі (e-participation), де онлайн-голосування орієнтоване на прийняття формалізованого рішення, а онлайн-опитування – на збір і аналіз думок, оцінок або поведінкових даних респондентів [1].

З концептуального погляду, системи електронного голосування визначаються як інформаційні системи, у яких один або кілька етапів голосування (ідентифікація, подання голосу, підрахунок результатів) виконуються за допомогою електронних засобів. Такі системи можуть бути реалізовані через вебзастосунки, мобільні додатки або спеціалізовані програмно-апаратні комплекси. Основними вимогами до них є забезпечення конфіденційності, цілісності даних, автентифікації користувачів та можливості перевірки коректності результатів [2].

На відміну від цього, системи онлайн-опитувань призначені для збору статистичної інформації без обов’язкової процедурної значущості результату. Вони використовуються для дослідження громадської думки, оцінки задоволеності, маркетингових досліджень або освітніх потреб. Основними критеріями якості таких систем є валідність отриманих даних, зручність використання, коректність формулювання питань та репрезентативність вибірки [3].

Ключова відмінність між цими двома типами систем полягає у їх функціональному призначенні. Онлайн-голосування використовується для прийняття формально значущих рішень (вибори, корпоративні рішення, референдуми), тоді як онлайн-опитування спрямовані на збір аналітичної інформації без юридично обов’язкового результату. Це означає, що системи голосування повинні забезпечувати значно вищий рівень безпеки та контролю цілісності даних.

Також відрізняються вимоги до анонімності. У системах голосування анонімність є критично важливою для запобігання впливу на вибір користувача, тоді як в опитуваннях вона може варіюватися залежно від цілей дослідження. Наприклад, у маркетингових або освітніх дослідженнях іноді допускається часткова ідентифікація користувача для аналізу сегментів аудиторії [4].

Існує декілька підходів до класифікації таких систем. Найпоширенішим є поділ за способом участі користувачів:

– Локальні системи – функціонують у контрольованому середовищі (навчальні заклади, організації, виборчі дільниці), де доступ до системи обмежений фізично або мережево.

– Віддалені (онлайн) системи – дозволяють користувачам брати участь у голосуванні або опитуванні з будь-якого місця через Інтернет.

Віддалені системи є більш зручними та масштабованими, однак вони потребують складніших механізмів захисту від несанкціонованого доступу, повторного голосування та зовнішнього втручання [5].

Другим критерієм класифікації є мета використання системи. За цим підходом виділяють:

– системи електронного голосування (e-voting);

– системи онлайн-опитувань;

– платформи громадської думки та пульс-опитування;

– змішані системи для інтерактивної взаємодії користувачів.

Кожен із цих типів має різні вимоги до обробки даних, рівня безпеки та способів представлення результатів. Наприклад, у голосуванні ключовим є

коректний підрахунок голосів, тоді як в опитуваннях – аналітична обробка та візуалізація даних.

Третім критерієм є рівень автентифікації користувачів. У системах онлайн-голосування зазвичай застосовуються суворі механізми перевірки особи користувача, такі як токени доступу, електронні підписи або багатофакторна автентифікація. Водночас в опитуваннях часто використовується анонімний або напіванонімний доступ, що дозволяє підвищити чесність відповідей, але зменшує контроль за унікальністю респондента [2].

Четвертим критерієм класифікації є архітектура системи. Централізовані системи передбачають збереження всіх даних на одному сервері, що спрощує управління, але створює ризик єдиної точки відмови. Децентралізовані або розподілені системи підвищують надійність за рахунок розподілу навантаження між кількома вузлами. У сучасних реалізаціях також використовуються криптографічні методи захисту даних для забезпечення прозорості та незмінності результатів [6].

Тобто системи онлайн-голосувань та опитувань є важливим елементом сучасної цифрової інфраструктури взаємодії з користувачами. Попри спільну технологічну основу, вони суттєво відрізняються за метою, рівнем безпеки, вимогами до автентифікації та методами обробки даних. Класифікація таких систем дозволяє більш чітко визначити вимоги до їх проєктування та реалізації, що є критично важливим для створення ефективної вебсистеми з підтримкою реального часу та аналітики.

1.2 Методи організації електронного голосування: безпека, анонімність, достовірність

Електронне голосування є складною інформаційною процедурою, що охоплює етапи реєстрації користувача, автентифікації, подання голосу, його передачі через мережу, зберігання та підрахунку результатів. У науковій

літературі підкреслюється, що ефективна організація електронного голосування неможлива без комплексного забезпечення трьох базових властивостей: безпеки, анонімності та достовірності результатів. Ці характеристики взаємопов’язані та формують єдину систему вимог, оскільки порушення будь-якої з них призводить до втрати довіри до всієї системи голосування.

Безпека електронного голосування визначається як здатність системи протистояти внутрішнім і зовнішнім загрозам, забезпечуючи захист даних на всіх етапах їх життєвого циклу. У класичному розумінні інформаційної безпеки вона включає конфіденційність, цілісність, доступність, автентичність та невідмовність дій користувачів.

У контексті електронного голосування ці принципи реалізуються таким чином:

- Конфіденційність гарантує, що зміст голосу не може бути переглянутий сторонніми особами;
- Цілісність забезпечує неможливість зміни голосу після його подання;
- Автентичність підтверджує, що голос подано лише уповноваженим користувачем;
- Доступність гарантує стабільну роботу системи під час голосування;
- Невідмовність запобігає можливості заперечення факту голосування.

Дослідники зазначають, що найбільш вразливими є віддалені (online) системи голосування, оскільки вони працюють у неконтрольованому середовищі користувача, що створює ризики атак через шкідливе програмне забезпечення, перехоплення трафіку та підміну даних [7].

Основним інструментом забезпечення безпеки електронного голосування є криптографія. Сучасні наукові підходи виділяють кілька ключових методів:

1. Гомоморфне шифрування. Дозволяє виконувати математичні операції над зашифрованими голосами без їх розшифрування. Це означає, що підрахунок результатів може здійснюватися без доступу до індивідуальних виборів користувачів. Такий підхід значно підвищує рівень конфіденційності.

2. Сліпі підписи. Механізм, який дозволяє виборчому органу підтвердити право користувача голосувати, не отримуючи інформації про зміст його вибору. Це забезпечує розділення процесів автентифікації та голосування, що є критичним для збереження анонімності.

3. Докази з нульовим розголошенням (Zero-Knowledge Proofs). Ці методи дозволяють довести правильність виконаної дії (наприклад, що голос є дійсним), не розкриваючи саму інформацію про голос. Це значно підвищує рівень перевірюваності системи [6; 7].

Анонімність у системах електронного голосування означає неможливість встановлення зв’язку між конкретним виборцем і його голосом. Це є фундаментальною умовою демократичних процесів, оскільки гарантує свободу волевиявлення без страху тиску або впливу.

Для досягнення анонімності застосовуються такі механізми:

- розділення процесів автентифікації та подання голосу;
- використання одноразових токенів доступу;
- застосування криптографічних протоколів (blind signatures, mix-networks);
- шифрування бюлетенів перед зберіганням.

Окремо підкреслюється, що анонімність повинна бути збалансована з можливістю перевірки результатів, оскільки повна відсутність контролю може призвести до фальсифікацій або повторного голосування [7].

У віддалених системах голосування досягнення повної анонімності є складнішим, ніж у традиційних виборах. Це пов’язано з тим, що користувач використовує власний пристрій та мережу, які можуть бути скомпрометовані. Додатковими ризиками є:

- можливість встановлення шпигунського програмного забезпечення;
- контроль пристрою третім особам;
- мережеве перехоплення даних;
- соціальний або психологічний тиск на виборця.

Отже, анонімність в електронному голосуванні є не лише криптографічною, але й організаційною проблемою [8].

Достовірність електронного голосування означає, що всі голоси коректно враховані, жоден голос не втрачено або не змінено, а фінальний результат точно відображає волевиявлення користувачів.

У наукових дослідженнях виділяють два рівні достовірності:

Операційна достовірність – стабільність роботи системи під час голосування (сервери, мережа, база даних);

Результативна достовірність – правильність підрахунку та можливість перевірки результатів.

Для забезпечення достовірності застосовуються:

- резервування серверів;
- логування всіх операцій;
- механізми перевірки цілісності даних;
- аудит результатів;
- end-to-end verifiability (скрізна перевірюваність).

Ці підходи дозволяють підвищити довіру до системи навіть у випадку часткових збоїв або атак [8; 9].

Технологія блокчейн часто розглядається як інноваційний метод організації електронного голосування. Її основною перевагою є незмінність даних: після внесення голосу до ланцюга блоків його неможливо змінити без виявлення.

Переваги блокчейну:

- висока прозорість;
- захист від підробки даних;
- розподілена структура без єдиного центру контролю.

Недоліки:

- складність забезпечення анонімності;
- висока обчислювальна складність;
- необхідність додаткових криптографічних механізмів.

Отже, блокчейн є допоміжною технологією, але не повністю автономним рішенням для електронного голосування [9].

Організація електронного голосування включає послідовність етапів:

- реєстрація користувача;
- перевірка автентичності;
- видача права голосу;
- подання голосу;
- шифрування та передача даних;
- збереження результатів;
- підрахунок і верифікація.

Найбільш ефективною вважається модель, у якій автентифікація і голосування розділені логічно та технічно. Це дозволяє одночасно забезпечити контроль доступу та анонімність користувача. Якщо ці процеси об’єднані, виникає ризик витоку персональних даних та порушення конфіденційності [7; 9].

Отже, методи організації електронного голосування базуються на трьох ключових принципах: безпеці, анонімності та достовірності. Безпека забезпечує захист від атак і втручань, анонімність гарантує свободу волевиявлення користувачів, а достовірність підтверджує коректність отриманих результатів. Наукові джерела підкреслюють, що ефективна система електронного голосування повинна бути багаторівневою та поєднувати криптографічні, програмні й організаційні механізми захисту для забезпечення довіри до результатів.

1.3 Огляд сучасних рішень (Google Forms, Mentimeter, Kahoot, SurveyMonkey)

Сучасний ринок програмного забезпечення для проведення опитувань і тестування пропонує широкий спектр хмарних платформ, кожна з яких орієнтована на певний сценарій використання. У цьому підрозділі розглянуто чотири найпоширеніші рішення: Google Forms, Mentimeter, Kahoot та

SurveyMonkey, – з метою визначення їхніх ключових функціональних особливостей, переваг і обмежень.

1.3.1 Google Forms

Google Forms – це безкоштовний хмарний інструмент для створення форм і опитувань, що входить до екосистеми Google Workspace. Платформа була запущена у 2008 році як функція Google Sheets, а згодом перетворилася на самостійний продукт [10].

Сервіс підтримує широкий набір типів запитань: множинний вибір, відкриті запитання, шкали оцінювання (у тому числі зі значками зірок), матриці, завантаження файлів тощо [12].

Серед ключових можливостей платформи – автоматична генерація звітів у вигляді кругових і стовпчастих діаграм, підтримка умовної логіки переходів між розділами (branch logic), а також безшовна інтеграція з Google Sheets для розширеного аналізу даних [11].

Важливою перевагою є спільне редагування форм у реальному часі – аналогічно до Google Docs – та збереження всіх даних у хмарі без ризику втрати. У 2025 році платформа отримала AI-функцію «Help me create», що дозволяє генерувати форму на основі текстового запиту або існуючих документів (Docs, Sheets, PDF), а також AI-підсумовування відкритих відповідей за допомогою Gemini [12].

До обмежень Google Forms належать базовий рівень аналітики порівняно зі спеціалізованими платформами, відсутність ігрових механік, а також залежність від стабільного підключення до мережі. Умовна логіка застосовується лише до запитань із вибором або випадаючими списками [11].

Основні характеристики Google Forms подано у таблиці 1.1 [10; 11; 12].

Таблиця 1.1 – Характеристика Google Forms

Параметр	**Характеристика**
Вартість	Безкоштовно (у складі Google Workspace)
Типи запитань	Множинний вибір, відкриті, шкали, завантаження файлів, матриці
Аналітика	Базові діаграми, автоматичний зведений звіт, AI-підсумки (Gemini)
Інтеграції	Google Sheets, Google Drive, Google Classroom
Ігрові механіки	Відсутні
Мобільний доступ	Так (браузер + мобільний додаток)

1.3.2 Mentimeter

Mentimeter – шведська платформа для інтерактивних презентацій, заснована у 2012 році. Сервіс позиціонує себе як інструмент для перетворення традиційних презентацій на двосторонній діалог з аудиторією [13]. Станом на 2025 рік платформа нараховує понад 100 мільйонів користувачів у більш ніж 220 країнах [13].

Основна концепція полягає в тому, що учасники приєднуються до сесії за кодом, посиланням або QR-кодом з будь-якого пристрою без необхідності реєстрації, а їхні відповіді відображаються в реальному часі у вигляді динамічних візуалізацій. Платформа підтримує 13 типів інтерактивних запитань: хмари слів, відкриті відповіді, шкали, рейтинги, вікторини з таблицями лідерів тощо [13].

У 2024 році Mentimeter додав AI Menti Builder – функцію генерації інтерактивних презентацій на основі текстових підказок, порядків денних або нотаток [14]. Дослідження показують, що інтеграція Mentimeter у навчальний процес позитивно впливає на залученість студентів, формативне оцінювання та загальне навчальне середовище [14].

До обмежень платформи відносять суттєво обмежену безкоштовну версію (лише кілька слайдів) і порівняно високу вартість платних тарифів, залежність від стабільного Інтернету, а також те, що інструмент більш орієнтований на корпоративних користувачів, ніж на освітні потреби [13].

1.3.3 Kahoot

Kahoot – норвезька ігрова навчальна платформа, офіційно запущена у вересні 2013 року на основі дослідницького проєкту «Lecture Quiz» Норвезького університету науки і технологій. Сервіс базується на принципі гейміфікації: навчальні активності перетворюються на конкурентні ігри з таблицями лідерів, балами та часовими обмеженнями [16].

Станом на 2025 рік платформа зафіксувала понад 9 мільярдів учасників у більш ніж 200 країнах і налічує понад 30 мільйонів публічних «кахутів» (ігор) [16]. Наукові дослідження підтверджують ефективність Kahoot: у дослідженні клінічних студентів 93% учасників надали перевагу вивченню складних тем через ігровий формат Kahoot [17].

У 2024–2025 роках платформа активно розвиває AI-інструменти: генератор кахутів з PDF-документів, Wikipedia-статей, YouTube-відео і текстових підказок; режим лекції (Lecture Mode) без ігрових елементів для інклюзивного навчання; режим точності (Accuracy Mode), що усуває бонус за швидкість [15]. Kahoot інтегрується з Zoom, Microsoft Teams, Google Classroom і PowerPoint [16].

Основні обмеження: конкурентний формат може бути стресовим для окремих учнів; платформа більш придатна для закріплення знань, ніж для поглибленого вивчення матеріалу; безкоштовна версія обмежена у кількості учасників команди [17].

1.3.4 SurveyMonkey

SurveyMonkey – одна з найстаріших і найвідоміших платформ для онлайн-опитувань, заснована у 1999 році. Станом на 2025 рік це повнофункціональний інструмент для дослідників, маркетологів і великих організацій, що надає потужний аналітичний функціонал [19].

Платформа підтримує широкий спектр типів запитань: від множинного вибору і зіркових рейтингів до матричних шкал і складних фільтрів. SurveyMonkey пропонує розширену логіку переходів, рандомізацію блоків запитань, налаштувальні теми та автоматичні звіти. Для поширення опитувань доступні канали: електронна пошта, соціальні мережі, вбудовані веб-форми [19].

Серед ключових переваг – корпоративний рівень безпеки (шифрування, відповідність GDPR і HIPAA в Enterprise-плані), розширена аналітика з візуалізацією даних і крос-табуляцією, а також інтеграції з понад 100 сервісами: Salesforce, Office 365, Google Drive, Slack та ін. [18, 19].

Головним недоліком є цінова політика: безкоштовний план суттєво обмежений за кількістю запитань і відповідей, а платні плани починаються від $39/місяць при річній оплаті. Відсутні ігрові механіки та функції проведення синхронних інтерактивних сесій [18].

1.3.5 Порівняльний аналіз платформ

Для систематизації отриманих відомостей у таблиці 1.2 наведено узагальнений порівняльний аналіз розглянутих платформ за ключовими параметрами [10–19].

Таблиця 1.2 – Порівняльний аналіз платформ

Параметр	**Google Forms**	**Mentimeter**	**Kahoot**	**SurveyMonkey**
Рік запуску	2008	2012	2013	1999
Безкоштовний план	Так (повний)	Обмежений	Є базовий	Дуже обмежений
Ігрова механіка	Ні	Частково	Так (основа)	Ні
Реальний час	Частково	Так	Так	Ні
AI-функції	Так (Gemini)	Так (AI Builder)	Так (AI gen.)	Частково
Аналітика	Базова	Середня	Базова	Розширена
Інтеграції	Google екосистема	Zoom, Teams	Zoom, Teams, GC	100+ сервісів
Цільова аудиторія	Широка	Бізнес/освіта	Освіта/бізнес	Бізнес/дослідники
Мобільний доступ	Так	Так	Так	Так

Проведений аналіз дозволяє зробити висновок, що кожна з розглянутих платформ має чітко виражену спеціалізацію. Google Forms є оптимальним вибором для безкоштовного збору даних в екосистемі Google, Mentimeter – для інтерактивних ділових презентацій, Kahoot – для ігрового навчання, а SurveyMonkey – для масштабних досліджень із розширеними аналітичними потребами.

Жодна з платформ не орієнтована одночасно на гейміфіковане тестування, детальну статистику та адаптивну логіку запитань у контексті освітніх установ, що підтверджує актуальність розробки спеціалізованої системи.

2 ПРОЄКТУВАННЯ ВЕБСИСТЕМИ

2.1 Архітектура системи: клієнт–серверна модель, модулі збору даних, аналітики та візуалізації

Вебсистема «ГолосХаб» проєктується на основі клієнт-серверної моделі, яка передбачає логічний поділ системи на три взаємопов’язані рівні (рис. 2.1): рівень представлення (клієнтська частина), рівень бізнес-логіки (серверна частина) та рівень збереження даних (база даних SQLite). Така трирівнева архітектура забезпечує чітку структуризацію компонентів системи, зменшує їхню взаємозалежність (слабку зв’язність), спрощує подальше масштабування та дозволяє здійснювати незалежне тестування кожного рівня.

На концептуальному рівні користувач взаємодіє із системою через вебінтерфейс, який формує запити до серверної частини. Сервер, у свою чергу, обробляє ці запити, виконує необхідну бізнес-логіку, взаємодіє з базою даних та повертає результати у форматі структурованих JSON-відповідей.

Клієнтська частина отримує ці дані та відображає їх у вигляді інтерфейсних компонентів або графічних елементів. Таким чином, у системі формується повний цикл обробки даних: від введення користувачем до збереження, аналізу та візуалізації результатів.

На рівні представлення знаходиться клієнтський інтерфейс, реалізований як набір UI-компонентів, через які користувач ініціює дії: перегляд опитувань, голосування або створення нових опитувань. Ці дії трансформуються у HTTP-запити (GET або POST у форматі JSON), які надходять до серверної частини системи.

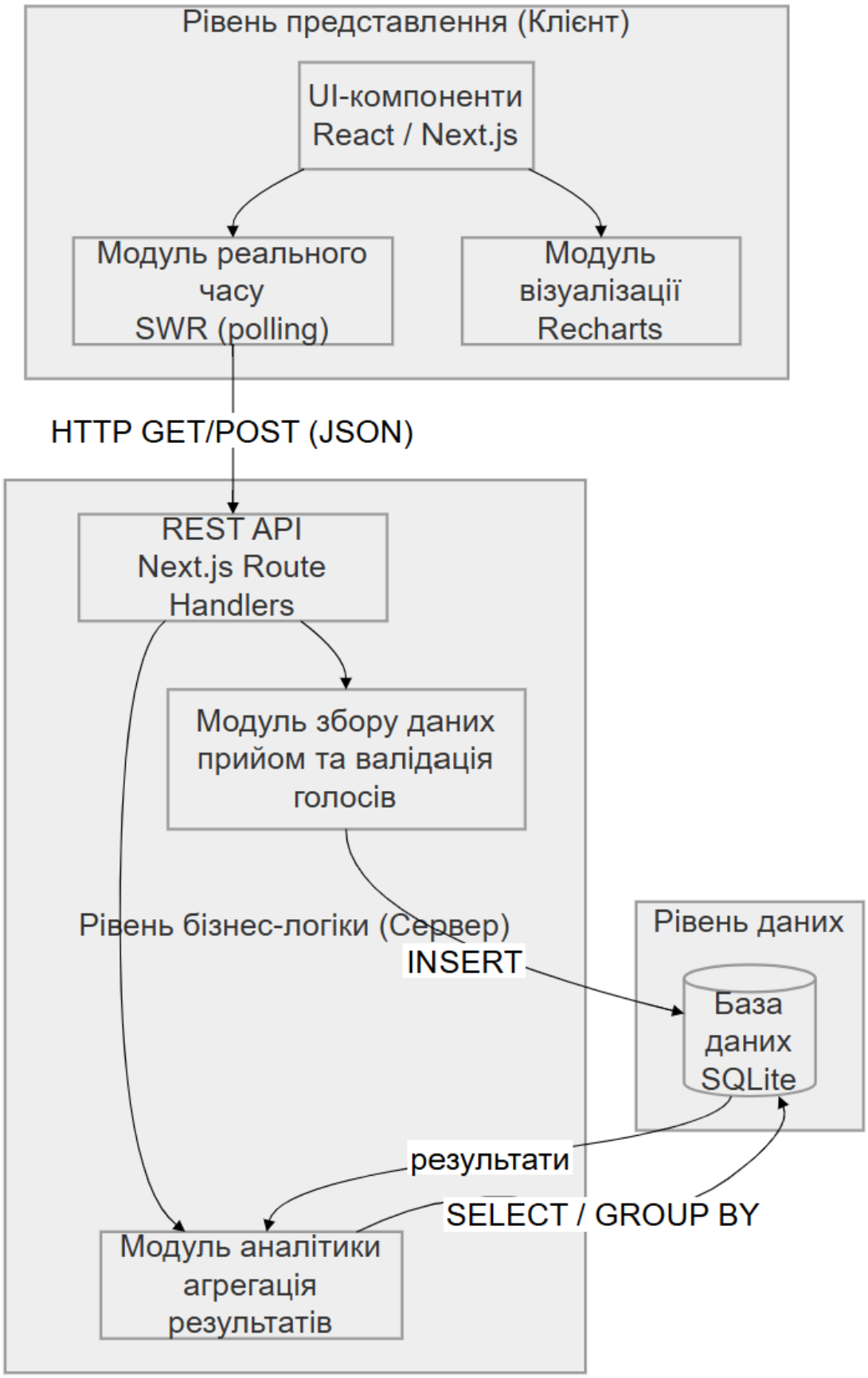


Рисунок 2.1 – Архітектурна схема клієнт-серверноївзаємодії вебсистеми «ГолосХаб»

Рівень бізнес-логіки приймає запити та виконує їхню обробку через REST API (Next.js Route Handlers). Тут реалізується основна логіка системи: перевірка коректності голосів, обробка запитів на створення опитувань, агрегація результатів та підготовка даних для аналітики. Далі сервер звертається до рівня даних, де виконується взаємодія з базою даних SQLite. На цьому етапі відбуваються операції INSERT для збереження нових даних та SELECT / GROUP BY для отримання агрегованих результатів.

Після виконання запитів база даних повертає результати на серверний рівень, який формує фінальну відповідь і передає її клієнту. Клієнтська частина, у свою чергу, оновлює інтерфейс та передає дані до модулів візуалізації, які відображають статистику у вигляді графіків і діаграм.

Отже, наведена архітектурна схема відображає загальну структуру вебсистеми та демонструє взаємодію між її основними рівнями. Проте для більш детального розуміння принципів функціонування системи необхідно розглянути складові компоненти кожного рівня та визначити їхнє функціональне призначення. З цією метою доцільно виокремити основні модулі вебсистеми, які забезпечують реалізацію передбачених проєктом функцій (табл. 2.1).

Таблиця 2.1 – Функціональні модулі вебсистеми «ГолосХаб»

Модуль	**Рівень системи**	**Призначення**	**Реалізація**
Модуль інтерфейсу користувача	Клієнтський рівень	Забезпечення взаємодії користувача із системою	React-компоненти (poll-list, poll-view, create-poll-form)
Модуль реального часу	Клієнтський рівень	Оновлення даних без перезавантаження сторінки	SWR (short polling, refreshInterval)
Модуль візуалізації даних	Клієнтський рівень	Графічне відображення результатів опитувань	Recharts (bar/pie charts)
Модуль збору даних	Серверний рівень	Прийом і валідація голосів користувачів	API: POST /api/polls/[id]/vote
Модуль аналітики	Серверний рівень	Обробка та агрегація результатів голосування	SQL: COUNT(), GROUP BY
Модуль доступу до даних	Серверний рівень	Взаємодія з базою даних SQLite	lib/db.ts, lib/queries.ts

Як видно з таблиці, система структурована таким чином, що кожен модуль виконує чітко визначену функцію в межах свого рівня. Клієнтські модулі відповідають за інтерфейсну взаємодію та візуалізацію даних, забезпечуючи зручність користувача та динамічне оновлення інформації. Серверні модулі реалізують бізнес-логіку системи, включаючи прийом голосів, їх перевірку та статистичну обробку. Окремо виділений модуль доступу до даних забезпечує централізовану роботу з базою даних, що підвищує структурованість і безпеку системи.

У проєктуванні передбачено механізм оновлення даних у реальному часі на основі short polling. Його суть полягає в тому, що клієнтський модуль періодично надсилає запити до сервера (наприклад, GET /api/polls/[id] через SWR з інтервалом у кілька секунд). Сервер у відповідь виконує агрегацію даних із бази та повертає актуальний стан результатів голосування.

Після отримання відповіді клієнт автоматично оновлює візуалізацію, що дозволяє користувачу спостерігати зміни в режимі майже реального часу без необхідності перезавантаження сторінки. Такий підхід є простішим за WebSocket-рішення, не потребує постійного з’єднання та є достатньо ефективним для проєктного рівня системи.

Отже, у процесі проєктування вебсистеми «ГолосХаб» було обрано клієнт-серверну архітектуру з трирівневою організацією, що включає рівень представлення, рівень бізнес-логіки та рівень даних. Запропонована структура забезпечує чіткий розподіл функцій між компонентами системи, сприяє підвищенню її модульності та створює передумови для подальшого масштабування й супроводу. Визначено основні функціональні модулі, відповідальні за збір та обробку голосів, аналітику результатів і їх візуалізацію. Крім того, передбачено механізм оновлення даних у режимі, наближеному до реального часу, що дозволить забезпечити оперативне відображення результатів опитувань. Таким чином, розроблена архітектура відповідає функціональним вимогам проєктованої вебсистеми та слугує основою для подальшого вибору технологій і деталізації структури даних.

2.2 Вибір технологій

Вибір технологій для проєктування вебсистеми «ГолосХаб» здійснювався з урахуванням функціональних і нефункціональних вимог, визначених у попередньому розділі. Основними критеріями відбору стали продуктивність системи, простота реалізації та розгортання, підтримка роботи з даними в режимі реального часу, зручність подальшого супроводу програмного коду та відповідність навчальному характеру проєкту. Обраний технологічний стек дозволяє реалізувати всі необхідні функції системи, забезпечуючи при цьому достатній рівень надійності та масштабованості.

Для розробки вебсистеми доцільно використовувати мову програмування TypeScript [20], яка є надбудовою над JavaScript і забезпечує статичну типізацію. Застосування TypeScript сприяє зменшенню кількості помилок на етапі розробки, підвищує читабельність програмного коду та полегшує його подальшу підтримку. Наявність механізму перевірки типів ще до виконання програми є особливо важливою при створенні багатокомпонентних вебзастосунків, де відбувається постійний обмін даними між клієнтською та серверною частинами.

Як основу для побудови вебсистеми обрано фреймворк Next.js [21], що базується на бібліотеці React та надає розширені можливості для створення сучасних вебзастосунків. Використання архітектури App Router дозволяє поєднати клієнтську та серверну логіку в межах одного проєкту, що спрощує організацію коду та зменшує складність розробки. Крім того, Next.js підтримує серверний рендеринг, генерацію API-маршрутів за допомогою Route Handlers та ефективне керування маршрутизацією.

Для побудови користувацького інтерфейсу доцільно застосовувати бібліотеку React [22], яка реалізує компонентний підхід до розробки. Компонентна структура забезпечує повторне використання окремих елементів інтерфейсу, спрощує підтримку коду та дозволяє ефективно оновлювати лише ті частини сторінки, які зазнали змін. Завдяки декларативному принципу побудови інтерфейсу розробка стає більш передбачуваною та зрозумілою.

Для стилізації елементів інтерфейсу обрано комбінацію Tailwind CSS [23] та бібліотеки компонентів shadcn/ui [24]. Tailwind CSS забезпечує швидке створення адаптивного дизайну за допомогою готових утилітарних класів, що значно скорочує час розробки. Використання shadcn/ui дозволяє застосовувати готові інтерфейсні компоненти, які відповідають сучасним вимогам доступності та забезпечують єдиний стиль оформлення всієї системи.

Зберігання даних у проєктованій системі передбачається за допомогою SQLite [25]. Дана система керування базами даних є вбудованою, не потребує встановлення окремого серверного програмного забезпечення та дозволяє зберігати всю базу даних у вигляді одного файлу. Це значно спрощує процес налаштування та розгортання застосунку. Водночас SQLite підтримує повноцінний SQL-функціонал, включаючи зовнішні ключі, індекси, агрегатні функції та транзакції, що робить її достатньо потужним інструментом для реалізації навчального проєкту.

Обмін даними між клієнтською та серверною частинами, а також оновлення результатів голосування в режимі, наближеному до реального часу, пропонується реалізувати за допомогою бібліотеки SWR [26]. Її використання забезпечує автоматичне кешування отриманих даних, синхронізацію стану між компонентами та можливість періодичного повторного опитування сервера. Такий підхід є простішим у реалізації порівняно з використанням WebSocket-з’єднань і повністю відповідає вимогам навчального проєкту.

Для графічного представлення результатів голосувань доцільно використовувати бібліотеку Recharts [27], яка побудована на основі React та забезпечує створення інтерактивних діаграм за декларативним принципом. Її функціональні можливості дозволяють реалізовувати стовпчикові, кругові та інші типи діаграм, необхідні для візуального аналізу результатів опитувань.

З метою покращення користувацького досвіду передбачено використання бібліотеки Sonner [28], яка забезпечує відображення коротких інформаційних повідомлень (toast-сповіщень) про результати виконання певних дій. Зокрема, такі повідомлення можуть використовуватися для інформування про успішне

створення опитування, фіксацію голосу або виникнення помилки під час обробки запиту.

Отже, обраний технологічний стек забезпечує реалізацію всіх функціональних вимог проєктованої вебсистеми та відповідає поставленим критеріям продуктивності, простоти розгортання та зручності подальшої підтримки. Застосування сучасних вебтехнологій створює основу для ефективної реалізації системи онлайн-голосувань та опитувань з аналітикою результатів у реальному часі.

2.3 Моделювання даних: структура користувачів, опитувань, варіантів відповідей, результатів

Одним із ключових етапів проєктування вебсистеми є моделювання даних, оскільки саме від правильно побудованої структури бази даних залежить ефективність зберігання інформації, швидкість виконання запитів та забезпечення цілісності даних. На цьому етапі визначаються основні сутності предметної області, їхні атрибути та взаємозв’язки між ними. Для вебсистеми онлайн-голосувань та опитувань «ГолосХаб» логічна модель даних була сформована з урахуванням функціональних вимог, визначених у попередніх розділах, а саме необхідності створення опитувань, збереження варіантів відповідей, фіксації голосів користувачів та формування аналітичних результатів.

Запропонована модель даних включає три основні сутності: опитування (Poll), варіанти відповідей (Option) та голоси (Vote). Водночас у системі передбачається використання спрощеної моделі користувачів. Замість повноцінної реєстрації та авторизації учасники голосування ідентифікуються за допомогою унікального анонімного ідентифікатора voterId, який генерується та зберігається на стороні клієнта. Такий підхід дозволяє зберегти конфіденційність

користувачів і водночас мінімізувати ризик повторного голосування в межах одного опитування.

Центральною сутністю моделі є опитування, яке містить основну інформацію про питання, що виноситься на голосування. Кожне опитування характеризується унікальним ідентифікатором, текстом питання, додатковим описом, режимом голосування та датою створення.

Передбачено підтримку двох режимів участі: вибір одного варіанта відповіді або можливість обрати декілька варіантів одночасно. Це дозволяє використовувати систему як для класичних голосувань, так і для більш гнучких форматів анкетування.

Структуру сутності «Опитування» наведено в таблиці 2.2.

Таблиця 2.2 – Структура таблиці polls

Поле	**Тип даних**	**Обмеження**	**Опис**
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор опитування
question	TEXT	NOT NULL	Текст питання
description	TEXT	NULL	Додатковий опис опитування
multiple	INTEGER	NOT NULL DEFAULT 0	Режим голосування: 0 – один варіант, 1 – декілька варіантів
created_at	TEXT	NOT NULL	Дата та час створення опитування

Як видно з таблиці 2.2, основним атрибутом є поле question, яке містить текст запитання. Поле description є необов’язковим і використовується для надання користувачам додаткових пояснень щодо мети або умов проведення голосування. Атрибут multiple визначає тип голосування та впливає на подальшу логіку обробки голосів.

Наступною сутністю є варіанти відповідей, які безпосередньо пов’язані з конкретним опитуванням. Одне опитування може містити декілька варіантів відповідей, при цьому кожен варіант належить лише одному опитуванню.

Структуру сутності «Варіанти відповідей» подано в таблиці 2.3.

Таблиця 2.3 – Структура таблиці options

Поле	Тип даних	Обмеження	Опис
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор варіанта відповіді
poll_id	INTEGER	NOT NULL, FOREIGN KEY	Посилання на опитування
text	TEXT	NOT NULL	Текст варіанта відповіді

Наявність зовнішнього ключа poll_id забезпечує зв’язок між таблицями опитувань та варіантів відповідей. Такий підхід дозволяє підтримувати цілісність даних і гарантує, що кожен варіант відповіді буде належати лише до існуючого опитування.

Для фіксації результатів голосування використовується сутність «Голос», яка відображає факт вибору певного варіанта відповіді конкретним учасником.

Структуру таблиці голосів наведено в таблиці 2.4.

Таблиця 2.4 – Структура таблиці votes

Поле	Тип даних	Обмеження	Опис
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор голосу
poll_id	INTEGER	NOT NULL, FOREIGN KEY	Посилання на опитування
option_id	INTEGER	NOT NULL, FOREIGN KEY	Посилання на обраний варіант відповіді
voter_id	TEXT	NOT NULL	Анонімний ідентифікатор учасника
created_at	TEXT	NOT NULL	Дата та час подання голосу

Збереження ідентифікатора voter_id дозволяє реалізувати механізм контролю повторних голосувань без необхідності створення окремої таблиці користувачів. При цьому особисті дані учасників не збираються та не зберігаються, що позитивно впливає на забезпечення конфіденційності та спрощує архітектуру системи.

Для наочного представлення взаємозв’язків між сутностями було розроблено ER-модель бази даних (рис. 2.2).

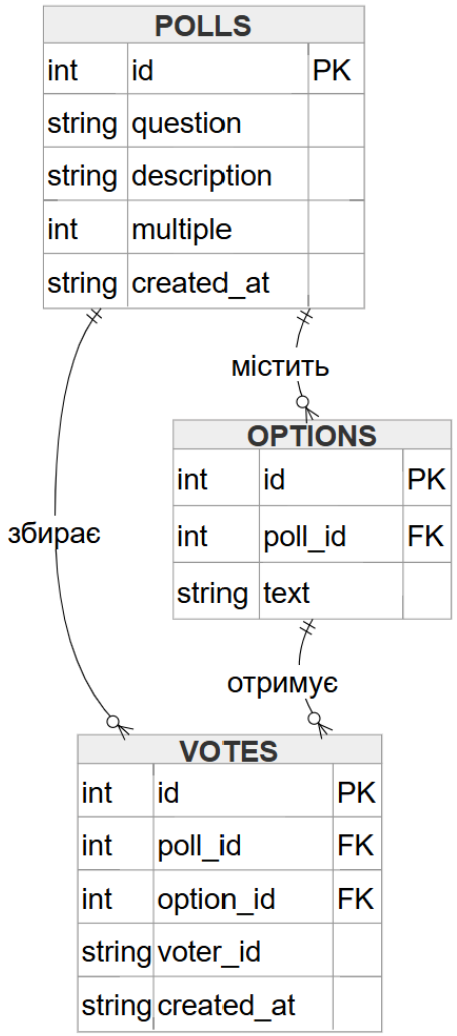


Рисунок 2.2 – ER-діаграма логічної моделі даних вебсистеми «ГолосХаб»

Наведена діаграма відображає основні зв’язки між сутностями системи. Сутність POLLS є центральною та пов’язана із сутністю OPTIONS зв’язком типу «один-до-багатьох», оскільки одне опитування може містити декілька варіантів відповідей. Водночас кожен варіант відповіді належить лише одному опитуванню.

Між сутностями OPTIONS та VOTES також реалізовано зв’язок «один-до-багатьох»: один варіант відповіді може отримати необмежену кількість голосів, тоді як кожен голос фіксує вибір лише одного варіанта. Крім того, таблиця голосів містить посилання на відповідне опитування, що спрощує формування аналітичних запитів і прискорює обчислення статистичних показників.

Для забезпечення цілісності даних у проєктованій системі передбачається використання зовнішніх ключів із каскадним видаленням записів. Це означає, що при видаленні опитування автоматично видалятимуться пов’язані з ним варіанти відповідей і голоси, що унеможливлює появу так званих «осиротілих» записів у базі даних.

З метою підвищення продуктивності системи також доцільним є створення індексів за полями poll_id та option_id, які найчастіше використовуватимуться під час виконання запитів на отримання статистики голосування. Наявність індексів сприятиме скороченню часу виконання операцій агрегації та забезпечить оперативне формування результатів навіть у разі збільшення кількості записів.

Отже, запропонована модель даних охоплює всі основні об’єкти предметної області та забезпечує ефективне зберігання інформації про опитування, варіанти відповідей і результати голосування. Використання зовнішніх ключів, індексів та механізму анонімної ідентифікації учасників дозволяє забезпечити цілісність даних, належний рівень продуктивності та відповідність функціональним вимогам проєктованої вебсистеми.

2.4 UML-діаграми

Для формалізації процесів функціонування вебсистеми «ГолосХаб» на етапі проєктування було використано UML-діаграми, які дають змогу відобразити систему з різних точок зору. Застосування UML-моделювання дозволяє структурувати вимоги до програмного продукту, визначити взаємодію між користувачами та системою, описати внутрішню організацію основних сутностей, а також простежити послідовність виконання операцій під час роботи із системою. У межах проєктування вебсистеми було розроблено діаграму варіантів використання (Use Case Diagram), діаграму класів (Class Diagram) та діаграму послідовності (Sequence Diagram).

Діаграма сценаріїв використання (рис. 2.3) призначена для опису функціональних можливостей системи з позиції кінцевих користувачів. Вона демонструє, які дії можуть виконувати актори системи та яким чином вони взаємодіють із програмним продуктом.

У вебсистемі «ГолосХаб» передбачено двох основних акторів: Автор опитування та Учасник. Особливістю проєкту є відсутність механізму реєстрації та авторизації, тому ролі користувачів визначаються не наявністю облікового запису, а набором виконуваних дій. Один і той самий користувач може виступати як автор опитування, створюючи власні голосування, або як учасник, беручи участь у вже створених опитуваннях.

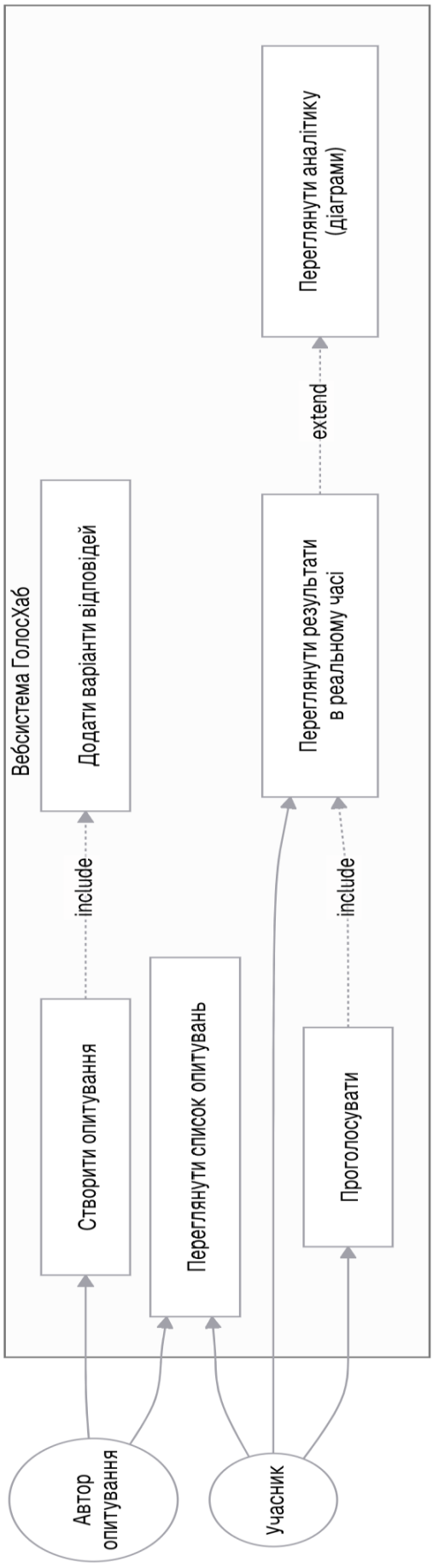


Рисунок 2.3 – Use Case-діаграма вебсистеми «ГолосХаб»

Представлена діаграма відображає основні функціональні можливості системи. Автор опитування може створювати нові голосування та додавати до них варіанти відповідей. Учасник має можливість переглядати перелік доступних опитувань, брати участь у голосуванні та аналізувати отримані результати. Крім того, передбачено перегляд результатів у режимі, наближеному до реального часу, а також їх візуальне представлення у вигляді діаграм.

Для більш детального опису сценаріїв використання сформовано таблицю основних варіантів взаємодії користувачів із системою (табл. 2.5).

Таблиця 2.5 – Основні сценарії використання вебсистеми «ГолосХаб»

Сценарій	**Актор**	**Короткий опис**
Створити опитування	Автор опитування	Введення питання, опису, вибір режиму голосування та додавання варіантів відповідей
Переглянути список опитувань	Учасник	Ознайомлення з доступними для голосування опитуваннями
Проголосувати	Учасник	Вибір одного або кількох варіантів відповіді та надсилання голосу
Переглянути результати	Учасник	Отримання актуальної інформації про кількість голосів і відсотковий розподіл
Переглянути аналітику	Учасник	Аналіз результатів за допомогою графічних засобів візуалізації

У таблиці наведено основні сценарії взаємодії користувачів із вебсистемою «ГолосХаб», які відображають її базову функціональність. Вона демонструє, які дії доступні двом ключовим акторам – автору опитувань та учаснику – і які операції вони можуть виконувати в межах системи. Зокрема, автор опитування відповідає за створення голосувань і налаштування їх структури, тоді як учасник взаємодіє з уже створеними опитуваннями, здійснює голосування та аналізує отримані результати.

Крім того, таблиця показує повний цикл роботи з опитуванням: від його створення та публікації до перегляду статистики та візуальної аналітики. Таким чином, вона узагальнює ключові функції системи та відображає логіку взаємодії користувачів із різними етапами життєвого циклу опитування.

Для відображення логічної структури системи та взаємозв'язків між її основними компонентами було побудовано діаграму класів (рис. 2.4). Вона демонструє ключові сутності предметної області, їхні атрибути, методи та асоціації між ними.

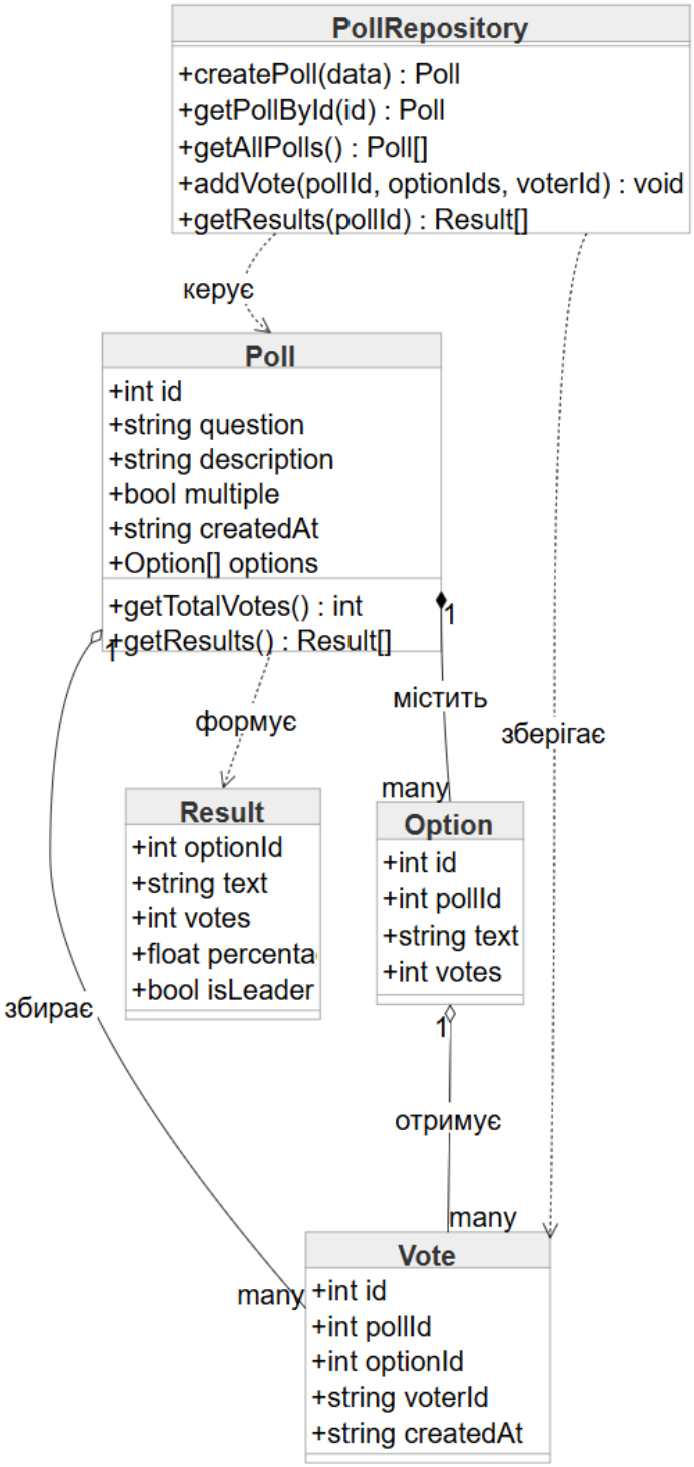


Рисунок 2.4 – Діаграма класів вебсистеми «ГолосХаб»

В основі моделі лежить клас Poll, який описує опитування та містить інформацію про його параметри. Кожне опитування може включати декілька варіантів відповідей, що реалізовано через зв’язок «один-до-багатьох» із класом Option.

Клас Vote представляє окремий факт голосування та забезпечує зв’язок між конкретним варіантом відповіді та анонімним ідентифікатором користувача. Наявність атрибута voterId дозволяє уникнути повторного голосування без використання складних механізмів авторизації.

Окрему роль відіграє клас Result, який не відповідає окремій таблиці бази даних, а формується динамічно модулем аналітики. Він містить обчислені показники, зокрема кількість голосів, відсотковий розподіл та інформацію про варіант-лідер.

Доступ до даних інкапсульовано в класі PollRepository, який виконує функції посередника між бізнес-логікою та базою даних. Такий підхід сприяє дотриманню принципу розподілу відповідальності та підвищує підтримуваність системи.

Для моделювання динамічної поведінки системи було використано діаграму послідовності (рис. 2.5). Вона відображає процес взаємодії між користувачем, серверними компонентами та базою даних під час подання голосу та подальшого оновлення результатів.

Представлена діаграма демонструє два взаємопов’язані процеси. Перший процес охоплює безпосереднє подання голосу користувачем. Після отримання запиту сервер виконує перевірку на наявність попереднього голосування з використанням анонімного ідентифікатора. У разі успішного проходження перевірки інформація про голос зберігається в базі даних, а користувач отримує підтвердження виконаної дії.

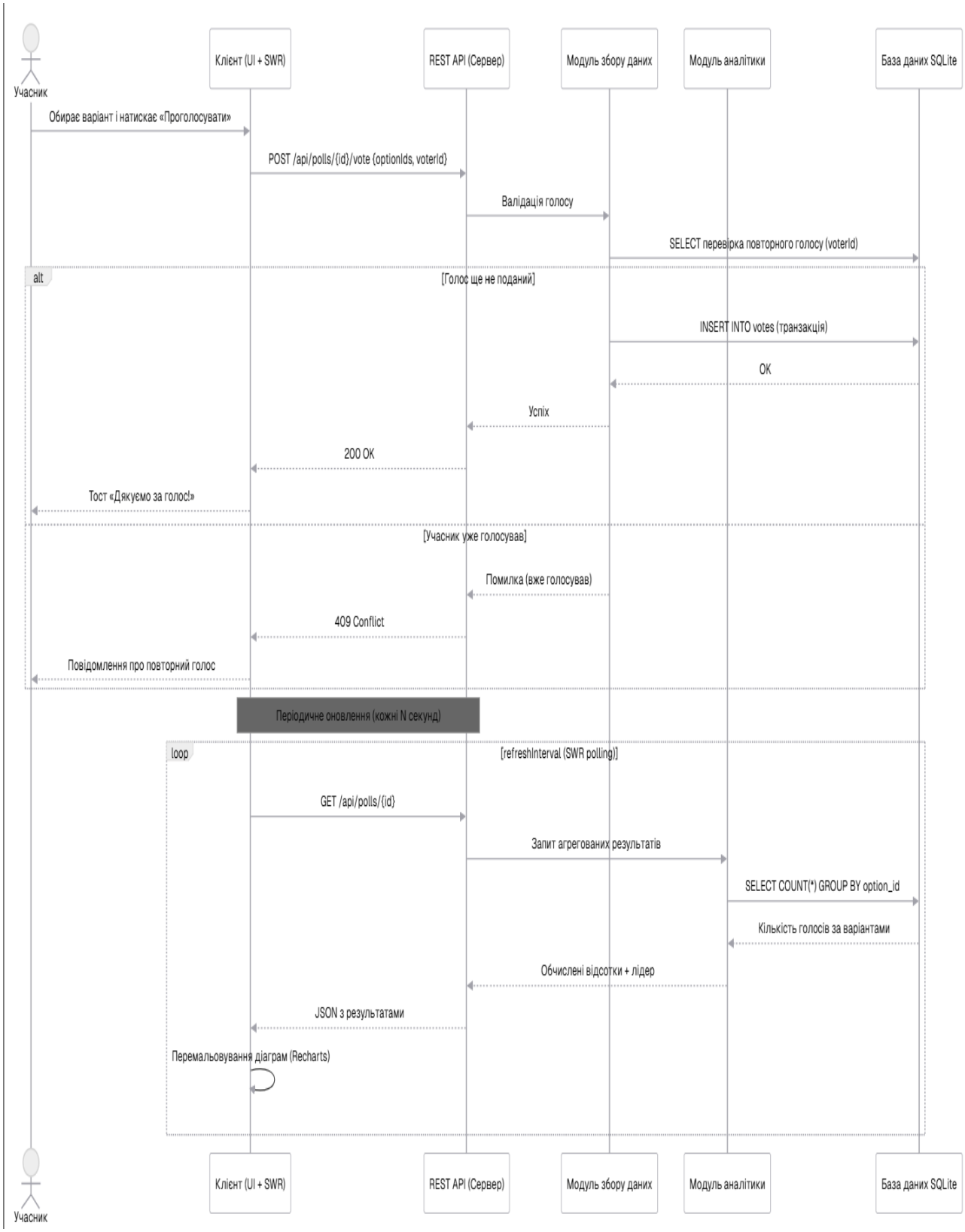


Рисунок 2.5 – Діаграма послідовності процесу голосування та оновлення результатів

Другий процес відображає механізм автоматичного оновлення результатів голосування. Завдяки використанню періодичного опитування клієнтська частина системи регулярно надсилає запити до сервера для отримання актуальної статистики. Модуль аналітики виконує агрегацію даних, обчислює необхідні показники та передає їх клієнтському застосунку для повторної побудови діаграм. Такий підхід забезпечує своєчасне оновлення інформації без необхідності ручного перезавантаження сторінки.

Отже, використання UML-діаграм дало змогу комплексно описати вебсистему «ГолосХаб» на етапі проєктування. Діаграма сценаріїв використання відобразила функціональні можливості системи та взаємодію користувачів із нею, діаграма класів – структуру основних сутностей і зв’язків між ними, а діаграма послідовності – логіку обробки голосів та оновлення результатів у режимі реального часу. Сукупність цих моделей забезпечує цілісне уявлення про майбутню систему та створює основу для її подальшої реалізації.

3 РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ

3.1 Реалізація інтерфейсу користувача: створення опитувань, участь у голосуванні

Інтерфейс користувача вебсистеми «ГолосХаб» реалізовано на основі компонентного підходу React із використанням бібліотеки shadcn/ui, що забезпечує узгоджену дизайн-систему та високу швидкість розробки інтерфейсних елементів. Уся структура побудована за принципом mobile-first, що означає пріоритет адаптації під мобільні пристрої з подальшим розширенням для більших екранів. Архітектурно інтерфейс поділено на окремі сторінки та перевикористовувані компоненти, що забезпечує модульність, повторне використання коду та спрощене масштабування системи (табл. 3.1).

Таблиця 3.1 – Структура сторінок та компонентів

Маршрут	**Сторінка**	**Ключові компоненти**	**Призначення**
/	Головна	SiteHeader, PollList, PollCard	Відображення списку опитувань та базової інформації
/create	Створення опитування	CreatePollForm	Формування нового опитування користувачем
/poll/[id]	Сторінка опитування	PollView, ResultsChart	Участь у голосуванні та перегляд результатів

Таблиця відображає структуру маршрутизації вебсистеми та зв’язок між URL-адресами, відповідними сторінками, ключовими компонентами інтерфейсу та їх функціональним призначенням. Вона показує, як система організовує навігацію: головна сторінка забезпечує перегляд списку опитувань, сторінка створення призначена для формування нового опитування, а сторінка конкретного опитування об’єднує процес голосування та перегляд результатів у вигляді візуалізацій.

Процес створення опитування (рис. 3.1) реалізовано через компонент CreatePollForm, який є основною точкою входу для формування нових опитувань у системі. Користувач взаємодіє з формою, вводячи текст питання, додатковий опис (за потреби), обираючи режим голосування (одновибірковий або мультивибірковий), а також динамічно додаючи або видаляючи варіанти відповідей.

На рівні клієнта передбачено первинну валідацію введених даних, що дозволяє перевірити коректність форми ще до відправки запиту на сервер. Після підтвердження дані передаються через HTTP-запит POST /api/polls, де відбувається їх подальша обробка.

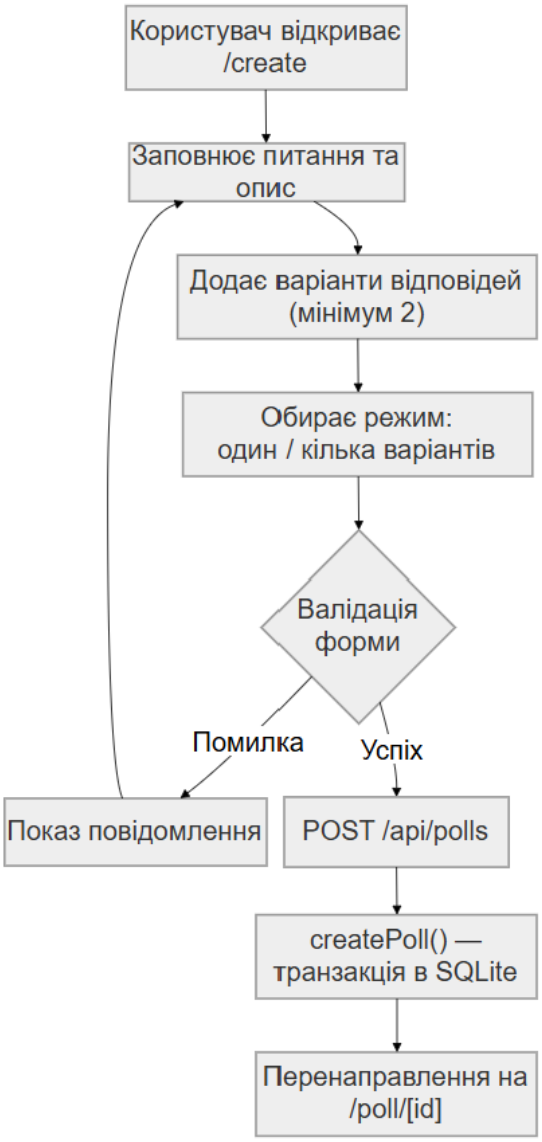


Рисунок 3.1 – Діаграма алгоритму створення опитування

Після отримання запиту серверна функція createPoll() виконує запис даних у базу SQLite в межах транзакції, що гарантує цілісність даних: або створюються всі елементи опитування (питання та варіанти відповідей), або ж у разі помилки не створюється жоден запис. Такий підхід забезпечує атомарність операцій та захист від неконсистентного стану бази даних.

Після створення опитування користувач переходить до сторінки PollView, де реалізовано механізм голосування. Відображення інтерфейсу залежить від параметра multiple: якщо дозволено один варіант відповіді, використовуються радіокнопки, якщо кілька – чекбокси. Це дозволяє гнучко адаптувати UI до різних типів опитувань.

Після вибору варіанта користувач надсилає голос, який обробляється сервером і зберігається у таблиці голосів. Після успішного голосування інтерфейс автоматично перемикається у режим перегляду результатів, де відображаються відсотковий розподіл відповідей, прогрес-бари та діаграма.

Стани інтерейсів описані в таблиці 3.2.

Таблиця 3.2 – Стан інтерфейсу під час голосування

Стан інтерфейсу	**Умова**	**Відображення**
Голосування	Користувач ще не голосував	Список варіантів та кнопка «Проголосувати»
Результати	Голос уже подано (hasVoted = true)	Відсотки, прогрес-бари, графік розподілу, позначення лідера
Помилка	Невірні дані або повторне голосування	Повідомлення про помилку (Sonner toast)

Після відправки голосу виконується перевірка на сервері для запобігання повторному голосуванню, зокрема через ідентифікатор voterId. У разі повторної спроби система повертає помилку, яка обробляється на клієнті у вигляді сповіщення.

Таким чином, реалізація інтерфейсу користувача у системі «ГолосХаб» забезпечує повний цикл взаємодії – від створення опитування до голосування та

перегляду результатів – із чітким розділенням логіки на клієнтську та серверну частини, що підвищує масштабованість і підтримуваність системи.

3.2 Механізм збору та обробки голосів у реальному часі

Реалізація механізму збору та обробки голосів у системі «ГолосХаб» базується на поєднанні серверної логіки обробки подій голосування та клієнтського механізму періодичного оновлення даних. Такий підхід дозволяє забезпечити ефект «реального часу» без використання складних технологій постійного з’єднання, зокрема WebSocket, що є доцільним у межах навчального проєкту з використанням SQLite.

Перед реалізацією системи було розглянуто основні підходи до оновлення даних у реальному часі у вебзастосунках (табл. 3.3). Вони відрізняються складністю впровадження, навантаженням на сервер та рівнем «миттєвості» оновлення даних.

Таблиця 3.3 – Підходи до реалізації «реального часу»

Підхід	**Принцип роботи**	**Переваги**	**Недоліки**	**Доцільність для проєкту**
Short Polling	Клієнт періодично надсилає HTTP-запити	Простота реалізації, сумісність з будь-яким бекендом	Додаткове навантаження, затримка оновлення	Обрано
Long Polling	Сервер утримує запит до появи нових даних	Менше зайвих запитів	Складніша серверна логіка	Надлишковий
WebSocket	Постійне двостороннє з’єднання	Миттєва синхронізація	Складна інфраструктура	Надлишковий для SQLite-проєкту
SSE	Потік подій від сервера до клієнта	Простіший за WebSocket	Обмежена гнучкість	Альтернативний варіант

У межах даного проєкту обрано підхід Short Polling, оскільки він найкраще відповідає навчальним цілям та не потребує додаткової серверної

інфраструктури. Реалізація здійснюється через бібліотеку SWR, яка автоматизує періодичні запити, кешування та повторну валідацію даних, що значно спрощує розробку.

Механізм роботи системи можна умовно поділити на два незалежні процеси: запис голосу користувача та отримання оновлених результатів через циклічні запити (рис. 3.2). Їхня взаємодія забезпечує синхронне оновлення даних у всіх відкритих клієнтах.

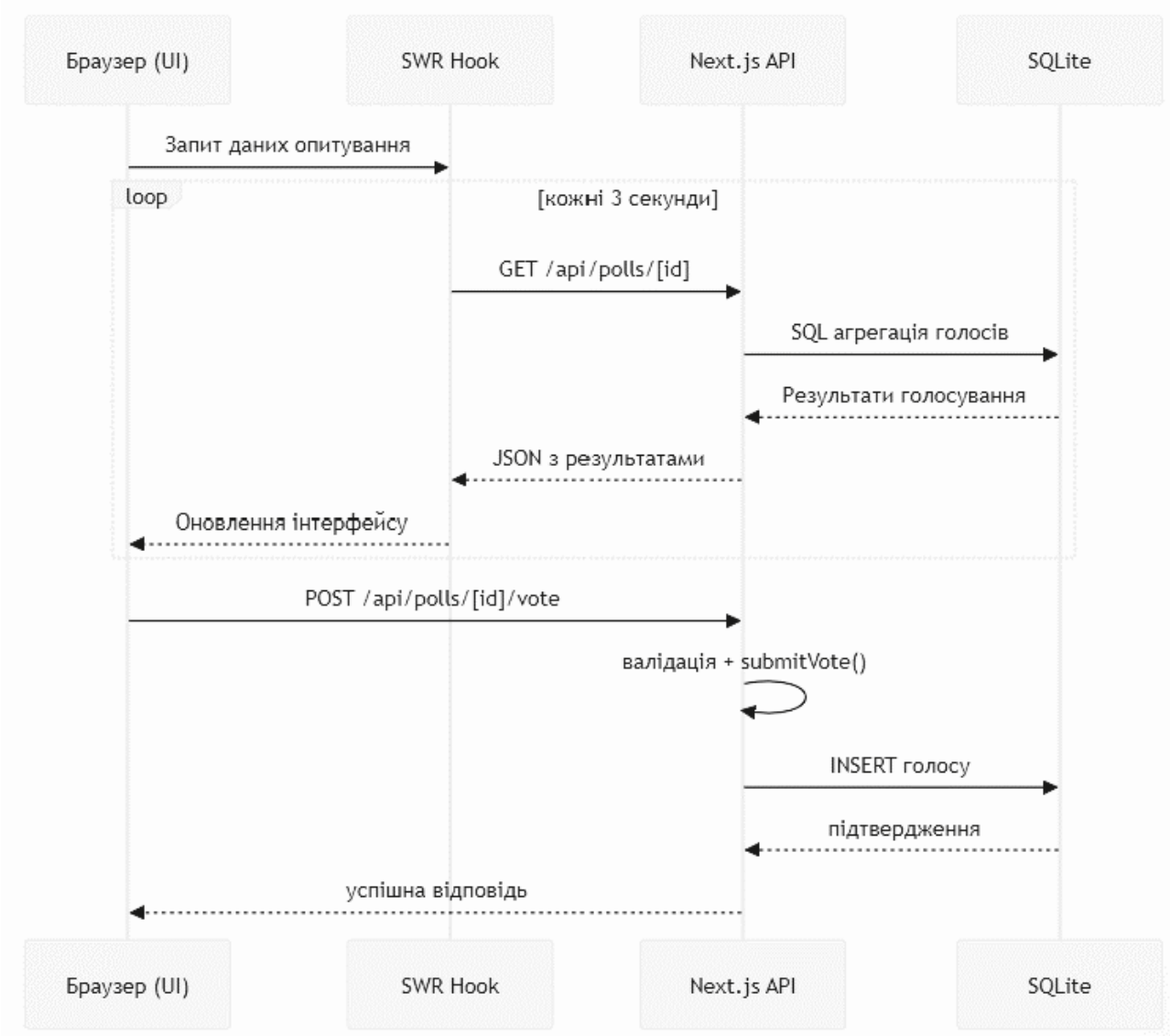


Рисунок 3.2 – Діаграма потоку даних у реальному часі

Схема sequenceDiagram показує повний цикл взаємодії між інтерфейсом користувача та серверною частиною системи під час голосування в реальному часі.

На першому етапі браузер (UI) через SWR-хук регулярно запитує дані опитування з інтервалом у 3 секунди. SWR надсилає GET-запит до Next.js API, який звертається до SQLite-бази даних, виконує агрегацію голосів і повертає актуальні результати у форматі JSON. Отримані дані передаються назад у UI, що забезпечує автоматичне оновлення інтерфейсу без перезавантаження сторінки.

Другий процес – це безпосереднє голосування. Коли користувач обирає варіант і надсилає голос, UI виконує POST-запит до API. Сервер проводить валідацію даних, викликає функцію збереження голосу (submitVote()), записує результат у SQLite через INSERT-запит і підтверджує успішне збереження. Після цього UI отримує відповідь про успішне голосування.

Обробка голосів реалізована через API-ендпоінт POST /api/polls/[id]/vote. Його основне завдання полягає у прийомі даних від клієнта, їх валідації та подальшому збереженні в базі даних.

У коді (додаток Б.1) реалізовано такі етапи обробки:

По-перше, з динамічного маршруту витягується ідентифікатор опитування. Далі сервер зчитує тіло запиту та нормалізує вхідні дані, перетворюючи optionIds у масив рядкових значень. Також перевіряється або створюється анонімний ідентифікатор користувача voterId, який зберігається на клієнті та дозволяє запобігати повторному голосуванню.

Після цього виконується базова валідація: перевіряється наявність вибраних варіантів відповіді. У разі помилки сервер повертає відповідний HTTP-код. Якщо дані коректні, викликається функція submitVote(), яка відповідає за запис голосу у базу даних.

Важливою особливістю реалізації є використання export const dynamic = "force-dynamic", що вимикає кешування відповідей Next.js і гарантує, що кожен запит буде оброблений безпосередньо сервером.

Функція submitVote() (додаткок Б.2) виконує ключову роль у серверній частині системи. Вона реалізує транзакційний запис голосів у базу SQLite та забезпечує цілісність даних.

Спочатку виконується перевірка існування опитування та визначається його режим роботи – одиночний або множинний вибір. У випадку одиночного режиму система автоматично обмежує кількість вибраних варіантів одним елементом, навіть якщо клієнт надіслав декілька значень.

Далі перевіряється коректність кожного варіанта відповіді через SQL-запит до таблиці options, що унеможливлює голосування за неіснуючі або сторонні варіанти.

Запис голосів здійснюється в межах транзакції (BEGIN / COMMIT / ROLLBACK). Це означає, що всі голоси або зберігаються повністю, або не зберігаються взагалі у разі виникнення помилки. Такий підхід забезпечує атомарність операцій і захист від часткових записів у базі даних.

Читання актуального стану опитування виконується через GET /api/polls/[id] (додаток Б.3). Саме цей ендпоінт використовується клієнтом для періодичного оновлення даних.

У межах серверної логіки виконується SQL-агрегація голосів за допомогою функції COUNT(*) та групування за option_id. Це дозволяє отримати актуальну кількість голосів для кожного варіанту без додаткової обробки на клієнті.

Отримані дані одразу перетворюються у структурований JSON-формат, який включає список варіантів, кількість голосів та розраховані значення для побудови діаграм.

На клієнтській стороні використовується бібліотека SWR, яка реалізує механізм short polling. У коді (додаток Б.4) визначено інтервал refreshInterval, який запускає автоматичне оновлення даних кожні кілька секунд.

Це означає, що браузер періодично надсилає запит до сервера, отримує оновлені результати та автоматично оновлює компоненти інтерфейсу без

перезавантаження сторінки. Таким чином користувач бачить актуальний стан голосування майже миттєво.

Додатково реалізовано локальне оновлення стану після відправки голосу: після успішного POST-запиту інтерфейс одразу перемикається у режим перегляду результатів, що підсилює відчуття інтерактивності.

Для аналізу процесу взаємодії між користувачем, інтерфейсом, серверною частиною та базою даних у системі голосування використано діаграму послідовностей (sequenceDiagram). Вона дозволяє наочно відобразити два ключові сценарії роботи системи: надсилання голосу та отримання оновлених результатів у реальному часі (рис. 3.3).

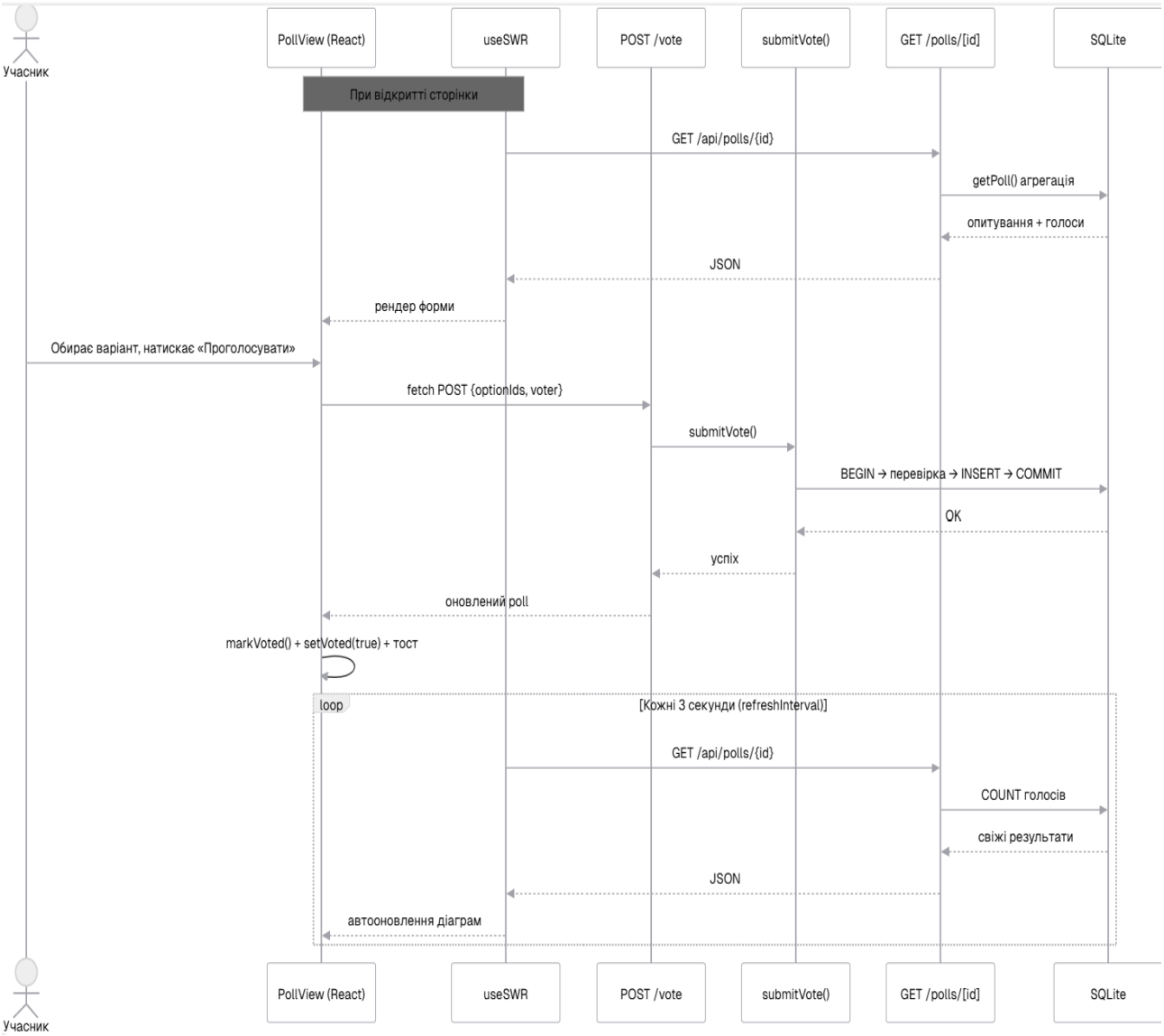


Рисунок 3.3 – Діаграма життєвого циклу голосу

Діаграма демонструє, що користувач через інтерфейс обирає варіант відповіді, після чого UI надсилає POST-запит до API. Сервер обробляє запит, виконує запис голосу в базу даних SQLite та повертає підтвердження успішної операції.

Паралельно з цим у системі працює циклічний процес оновлення даних: кожні 3 секунди UI надсилає GET-запит до API для отримання актуального стану опитування. API виконує підрахунок голосів у базі даних і повертає оновлені результати у форматі JSON, які одразу відображаються в інтерфейсі користувача.

Таким чином діаграма відображає поєднання подієвого підходу (голосування) та періодичного опитування (оновлення результатів), що забезпечує ефект «реального часу» у системі.

Отже, розглянуто механізм збору та обробки голосів у реальному часі в системі «ГолосХаб». Проаналізовано основні підходи до реалізації оновлення даних, серед яких обґрунтовано вибір short polling як найбільш доцільного для навчального проєкту рішення. Описано архітектуру взаємодії між клієнтом, API та базою даних, яка поєднує подієвий запис голосів через HTTP-запити з періодичним отриманням оновлених результатів за допомогою SWR. Це забезпечує актуальність даних без використання складних технологій постійного з’єднання. Таким чином, реалізований підхід дозволяє досягти ефекту «реального часу» при збереженні простоти архітектури, стабільності роботи та зручності подальшого масштабування системи.

3.3 Аналітика результатів: побудова графіків, діаграм, статистичних звітів

Аналітичний модуль системи «ГолосХаб» відповідає за обробку сирих даних голосування та їх перетворення у зручні для сприйняття статистичні показники й візуальні елементи. Уся логіка винесена в окремий файл

lib/analytics.ts, що реалізує принцип розділення бізнес-логіки та UI, підвищуючи тестованість і повторне використання функцій.

Аналітика побудована на основі набору чистих функцій, які не мають побічних ефектів і працюють лише з переданими даними. Це дозволяє використовувати їх як на сервері (API-рівень), так і на клієнті (для відображення результатів).

Основні функції модуля наведені в таблиці 3.4:

Таблиця 3.4 – Основні функції модуля

Функція	**Вхідні дані**	**Результат**	**Призначення**
computeResults()	масив варіантів із кількістю голосів	масив із відсотками та позначкою лідера	Основна агрегація для візуалізації
totalVotes()	список варіантів	загальна кількість голосів	Підрахунок сумарної активності
normalizeSelection()	вибрані ID + режим голосування	нормалізований масив ID	Обробка одиночного/множинного вибору
getStats()	дані системи	глобальні статистичні показники	Аналітика платформи

Такий підхід дозволяє централізувати всі обчислення та уникнути дублювання логіки в різних компонентах системи.

Основною функцією аналітичного модуля є computeResults(), яка виконує розрахунок відсоткового розподілу голосів та визначення лідируючого варіанту.

```
export function computeResults(options: PollOption[]): OptionResult[] {
  const total = options.reduce((acc, o) => acc + o.votes, 0)
  const maxVotes = options.reduce((m, o) => Math.max(m, o.votes), 0)

  return options.map((o) => ({
    id: o.id,
    label: o.label,
    votes: o.votes,
```

```
    percentage: total === 0 ? 0 : Math.round((o.votes / total) * 1000) / 10,
    isLeader: maxVotes > 0 && o.votes === maxVotes,
  }))
}
```

У межах цієї функції:

- обчислюється загальна кількість голосів;
- визначається максимальне значення для встановлення лідера;
- для кожного варіанта розраховується відсоткове співвідношення;
- передбачено захист від ділення на нуль у випадку відсутності голосів.

Таким чином формується повністю готова структура даних для відображення в інтерфейсі.

Отримані аналітичні результати подаються у вигляді графічних елементів, що забезпечують швидке сприйняття інформації користувачем.

У системі реалізовано два основні типи візуалізації:

Прогрес-бари – відображають частку голосів для кожного варіанту безпосередньо у списку відповідей. Реалізовано за допомогою компонентів shadcn/ui Progress.

Стовпчикові діаграми – використовуються для загального аналізу розподілу голосів. Побудовані на основі бібліотеки Recharts у компоненті ResultsChart.

Крім того, на головній сторінці застосовуються статистичні картки, які відображають глобальні показники системи (загальна кількість опитувань, голосів та учасників), отримані через функцію getStats().

Окрім візуалізації окремих опитувань, система формує узагальнені статистичні показники, які відображають загальний стан платформи. Функція getStats() агрегує дані про кількість опитувань, загальну активність користувачів та кількість голосів.

Ці дані використовуються для побудови інформаційних панелей (dashboard), що дозволяють оцінити рівень залученості користувачів та популярність опитувань.

Отже, реалізований аналітичний модуль забезпечує ефективну обробку результатів голосування та їх подальше представлення у зрозумілому графічному вигляді. Завдяки використанню чистих функцій, уніфікованої структури даних та сучасних бібліотек візуалізації, система підтримує гнучке масштабування та просте розширення аналітичних можливостей у майбутньому.

3.4 Забезпечення безпеки: шифрування даних, захист від повторного голосування

У межах навчального проєкту «ГолосХаб» реалізовано базовий, але структурно коректний рівень безпеки, який забезпечує цілісність даних, захист від типових атак та запобігає основним сценаріям зловживання. Оскільки система не використовує повноцінну реєстрацію користувачів, акцент зроблено на анонімній ідентифікації та серверній валідації.

У системі проаналізовано типові загрози для вебдодатків голосування та впроваджено відповідні контрзаходи (табл. 3.5).

Таблиця 3.5 – Основні загрози та механізми захисту

Загроза	Механізм захисту	Реалізація
SQL-ін’єкції	Параметризовані запити	Використання db.prepare() із плейсхолдерами ?
Повторне голосування	Анонімний voterId	Зберігання UUID у localStorage
Підробка даних запиту	Серверна валідація	Перевірка відповідності option_id до poll_id
Порушення цілісності БД	Транзакції та FK	BEGIN/COMMIT/ROLLBACK, зовнішні ключі
Некоректні запити	Валідація на клієнті/сервері	Перевірка обов’язкових полів і формату даних

Таблиця відображає основні типи загроз безпеці в системі голосування та відповідні механізми їх усунення. Для захисту від SQL-ін’єкцій застосовуються параметризовані запити, що виключають можливість виконання шкідливого SQL-коду. Повторне голосування обмежується за допомогою анонімного voterId, який зберігається в браузері користувача.

Підробка запитів запобігається серверною валідацією, яка перевіряє коректність зв’язку між опитуванням і варіантом відповіді. Цілісність бази даних забезпечується транзакціями та зовнішніми ключами, що гарантує атомарність операцій. Некоректні або неповні дані відсіюються завдяки валідації як на клієнтському, так і на серверному рівнях.

Однією з ключових загроз є можливість підміни SQL-запитів через користувацький ввід. Для запобігання цьому у системі використано параметризовані запити SQLite.

Замість конкатенації рядків застосовується метод db.prepare(), який відокремлює SQL-структуру від вхідних даних. Це повністю нейтралізує можливість SQL-ін’єкцій, оскільки значення передаються як параметри, а не як частина запиту. Реалізація наведена у додатку Б.1 – модуль роботи з базою даних.

Оскільки система не використовує реєстрацію користувачів, ідентифікація учасника здійснюється за допомогою анонімного voterId, який генерується один раз для кожного браузера.

```
export function getVoterId(): string {
  const key = "voteshub:voter-id"
  let id = window.localStorage.getItem(key)

  if (!id) {
    id = crypto.randomUUID()
    window.localStorage.setItem(key, id)
  }
```

```
  return id
}
```

Цей ідентифікатор:

- генерується один раз при першому вході;
- зберігається у localStorage;
- використовується для всіх подальших голосів.

На основі цього ID система може визначити, чи вже голосував користувач у конкретному опитуванні, і блокувати повторну участь на рівні API. Фрагмент клієнтської логіки наведений у додатку Б.2 – модуль роботи з користувачем.

Для забезпечення коректності збереження голосів використано комбінацію серверних перевірок та механізмів бази даних:

- перевірка належності варіанта відповіді – сервер підтверджує, що option_id належить відповідному poll_id;
- транзакції SQLite – усі операції запису виконуються в межах BEGIN/COMMIT/ROLLBACK, що гарантує атомарність;
- зовнішні ключі (FOREIGN KEY) – забезпечують логічну цілісність між таблицями polls, options і votes.

У випадку будь-якої помилки під час запису виконується відкат транзакції, що запобігає появі неконсистентних даних у базі. Реалізація серверної логіки наведена у додатку Б.3 – API голосування.

У навчальній версії системи безпосереднє шифрування даних на рівні застосунку не реалізується, оскільки воно забезпечується транспортним рівнем:

- усі запити виконуються через HTTPS;
- шифрування даних у транзиті забезпечується протоколом TLS;
- при розгортанні (наприклад, на Vercel) HTTPS активується автоматично.

Таким чином, дані голосування захищені під час передачі між клієнтом і сервером без необхідності додаткової криптографії на рівні коду.

Отже, реалізована модель безпеки забезпечує базовий рівень захисту системи «ГолосХаб» від основних загроз, характерних для вебзастосунків без авторизації. Поєднання параметризованих запитів, анонімної ідентифікації користувачів, серверної валідації та транзакційної роботи з базою даних дозволяє гарантувати цілісність і коректність даних. У перспективі система може бути розширена повноцінною аутентифікацією та додатковими криптографічними механізмами захисту.

3.5 Тестування системи: функціональне, навантажувальне, юзабіліті

Тестування розробленої системи «ГолосХаб» здійснювалося комплексно та охоплювало три ключові напрями: функціональне тестування, оцінку роботи системи під навантаженням, а також юзабіліті-тестування, спрямоване на перевірку зручності користувацької взаємодії. Такий підхід дозволив не лише перевірити коректність бізнес-логіки, але й оцінити практичну придатність інтерфейсу та стабільність роботи системи в умовах активного використання.

Функціональне тестування було реалізовано за допомогою бібліотеки Jest із застосуванням ts-jest для підтримки TypeScript. Основна увага приділялася перевірці модуля аналітики, який відповідає за обчислення результатів голосування. Зокрема, тестувалися функції computeResults, totalVotes та normalizeSelection, що реалізують ключову бізнес-логіку системи. Перевірялося коректне обчислення відсоткового співвідношення голосів, правильне визначення лідера, а також обробка граничних випадків, коли голоси відсутні. Окремо тестувалася логіка нормалізації вибору користувача, яка забезпечує коректну обробку як одиночного, так і множинного голосування. Результати запуску тестів показали повну успішність виконання всіх сценаріїв, що підтверджує стабільність реалізованої логіки.

Крім цього, проводилося інтеграційне тестування, яке передбачало перевірку взаємодії між API та базою даних SQLite. Воно здійснювалося вручну

через HTTP-запити (зокрема за допомогою curl), що дозволило переконатися у коректній роботі ендпоінтів створення опитувань, голосування та отримання результатів. Усі запити повертали очікувані дані, що свідчить про правильну інтеграцію серверної логіки з рівнем зберігання даних.

Окрему увагу приділено навантажувальному тестуванню, метою якого було оцінити поведінку системи при збільшенні кількості голосів. Завдяки використанню індексів у базі даних за полями poll_id та option_id, а також оптимізованим SQL-запитам із агрегацією через COUNT(*), система демонструє стабільну продуктивність навіть при значному зростанні обсягу даних. Додатково транзакційна модель запису голосів із використанням BEGIN/COMMIT/ROLLBACK забезпечує відсутність конфліктів при одночасних запитах, що є критично важливим для коректності результатів.

Юзабіліті-тестування було спрямоване на оцінку зручності використання інтерфейсу системи з позиції кінцевого користувача. У ході перевірки основних сценаріїв – створення опитування, участі в голосуванні та перегляду результатів – було встановлено, що інтерфейс є інтуїтивно зрозумілим та адаптивним. Користувач отримує миттєвий зворотний зв’язок у вигляді сповіщень, а результати голосування оновлюються автоматично, що створює ефект роботи системи в реальному часі. Адаптивний дизайн забезпечує коректне відображення як на десктопних, так і на мобільних пристроях, що додатково підвищує зручність використання.

Таким чином, проведене тестування підтвердило, що система «ГолосХаб» функціонує стабільно, коректно обробляє дані, витримує базове навантаження та забезпечує комфортну взаємодію користувача з інтерфейсом.

3.6 Інструкція користувача

Система «ГолосХаб» розроблена як інтуїтивно зрозумілий вебзастосунок, що не потребує складного навчання або додаткової підготовки користувача. Всі

основні функції доступні через простий інтерфейс, розділений на логічні сторінки: перегляд опитувань, створення нового опитування та участь у голосуванні з переглядом результатів.

Після відкриття головної сторінки користувач потрапляє до списку доступних опитувань. Тут відображаються картки з питаннями, коротким описом та базовою інформацією про кількість голосів.

Для перегляду детальної інформації або участі у голосуванні необхідно обрати відповідне опитування, натиснувши на його картку. Інтерфейс автоматично перенаправляє користувача на сторінку опитування, де доступні варіанти відповідей та поточні результати.

Участь у голосуванні здійснюється шляхом вибору одного або кількох варіантів відповіді, залежно від налаштувань конкретного опитування. Після вибору користувач натискає кнопку «Проголосувати», після чого його відповідь надсилається на сервер і одразу враховується в загальній статистиці. У випадку успішного голосування інтерфейс автоматично перемикається в режим перегляду результатів, де відображаються відсотковий розподіл голосів, прогрес-бари та діаграма, що дозволяє швидко оцінити поточну ситуацію.

Для створення нового опитування користувач переходить на сторінку «Створити опитування». У відповідній формі необхідно ввести текст питання, за потреби додати опис, обрати режим голосування (один або кілька варіантів відповідей), а також заповнити список варіантів.

Система дозволяє динамічно додавати або видаляти варіанти, забезпечуючи гнучкість при формуванні опитування. Після заповнення всіх обов’язкових полів користувач підтверджує створення, і опитування одразу стає доступним для голосування.

Окремо варто відзначити, що результати опитувань оновлюються в автоматичному режимі без необхідності перезавантаження сторінки. Це забезпечується механізмом періодичного оновлення даних, завдяки чому користувач бачить актуальну інформацію в реальному часі. Додатково система

використовує візуальні елементи, такі як діаграми та прогрес-бари, що робить сприйняття результатів більш наочним і зрозумілим.

Таким чином, користувачеві достатньо виконати кілька простих дій – обрати опитування, проголосувати або створити нове – щоб повноцінно взаємодіяти з системою «ГолосХаб». Інтерфейс мінімізує кількість складних кроків і забезпечує швидкий доступ до всіх основних функцій без необхідності додаткового навчання.

ВИСНОВКИ

У межах бакалаврської роботи було розроблено та спроєктовано вебсистему для онлайн-голосувань та опитувань із підтримкою аналітики результатів у реальному часі. Досягнута мета роботи полягає у створенні функціональної, зручної та масштабованої системи, яка забезпечує повний цикл роботи з опитуваннями: їх створення, участь користувачів у голосуванні, автоматичну обробку результатів та їх візуалізацію.

У теоретичній частині було розглянуто основні поняття систем електронного голосування, проаналізовано сучасні підходи до організації опитувань, а також визначено ключові вимоги до подібних вебсистем. Окрему увагу приділено питанням безпеки, достовірності та анонімності голосування, що є критично важливими для забезпечення коректності результатів. Також проведено огляд існуючих рішень, таких як Google Forms, Mentimeter, Kahoot та SurveyMonkey, що дозволило визначити їх сильні та слабкі сторони та сформувати вимоги до власної системи.

У розділі проєктування було розроблено архітектуру системи на основі клієнт-серверноїмоделі, визначено основні модулі (збір даних, аналітика та візуалізація результатів), а також обґрунтовано вибір технологічного стеку. Побудовано структуру бази даних, що включає сутності користувачів, опитувань, варіантів відповідей і голосів. Додатково створено UML-діаграми (Use Case, Class Diagram, Sequence Diagram), які відображають ключові сценарії роботи системи та взаємодію її компонентів.

У практичній частині реалізовано повноцінний вебзастосунок, що дозволяє створювати опитування, брати участь у голосуванні та переглядати результати в реальному часі. Інтерфейс користувача побудований з акцентом на простоту та адаптивність, що забезпечує зручну роботу як на десктопних, так і на мобільних пристроях. Реалізовано механізм збору та обробки голосів із використанням

server-side логіки та циклічного оновлення даних, що дозволяє досягти ефекту «реального часу».

Окремим важливим компонентом системи є модуль аналітики, який забезпечує обчислення статистичних показників, побудову графіків і діаграм, а також формування узагальнених результатів опитувань. Це дозволяє користувачам швидко інтерпретувати дані та отримувати наочну інформацію про розподіл голосів.

Також у роботі реалізовано базові механізми безпеки, зокрема захист від SQL-ін’єкцій, запобігання повторному голосуванню за допомогою анонімного ідентифікатора користувача, серверну валідацію даних та транзакційний запис у базу даних. Це забезпечує цілісність та коректність збереженої інформації навіть за умов одночасної взаємодії багатьох користувачів.

Проведене тестування підтвердило коректність реалізації основних функцій системи. Функціональні тести засвідчили правильність роботи бізнес-логіки, навантажувальне тестування показало стабільність системи при зростанні кількості голосів, а юзабіліті-оцінка підтвердила зручність та інтуїтивність інтерфейсу.

Таким чином, розроблена вебсистема повністю відповідає поставленим вимогам і забезпечує ефективний інструмент для створення, проведення та аналізу онлайн-опитувань у реальному часі. Отримані результати можуть бути використані як основа для подальшого розширення функціональності, зокрема впровадження повноцінної аутентифікації користувачів, масштабування системи та інтеграції більш складних аналітичних інструментів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sampigethaya K., Poovendran R. A framework and taxonomy for comparison of electronic voting schemes. *Computers & Security*. 2006. Vol. 25, no. 2. P. 137–153. DOI: https://doi.org/10.1016/j.cose.2005.11.003.

2. Jardí-Cedó R., Pujol-Ahulló J., Castellà-Roca J., Viejo A. Study on poll-site voting and verification systems. *Computers & Security*. 2012. Vol. 31, no. 8. P. 989–1010. DOI: https://doi.org/10.1016/j.cose.2012.08.001.

3. Regmi P. R., Waithaka E., Paudyal A., Simkhada P., Van Teijlingen E. Guide to the design and application of online questionnaire surveys. *Nepal Journal of Epidemiology.* 2017. Vol. 6, no. 4. P. 640–644. DOI: https://doi.org/10.3126/nje.v6i4.17258.

4. Ball H. L. Conducting online surveys. *Journal of Human Lactation*. 2019. Vol. 35, no. 3. P. 413–417. DOI: https://doi.org/10.1177/0890334419848734.

5. Анісімова Л. А. Протоколи електронного голосування. *Наукові записки НаУКМА. Комп'ютерні науки*. 2013. Т. 151. С. 88–96.

6. Heinl M. P., Gölz S., Bösch C. Remote Electronic Voting in Uncontrolled Environments: A classifying survey. *ACM Computing Surveys*. 2022. Vol. 55, no. 8. P. 1–44. DOI: https://doi.org/10.1145/3551386.

7. Ostapets D., Motylenko V. Analysis of approaches of electronic voting systems implementation. *System Technologies*. 2025. Vol. 6, no. 155. P. 50–60. DOI: https://doi.org/10.34185/1562-9945-6-155-2024-06.

8. Appel A. W. Is Internet Voting Trustworthy? The Science and the Policy Battles. *University of New Hampshire Law Review*. 2023. Vol. 21, no. 2. P. 523. URL: https://scholars.unh.edu/unh_lr/vol21/iss2/9/ (дата звернення: 02.04.2026).

9. International Institute for Democracy and Electoral Assistance. Впровадження електронного голосування: основні аспекти. Стокгольм : International IDEA, 2018. 43 с. ISBN 978-91-7671-166-8.

10. Google Forms. Офіційна документація Google Workspace. URL: https://workspace.google.com/products/forms/ (дата звернення: 02.04.2026).

11. Involve.me. What Is Google Forms & How to Use It (Full Beginner's Guide 2025). URL: https://www.involve.me/blog/what-is-google-forms (дата звернення: 02.04.2026).

12. Cloud Skills Academy. What's new in Forms? URL: https://www.cloudskillsacademy.com/forms/whats-new-in-forms (дата звернення: 02.04.2026).

13. Wooclap. Mentimeter Review: Features, Pricing & Alternative. URL: https://www.wooclap.com/en/blog/mentimeter-review/ (дата звернення: 02.04.2026).

14. Khan M. A. Mentimeter Tool for Enhancing Student Engagement and Active Learning: A Literature Review. 2025. URL: https://www.researchgate.net/publication/389708725 (дата звернення: 02.04.2026).

15. Kahoot! Official Website. URL: https://kahoot.com/ (дата звернення: 02.04.2026).

16. Wooclap. Kahoot Review: Features, Pricing & Alternative. URL: https://www.wooclap.com/en/blog/kahoot-review/ (дата звернення: 02.04.2026).

17. Özdemir A. Kahoot! Game-based digital learning platform: A comprehensive meta-analysis. Journal of Computer Assisted Learning. 2025. DOI: https://doi.org/10.1111/jcal.13084.

18. SurveyMonkey. Офіційна документація. URL: https://www.surveymonkey.com/ (дата звернення: 02.04.2026).

19. SuperbWebsiteBuilders. SurveyMonkey Review 2025 – Features, Pricing, Pros & Cons. URL: https://superbwebsitebuilders.com/surveymonkey-review/ (дата звернення: 02.04.2026).

20. TypeScript. URL: https://www.typescriptlang.org/ (дата звернення: 02.04.2026).

21. Next.js by Vercel – the React framework. URL: https://nextjs.org/ (дата звернення: 02.04.2026).

22. React. URL: https://react.dev/ (дата звернення: 02.04.2026).

23. Tailwind CSS. URL: https://tailwindcss.com/ (дата звернення: 02.04.2026).

24. shadcn/ui. URL: https://ui.shadcn.com/ (дата звернення: 02.04.2026).

25. SQLite. URL: https://sqlite.org/ (дата звернення: 02.04.2026).

26. SWR. URL: https://swr.vercel.app/ (дата звернення: 02.04.2026).

27. Recharts. URL: https://recharts.org/ (дата звернення: 02.04.2026).

28. Sonner. shadcn/ui. URL: https://ui.shadcn.com/docs/components/sonner (дата звернення: 02.04.2026).

ДОДАТОК А

ІНТЕРФЕЙС КОРИСТУВАЧА

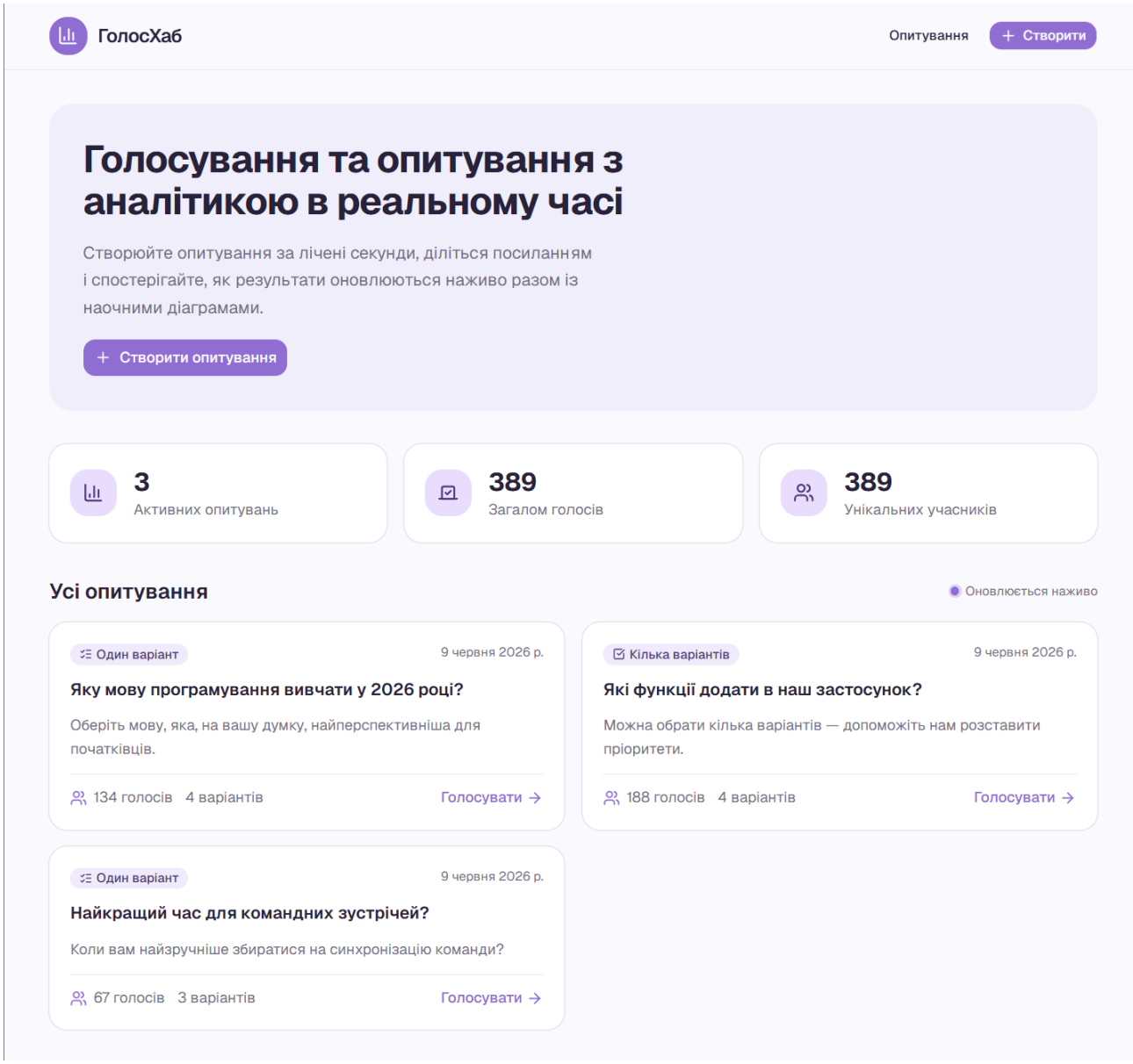


Рисунок А.1 – Головна сторінка

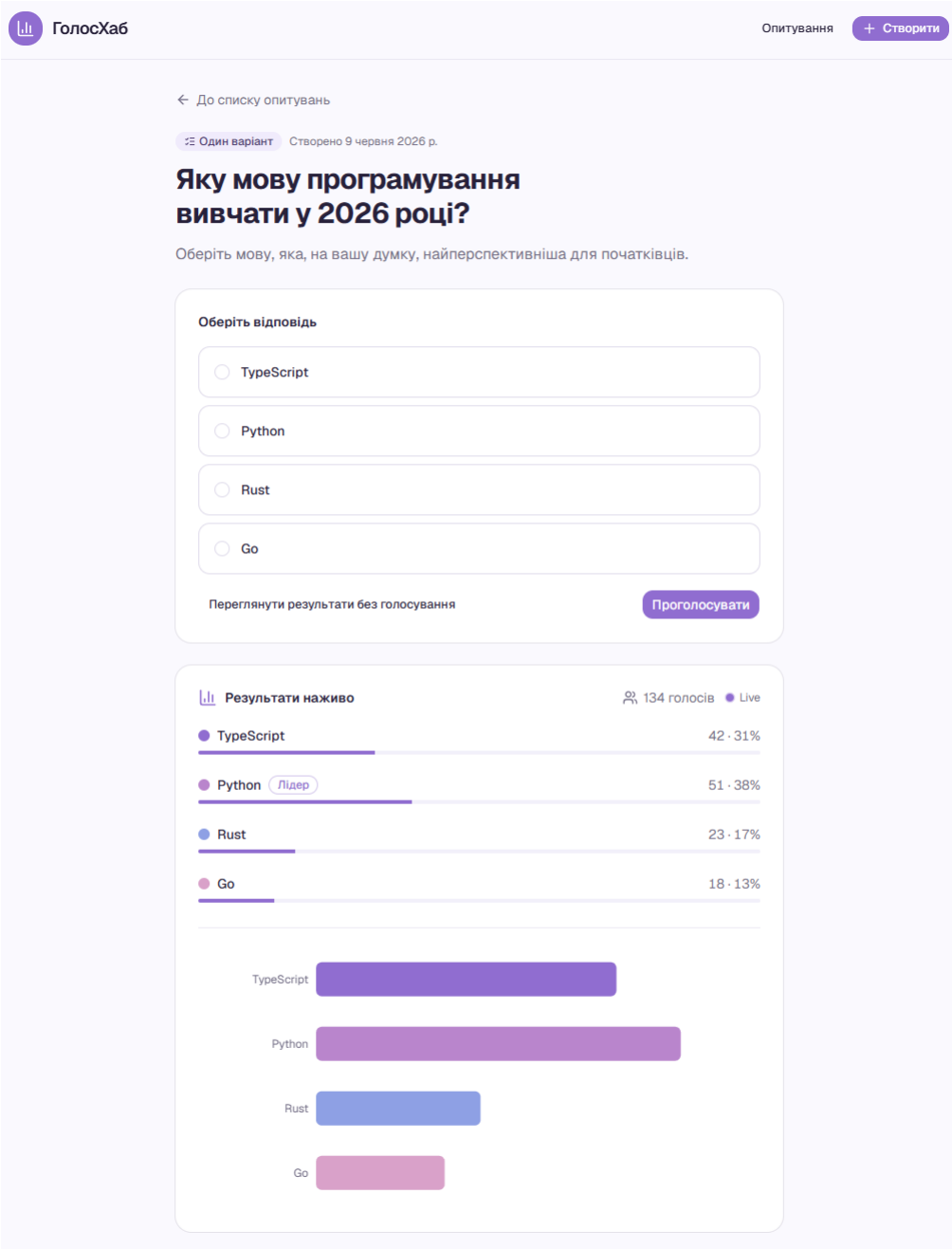


Рисунок А.2 – Сторінка опитування

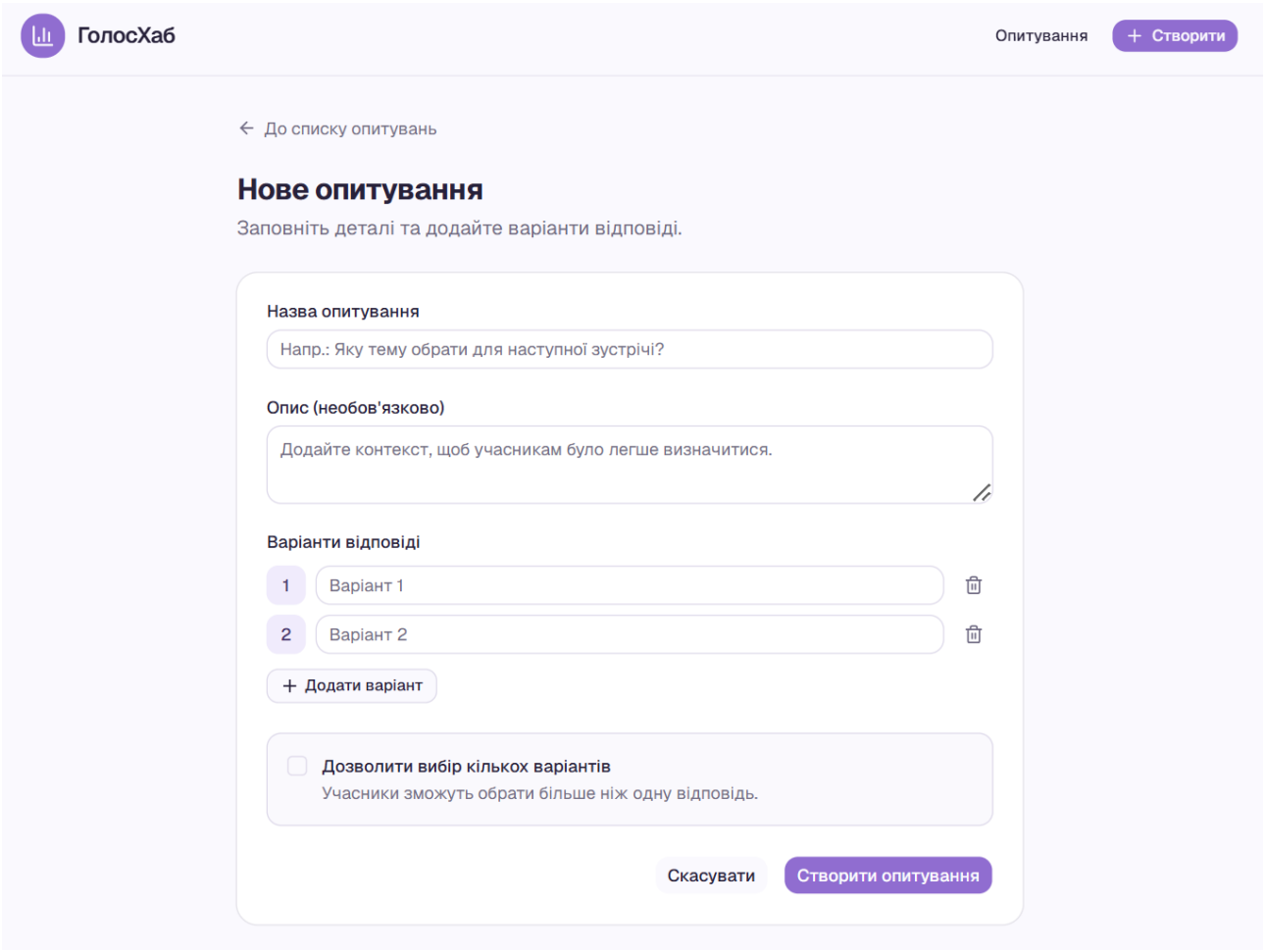


Рисунок А.3 – Сторінка створення опитування

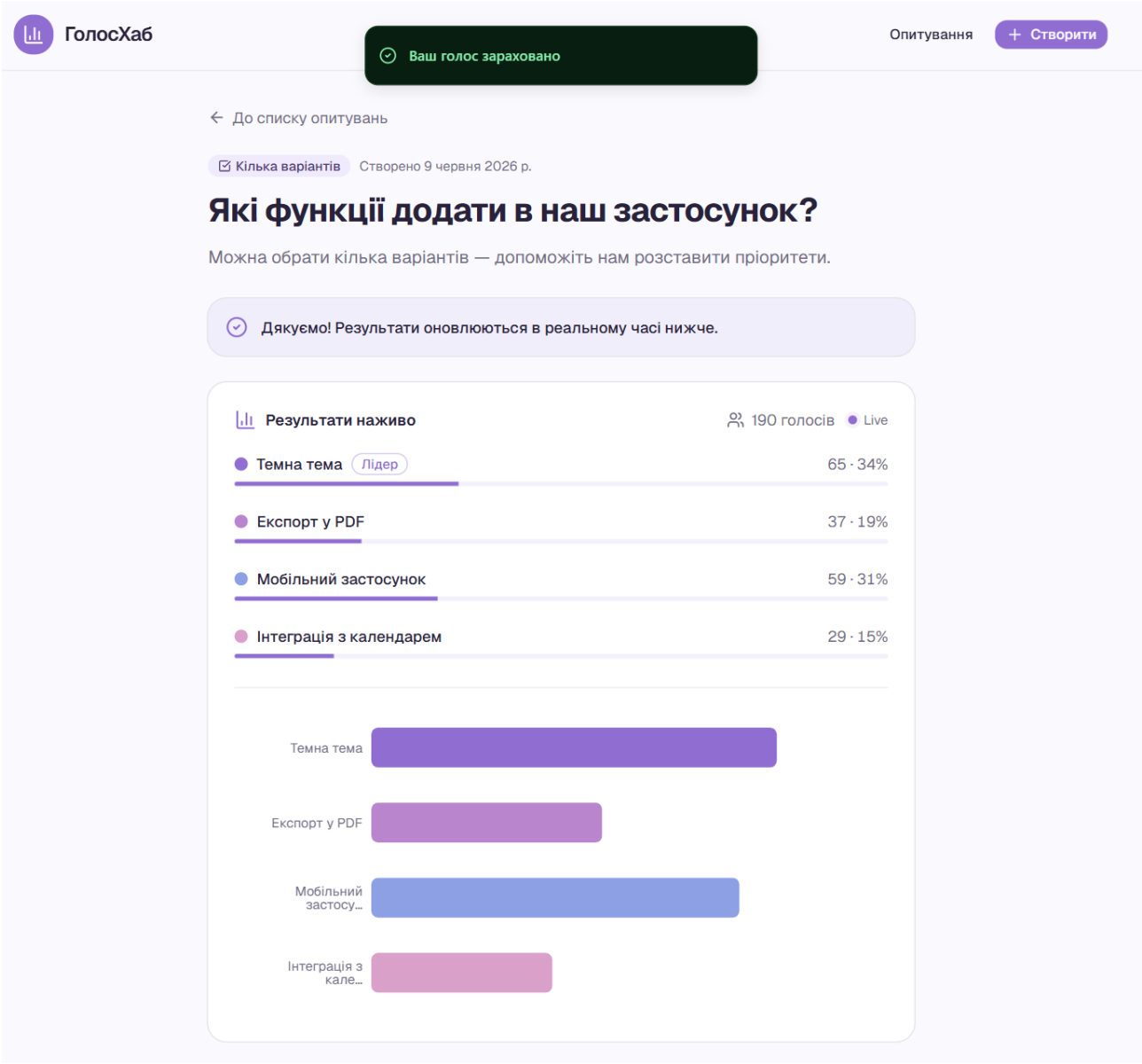


Рисунок А.4 – Сторінка після голосування

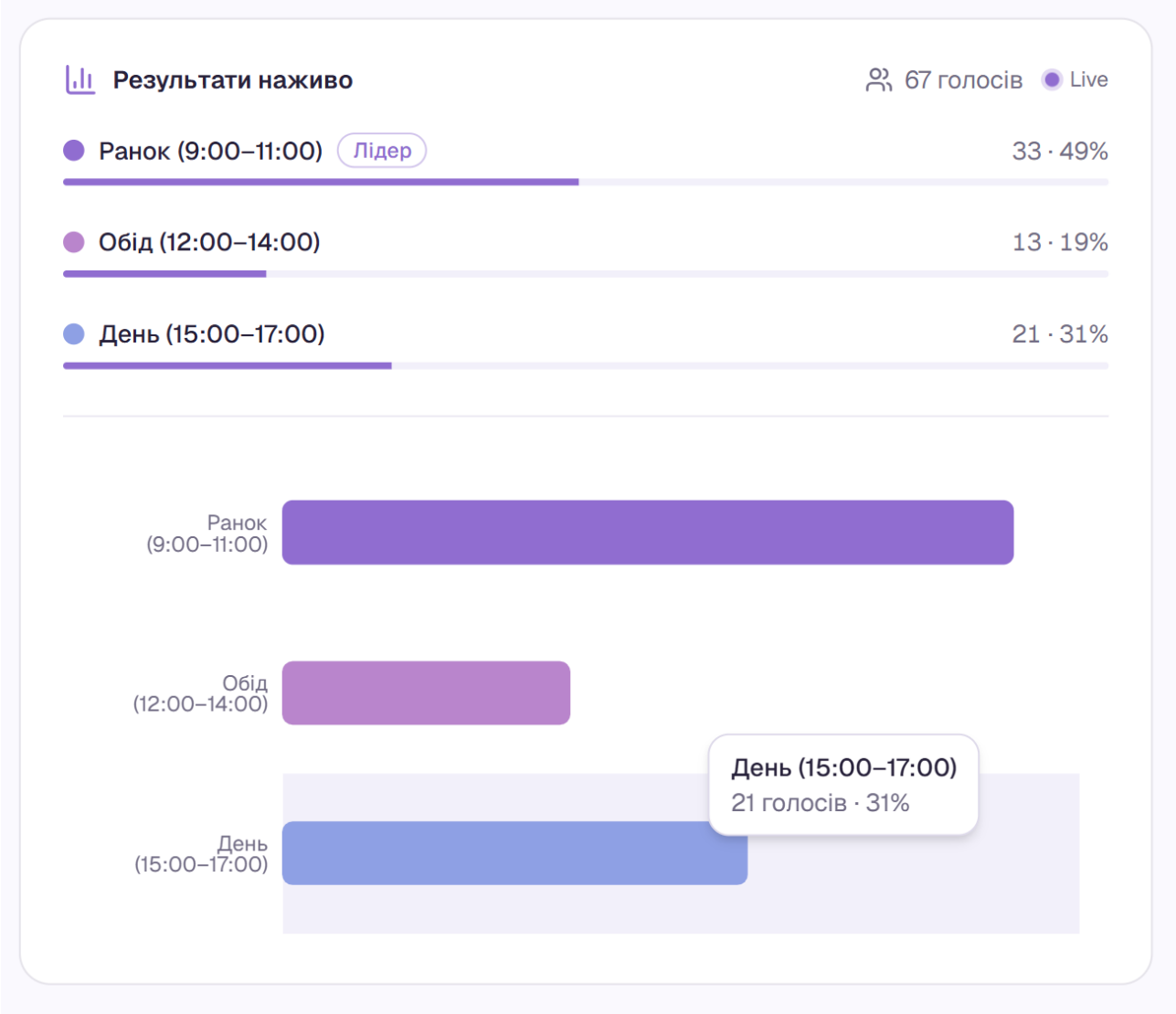


Рисунок А.5 – Відображення діаграм

ДОДАТОК Б

ЛІСТИНГ РЕАЛІЗАЦІЇ МЕХАНІЗМУ ЗБОРУ ТА ОБРОБКИ ГОЛОСІВ У РЕАЛЬНОМУ ЧАСІ

Лістинг Б.1 – Збір голосів

```
import { type NextRequest, NextResponse } from "next/server"
import { submitVote, getPoll } from "@/lib/queries"

export const dynamic = "force-dynamic"

export async function POST(
  req: NextRequest,
  { params }: { params: Promise<{ id: string }> },
) {
  try {
    const { id } = await params
    const body = await req.json()
    // Нормалізація вхідних даних: масив рядкових ID
    const optionIds = Array.isArray(body.optionIds)
      ? body.optionIds.map((o: unknown) => String(o))
      : []
    // Якщо ідентифікатор учасника відсутній – генеруємо анонімний
    const voter = String(body.voter ?? "").trim() || crypto.randomUUID()

    if (optionIds.length === 0) {
      return NextResponse.json(
```

```
        { error: "Оберіть хоча б один варіант" },
        { status: 400 },
      )
    }

    submitVote({ pollId: id, optionIds, voter })

    // Одразу повертаємо оновлений стан опитування
    const poll = getPoll(id)
    return NextResponse.json({ poll })
  } catch (err) {
    return NextResponse.json(
      { error: err instanceof Error ? err.message : "Помилка сервера" },
      { status: 500 },
    )
  }
}
```

Лістинг Б.2 – Бізнес-логіка запису голосу

```
export function submitVote(input: {
  pollId: string
  optionIds: string[]
  voter: string
}): void {
  const db = getDb()
  const poll = db
    .prepare("SELECT multiple FROM polls WHERE id = ?")
    .get(input.pollId) as { multiple: number } | undefined
```

```
if (!poll) throw new Error("Опитування не знайдено")

// Для одиночного вибору беремо лише перший варіант, навіть якщо клієнт надіслав більше
const optionIds =
  poll.multiple === 1 ? input.optionIds : input.optionIds.slice(0, 1)

if (optionIds.length === 0) throw new Error("Не обрано жодного варіанта")

// Підготовлені (параметризовані) запити – захист від SQL-ін’єкцій
const valid = db.prepare(
  "SELECT COUNT(*) AS c FROM options WHERE poll_id = ? AND id = ?",
)
const insertVote = db.prepare(
  "INSERT INTO votes (id, poll_id, option_id, voter) VALUES (?, ?, ?, ?)",
)

db.exec("BEGIN")                    // початок транзакції
try {
  for (const optId of optionIds) {
    const ok = valid.get(input.pollId, optId) as { c: number }
    if (ok.c === 0) throw new Error("Невірний варіант відповіді")
    insertVote.run(crypto.randomUUID(), input.pollId, optId, input.voter)
  }
  db.exec("COMMIT")                 // фіксація всіх голосів разом
} catch (err) {
  db.exec("ROLLBACK")               // відкат при будь-якій помилці
  throw err
```

```
  }
}
```

Лістинг Б.3 – Читання результатів

```
const options = db
  .prepare(
    `SELECT
      o.id, o.label, o.position,
      (SELECT COUNT(*) FROM votes v WHERE v.option_id = o.id) AS votes
    FROM options o
    WHERE o.poll_id = ?
    ORDER BY o.position ASC`,
  )
  .all(id) as PollOption[]

const totalVotes = options.reduce((acc, o) => acc + o.votes, 0)
```

Лістинг Б.4 – Оновлення в реальному часі

```
export function PollView({ pollId }: { pollId: string }) {
  // SWR автоматично опитує сервер кожні 3 секунди (short polling)
  const { data, error, isLoading } = useSWR<{ poll: PollDetail }>(
    `/api/polls/${pollId}`,
    fetcher,
    { refreshInterval: 3000 },
  )
```

```
const [selected, setSelected] = useState<string[]>([])
const [submitting, setSubmitting] = useState(false)
const [voted, setVoted] = useState(false)

// Надсилання голосу на сервер
async function handleVote() {
  if (selected.length === 0) {
    toast.error("Оберіть варіант відповіді")
    return
  }
  setSubmitting(true)
  try {
    const res = await fetch(`/api/polls/${pollId}/vote`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ optionIds: selected, voter: getVoterId() }),
    })
    const result = await res.json()
    if (!res.ok) throw new Error(result.error || "Помилка голосування")

    markVoted(pollId)        // позначаємо опитування як проголосоване
    setVoted(true)           // перемикаємо UI на режим результатів
    toast.success("Ваш голос зараховано")
  } catch (err) {
    toast.error(err instanceof Error ? err.message : "Помилка")
  } finally {
    setSubmitting(false)
  }
}
// ... рендеринг форми та результатів
```

```
}

// Універсальний завантажувач для SWR
export const fetcher = async (url: string) => {
  const res = await fetch(url)
  if (!res.ok) {
    const data = await res.json().catch(() => ({}))
    throw new Error(data.error || "Помилка завантаження")
  }
  return res.json()
}

// Стабільний анонімний ідентифікатор учасника (зберігається в localStorage)
export function getVoterId(): string {
  if (typeof window === "undefined") return ""
  const key = "voteshub:voter-id"
  let id = window.localStorage.getItem(key)
  if (!id) {
    id = crypto.randomUUID()
    window.localStorage.setItem(key, id)
  }
  return id
}
```