Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**Інформаційна технологія організації колективних опитувань**

Виконав: студенти 2-го курсу, групи
3КН-22м спеціальності 122
«Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Озменчук І. С.
(прізвище та ініціали)

Керівник: к.т.н., доц. кафедри КН
Колодний В.В.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., доц. кафедри САІТ
Крижановський Є.М.
(прізвище та ініціали)
«07» 12 2023 р.

**Допущено до захисту**
Завідувач кафедри КН
д.т.н., проф. Яровий А. А.
(прізвище та ініціали)
«08» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри КН**
**Д.т.н., проф. Яровий А.А.**
29 08 **2023 року**

# З А В Д А Н Н Я
# НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Озменчуку Іллі Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія організації колективних опитувань
керівник роботи д.т.н., професор кафедри КН Яровий А.А.
затверджені наказом вищого навчального закладу від "18" 09 2023 року №247
2. Строк подання студентом роботи 13.11. 2023 року

3. Вихідні дані до роботи:
Вхідні дані: Мова програмування повинна бути об'єктно-орієнтованою та підтримувати роботу з даними і управління базами даних, мінімальна кількість відповіді – 2, спосіб введення відповіді – відмітка, мінімальна кількість запитань – 5.

4. Зміст текстової частини:
Вступ, обґрунтування доцільності розробки інформаційної технології організації колективних опитувань, розробка інформаційної технології організації колективних опитувань, програмна реалізація інформаційної технології організації колективних опитувань, економічна частина, висновки, перелік використаних джерел, додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Фрагмент основних кодових елементів, модуля та правил, візуалізація обраних блок схем алгоритмів. Графічне представлення модулів. Алгоритм роботи системи. UML-діаграма роботи програмних модулів.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціалита посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | виконприй |
| 1-3 | Колодний В.В., к.т.н., доц. каф. КН | 29.08.23 | 21.11.23 |
| 4 | Кавецький В.В., к.е.н., доц. каф. ЕПВМ | 8.10.23 | 21.11.23 |
| | | | |

7. Дата видачі завдання 29.08. 2023 року

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів магістерської кваліфікаційної роботи | Строк виконання етапів роботи | Пмі |
|---|---|---|---|
| 1 | Обґрунтування доцільності розробки інформаційної технології організації колективних опитувань | 01.09.23 – 05.09.23 | |
| 2 | Розробка інформаційної технології організації колективних опитувань | 06.09.23 – 16.09.23 | |
| 3 | Програмна реалізація інформаційної технології організації колективних опитувань | 17.09.23 – 07.10.23 | |
| 4 | Підготовка економічної частини | 08.10.23 – 23.10.23 | |
| 5 | Апробація та/або впровадження результатів дослідження | 24.10.23 – 01.11.23 | |
| 6 | Оформлення пояснювальної записки, графічного матеріалу та презентації | 02.11.23 – 10.11.23. | |
| | | | |

Студент _____ (підпис) Озменчук І.С.

Керівник роботи _____ (підпис) Колодний В.В.

# АНОТАЦІЯ

УДК 004.89

Озменчук І.С. Інформаційна технологія організації колективних опитувань. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп`ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2023. 152 с.

На укр. мові. Бібліогр.: 20 назв; рис.: 31; табл. 14.

Актуальність даного дослідження обумовлена наростаючим інтересом організацій, дослідників та користувачів до збору інформації в цифрову епоху шляхом проведення опитувань. У зв'язку з цим, розробка ефективної системи організації колективних опитувань стає ключовим інструментом для зручного та ефективного збору даних від різних аудиторій. Порівняно з традиційними методами, такими як паперові анкети або телефонні опитування, ця система автоматизує процес збору даних, ефективно зменшуючи час і зусилля, необхідні для проведення опитувань.

Дослідження спрямоване на розробку та вдосконалення інформаційної технології для організації колективних опитувань. Застосовуються передові технології, такі як Nginx, Node.js, Express, OpenSMTPD та MongoDB для серверної частини, і Python та Vue.js для клієнтської частини. Це дозволяє створити систему, яка буде доступною та адаптованою до різних потреб організацій та користувачів. Результатом дослідження є інформаційна технологія організації колективних опитувань, що відрізняється вдосконаленою інформаційною моделлю, а отримані результати можуть бути використані для подальшого розвитку і вдосконалення інструментів збору та обробки даних.

Ключові слова: інформаційні технології, колективні опитування, онлайн-платформи, мобільні додатки, соціальні мережі, збільшення обсягу вибірки, конфіденційність, аналіз даних в реальному часі.

# ABSTRACT

Ozmenchuk I.S. Information technology for organizing collective surveys. Master's qualification work in specialty 122 "Computer Science", educational program "Artificial Intelligence Systems". Vinnytsia: VNTU, 2023. 152 c.

In Ukrainian. Bibliogr.: 20 titles; Figures: 31; Table 14.

The relevance of this study is due to the growing interest of organizations, researchers and users in collecting information in the digital age through surveys. In this regard, the development of an effective system for organizing collective surveys is becoming a key tool for convenient and efficient data collection from different audiences. Compared to traditional methods, such as paper questionnaires or telephone surveys, this system automates the data collection process, effectively reducing the time and effort required to conduct surveys.

The research is aimed at developing and improving information technology for organizing collective surveys. Advanced technologies are used, such as Nginx, Node.js, Express, OpenSMTPD and MongoDB for the server side, and Python and Vue.js for the client side. This allows us to create a system that will be accessible and adaptable to the different needs of organizations and users. The result of the study is an information technology for organizing collective surveys, which is characterized by an improved information model, and the results can be used for further development and improvement of data collection and processing tools.



Keywords: information technology, collective surveys, online platforms, mobile applications, social networks, increasing sample size, privacy, real-time data analysis.

# ЗМІСТ

# ВСТУП

**Актуальність теми дослідження.** В сучасному цифровому віці все більше організацій, дослідників та індивідуальних користувачів прагнуть отримати інформацію шляхом проведення опитувань. Інформаційна технологія організації колективних опитувань надає зручний та ефективний спосіб збирати дані з різних аудиторій. Традиційні методи проведення опитувань, такі як паперові анкети або телефонні опитування, можуть бути неефективною задачею. Інформаційна технологія організації колективних опитувань дозволяє автоматизувати процес збору даних, зменшуючи час та зусилля, необхідні для проведення опитування. Інформаційна технологія організації колективних опитувань може бути налаштована та адаптована до потреб різних організацій та користувачів. Вона може включати різноманітні функції, такі як варіанти відповідей, логіка галуження та аналітика результатів, що дозволяє створювати опитування, відповідні конкретним вимогам та цілям. Завдяки інтернету та сучасним технологіям, система організації колективних опитувань може бути доступною для широкого кола користувачів. Основою серверної частини архітектурного рішення для розробленої системи стануть такі технології як Nginx (власне веб-сервер), Node.js та Express (для розробки API), OpenSMTPD (поштовий сервер для авторизації), MongoDB(no-sql база даних для зберігання результатів опитувань). Клієнтська частина, або ж front-end, буде реалізована у вигляді веб-сторінки за допомогою мови Python та фреймворку Vue.js. Таким чином дана технологія спростить процес збору та обробки даних, буде зручною у використанні, буде гучною, адаптивною та масштабованою.

**Зв’язок роботи з науковими програмами, планами, темами.** Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних

технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

**Мета та завдання дослідження.** Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей процесу організації та проведення колективних опитувань.

Для досягнення поставленої мети необхідно виконати такі завдання:

– аналіз предметної області та програм аналогів створення опитувань;

– розробка структури інформаційної технології організації колективних опитувань;

– розробка бази даних та архітектури серверної частини;

– розробка програмної реалізації системи колективних опитувань;

– провести тестування програмного додатку проведення та організації колективних опитувань;

– виконати економічні розрахунки.

**Предметом дослідження** є програмні засоби проведення та організації колективних опитувань, враховуючи різні особливості опитувань.

**Об’єктом дослідження** є процес проведення та організації колективних опитувань з використанням інформаційної технології.

**Методи дослідження.** У роботі використано такі методи наукових досліджень: методи статистичного аналізу; методи прийняття рішень; методи об’єктно-орієнтованого програмування.

**Наукова новизна.** Запропоновано інформаційну технологію організації колективних опитувань, яка відрізняється від існуючих удосконаленою інформаційною моделлю, що дозволило розширити функціональні можливості процесу організації та проведення колективних опитувань.

**Практична значення одержаних результатів** полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення зокрема:

1. Розроблено алгоритм для інформаційної технології організації колективних опитувань;

2. Розроблено програмне забезпечення для інформаційної технології організації колективних опитувань, що розширює функціональні можливості процесу організації та проведення колективних опитувань.

**Достовірність теоретичних положень** магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю роботи програмних модулів, тестуванням програмної реалізації інформаційної технології організації колективних опитувань. Адекватність розробленої технології підтверджується результатами тестування та експериментальних досліджень.

**Особистий внесок магістранта.** Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У роботах, опублікованих у співавторстві, автору належать такі результати: [1] – розробка програмного застосунку для інформаційної технології організації колективних опитувань.

**Апробація.** Результати дослідження були представлені та обговорені на науково-технічній конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2024)" [1].

**Публікації.** За результатами магістерської кваліфікаційної роботи опубліковано дослідження опубліковано: 1 тези доповідей конференцій [1].

# 1 ОБРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ОПИТУВАНЬ

## 1.1 Аналіз предметної області

Існує багато різних систем організації опитувань, які варіюються за функціональністю, масштабом та способом реалізації. Для зручності всі системи організації опитувань було винесено в чотири окремих види[2]:

1. Веб-платформи для опитувань. Це онлайн-сервіси, які дозволяють створювати та проводити опитування через веб-інтерфейс. Вони забезпечують широкий функціонал, включаючи розташування питань, варіанти відповідей, аналітику результатів та можливості експорту даних;
2. Самостійні програми для опитувань. Це програми, які встановлюються на локальний комп'ютер або сервер та надають можливість проводити опитування. Вони мають більш широкі можливості налаштування, включаючи дизайн опитувань, управління користувачами та розширені засоби аналізу даних;
3. Мобільні додатки для опитувань. Це додатки, які можна встановити на мобільний пристрій та використовувати для проведення опитувань;
4. Внутрішня система опитувань. Це системи, які розробляються або впроваджуються безпосередньо в організації з метою проведення внутрішніх опитувань серед співробітників або клієнтів.

Будь-який програмний продукт повинен мати інтуїтивний і простий інтерфейс, що дозволяє користувачам швидко створювати, налаштовувати та розповсюджувати опитування без необхідності спеціальних навичок або навчання. Щодо систем організації опитувань, продукт повинен надавати зручні інструменти для аналізу та візуалізації результатів опитувань. Він повинен допомагати користувачам зрозуміти дані, виділяти ключові відомості та приймати обґрунтовані рішення на основі результатів. Також програмний засіб повинен підтримувати різноманітні типи питань, логіку питань, вигляд та

оформлення опитувань. Він повинен дозволяти налаштовувати опитування відповідно до конкретних потреб користувача[3].

У випадку якщо система організації опитувань буде реалізована у вигляді веб-застосунку, сервіс повинен забезпечувати високий рівень безпеки для збереження та передачі даних опитувань. Він повинен гарантувати конфіденційність відповідей і захищати приватну інформацію користувачів.

Ідеальний сервіс повинен бути здатним працювати з великим обсягом опитувань і відповідей, забезпечуючи стабільну роботу та швидкий доступ до результатів. Він повинен мати масштабну інфраструктуру, щоб забезпечити високу продуктивність.

Інтеграція з іншими інструментами. Сервіс повинен мати можливість інтеграції з іншими інструментами, такими як CRM-системи, електронна пошта, соціальні мережі тощо. Це дозволить легко обробляти та використовувати отримані дані в інших аспектах діяльності організації.

Розробка програми зазвичай включає кілька етапів, які можуть варіюватись залежно від методології розробки і конкретного проекту. Основні етапи розробки програми включають такі[4]:

– Аналіз вимог. На цьому етапі встановлюються та аналізуються вимоги до програмного продукту. Визначаються функціональні та нефункціональні вимоги, проводиться дослідження предметної області та взаємодії з користувачами;

– Проектування. На цьому етапі створюється архітектура програмного продукту, визначаються компоненти, модулі та їх взаємозв'язки. Проектування включає розробку структури даних, вибір технологій, дизайн інтерфейсу користувача та інші аспекти системи;

– Реалізація. На цьому етапі програма розробляється за допомогою вибраних технологій та програмних мов. Розробники пишуть код, реалізовують функціональність, тестують та виправляють помилки;

– Тестування. На цьому етапі проводяться різноманітні види тестування для перевірки якості програмного продукту. Це включає модульне

тестування окремих компонентів, функціональне тестування, інтеграційне тестування та системне тестування для перевірки працездатності та відповідності вимогам;

– Випуск та розгортання. Після успішного завершення тестування програмний продукт готується до випуску. Це включає підготовку документації, пакування програми та розгортання на призначеному середовищі;

– Супровід та підтримка. Після випуску програмного продукту проводиться супровід та підтримка. Це включає виявлення та виправлення помилок, вдосконалення функціональності, підтримку користувачів та можливість розширення функціоналу програми.

Ці етапи можуть виконуватись послідовно або паралельно залежно від обраної методології розробки, наприклад, водоспадної (waterfall), ітеративно-інкрементної (agile) чи гібридної. Кожен етап відіграє важливу роль у створенні якісної та функціональної програми.

## 1.2 Характеристика та аналіз сервісів аналогів

Так як в рамках даної магістерської роботи розроблятиметься веб-застосунок для організації опитувань, то і системи-аналоги будуть проаналізовані відповідного формату[5].

Для початку було розглянуто, напевно найвідомішу і найпоширеніший в широких масах сервіс Google Forms. Це безкоштовний сервіс від Google, який дозволяє створювати опитування з різними типами питань, налаштовувати логіку питань, отримувати відповіді та аналізувати результати. Google Forms є простим у використанні та має зручний інтерфейс (рис. 1.1).

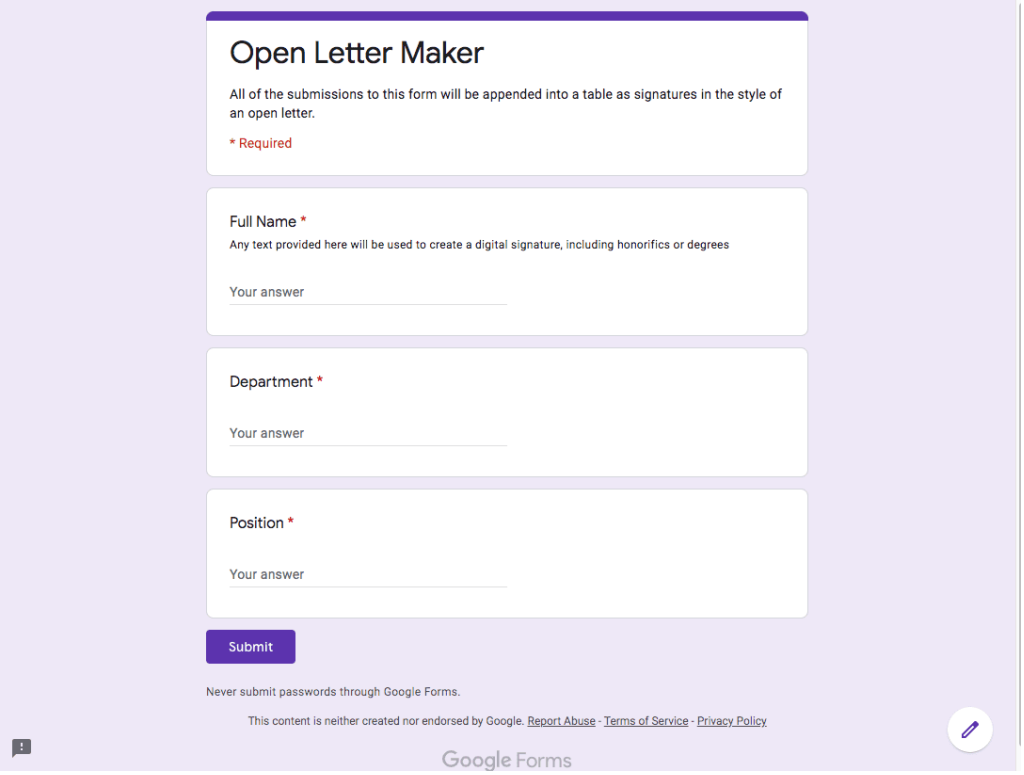


Рисунок 1.1 – Вигляд опитування в Google Forms

Google Forms надає зручний конструктор форм, де користувачі можуть легко створювати форми, додаючи різні типи питань, такі як вибір із списку, прапорці, випадаюче меню, коротка відповідь та багато інших. Користувачі також можуть додавати заголовки, описи, зображення та відео, щоб надати додатковий контекст. Користувачі можуть налаштовувати вигляд своїх форм, вибираючи теми оформлення, фонові зображення та кольори, що дозволяє створювати унікальні та привабливий дизайн. Хоча готових шаблонів практично немає, що є вже мінусом(рис. 1.2).

Після створення форми користувачі можуть легко ділитися нею, надсилати посилання на форму електронною поштою, публікувати на веб-сторінках або соціальних мережах. Користувачі можуть також обмежувати доступ до форми і контролювати, хто має право заповнювати форму.

Google Forms автоматично збирає відповіді в реальному часі і надає користувачам доступ до аналітики та статистики результатів. Користувачі

можуть переглядати відповіді в форматі таблиці, графіка або діаграми, а також експортувати дані у форматі Google Sheets для подальшого аналізу.

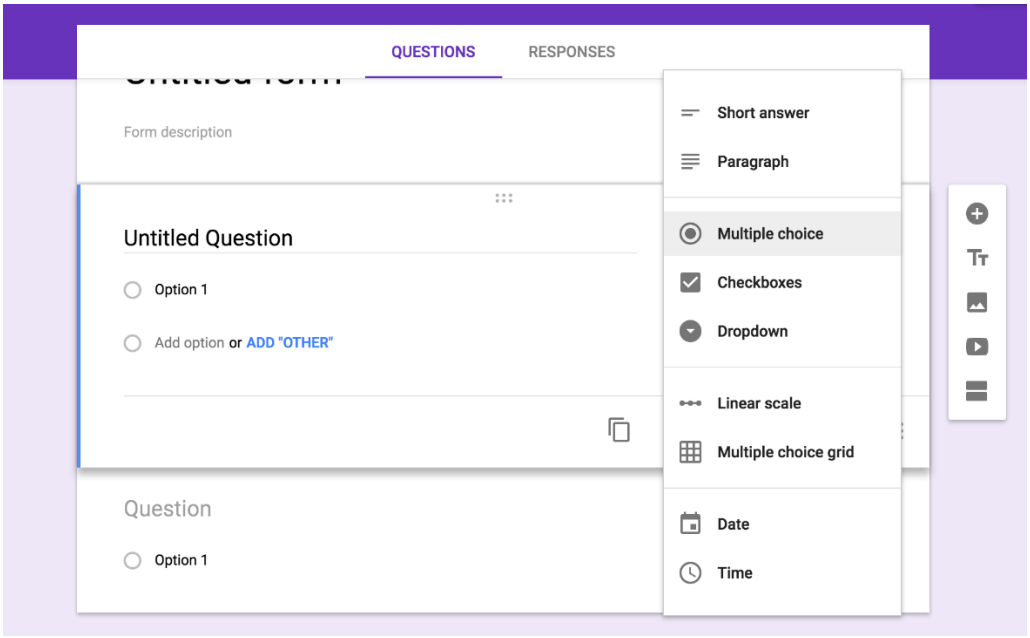


Рисунок 1.2 – Вигляд редактору опитувань Google Forms

Основними недоліками Google Forms є:

1. Обмежена налаштованість вигляду. Хоча Google Forms надає певну кількість налаштувань для вигляду форми, він має обмежену можливість настроювання дизайну та стилізації;

2. Відсутність додаткових функцій. Google Forms не має ряду додаткових функцій, які можуть бути корисними для певних сценаріїв. Наприклад, він може не мати вбудованих інструментів для проведення опитувань з оцінювання, розкритого тексту або діаграми Ганта;

3. Обмежена аналітика. Хоча Google Forms надає базову аналітику відповідей, вона може бути обмеженою для певних потреб. Іноді може виникнути необхідність в більш складних аналітичних функціях або інтеграції з іншими інструментами для детальнішого аналізу результатів;

4. Відсутність розширених контролів безпеки. Google Forms має базові функції безпеки, такі як обмеження доступу до форми, але він може бути обмежений для вимог організацій з високими стандартами безпеки даних.

Далі розглядається веб-сервіс Typeform, який є інтерактивним сервісом для створення форм, опитувань, анкет і збору даних. Typeform пропонує елегантні та сучасні шаблони форм з інтуїтивно зрозумілим інтерфейсом. Він надає можливість додавати медіа-елементи, зображення, відео та анімацію, що створює більш залучений та привабливий досвід для користувачів[6].

Typeform дозволяє налаштовувати форми під свої потреби. Користувачі можуть створювати змінні питання, галереї, умовні логіки та пропуски, що дозволяє збирати більш деталізовані дані та персоналізувати процес заповнення форми(рис. 1.3).

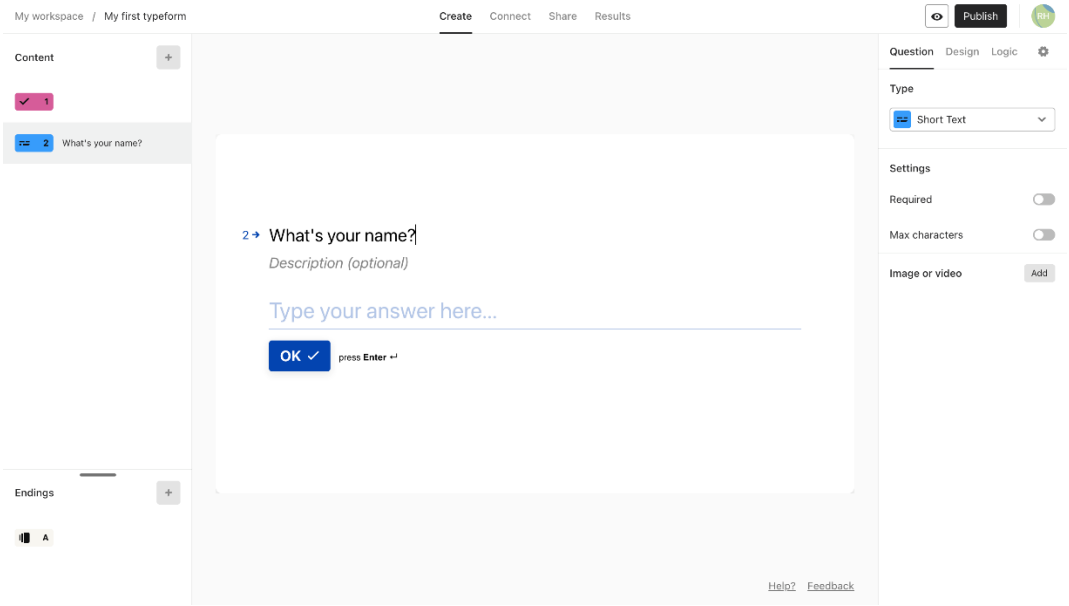


Рисунок 1.3 – Вигляд редактору опитувань Typeform

Typeform надає детальну аналітику відповідей, включаючи графіки, діаграми та статистику, яка допомагає зрозуміти дані та здійснювати аналіз результатів. Користувачі можуть також імпортувати дані в зовнішні інструменти аналізу даних для подальшого вивчення.

Також Typeform має ряд інтеграцій з популярними інструментами, такими як Google Sheets, Slack, MailChimp, Zapier та іншими, що дозволяє автоматизувати процеси та обробку зібраних даних.

А ще Typeform пропонує можливість створювати анкети з елементами гри, такими як вікторини, загадки або виклики, що робить процес заповнення більш захопливим та цікавим для учасників (рис. 1.4).

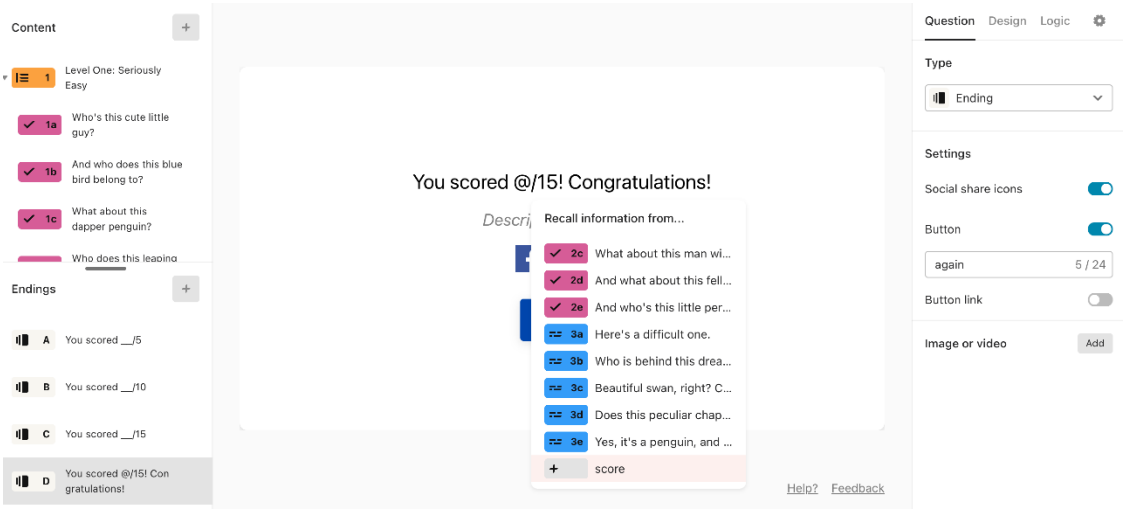


Рисунок 1.4 – Приклад створення вікторини у Typeform

Незважаючи на багато переваг, сервіс Typeform також має деякі недоліки:

1. Обмежена безкоштовна версія. Безкоштовний план Typeform має свої обмеження, такі як обмежену кількість відповідей і обмеження на функціональні можливості;

2. Відсутність повного контролю над даними. Збираючи дані за допомогою Typeform, ви залежите від сервісу для збереження та керування даними. Це може викликати певні обмеження щодо безпеки та конфіденційності даних;

3. Обмежена налаштованість дизайну. Typeform надає певну налаштованість дизайну, але ці можливості можуть бути обмеженими для користувачів, які шукають велику гнучкість у стилізації форми;

4. Відсутність розширених функцій. У порівнянні з іншими сервісами для опитувань, Typeform може відсутні деякі розширені функції або інструменти, які можуть бути корисними для певних сценаріїв опитувань. Наприклад, він може не мати вбудованих інструментів для складних умовних логік або опитувань з більшою кількістю рівнів.

Наступним розглянуто Qualtrics. Це платформа для проведення опитувань та досліджень, яка пропонує розширені можливості для створення складних опитувань, налаштування логіки питань, аналізу результатів та збору даних. Вона широко використовується у наукових та бізнес-дослідженнях[6].

Qualtrics надає велику свободу для створення та налаштування опитувальних форм, дозволяє налаштовувати умовні логіки, що дає можливість створювати відповіді та питання, що залежать від попередніх відповідей (рис. 1.5).

Qualtrics надає потужні аналітичні інструменти, що дозволяють аналізувати та інтерпретувати зібрані дані. Є можливість створювати графіки, діаграми та звіти, а також проводити статистичний аналіз для зрозуміння результатів опитувань.

Сервіс Qualtrics має можливості інтеграції з різними зовнішніми інструментами і сервісами, такими як CRM-системи, соціальні мережі, аналітичні платформи тощо.

Також Qualtrics забезпечує високий рівень захисту даних, включаючи шифрування та інші заходи безпеки. Ваші дані залишаються конфіденційними та захищеними.

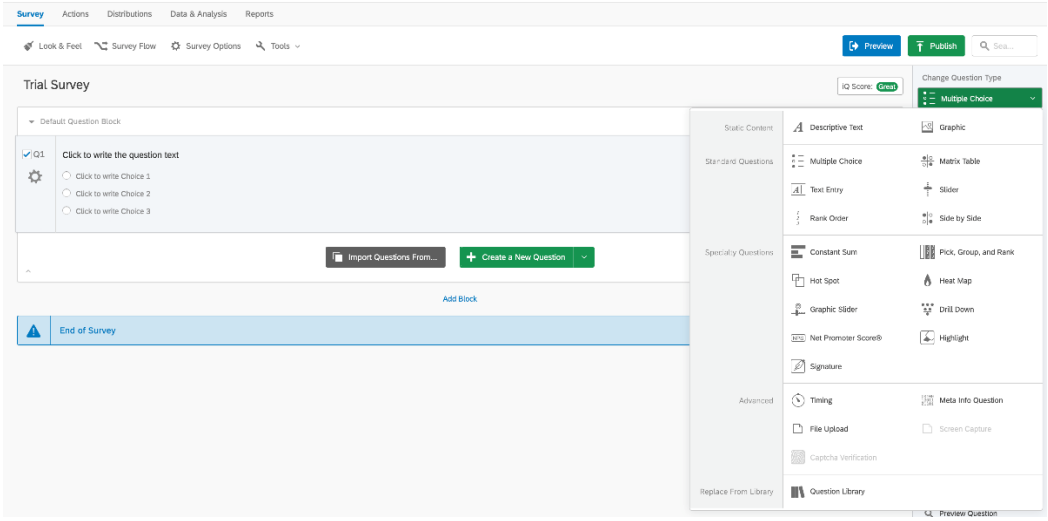


Рисунок 1.5 – Вигляд редактора опитувань Qualtrics

Серед недоліків даного веб-сервісу варто виділити наступні:

1. Вартість. Qualtrics є комерційним сервісом, тому його використання може бути витратним, особливо для малих організацій або індивідуальних користувачів;

2. Складність. Qualtrics є потужним інструментом, але в той же час він може бути складним для новачків. Інтерфейс та налаштування можуть вимагати певного часу та зусиль для освоєння;

3. Відсутність повного контролю над даними. Збираючи дані за допомогою Qualtrics, ви залежите від сервісу для збереження та захисту даних.

### 1.3 Постановка задачі

Провівши аналіз предметної області та сервісів-аналогів організації опитувань можна сформулювати задачу. Дана система повинна представляти із себе веб-сервіс, так як це являється найбільш універсальним рішенням в даному випадку. Такий вибір забезпечить доступність, відносну простоту реалізації, безпечність та кросплатформність розроблюваної системи.

Для створення конкурентоспроможної системи організації колективних опитувань було визначено наступні умови:

1. Привабливий дизайн. Застосунок повинен мати зручний, інтуїтивний та привабливий користувацький інтерфейс. Це стосується і редактору опитувань і власне самих опитувань;

2. Гнучкість редагування. Сервіс повинен мати широкі можливості для створювання та редагування опитувань;

3. Зручна модель поширення. Користувач повинен мати можливість поширювати створене опитування за допомогою унікально згенерованого посилання;

4. Функціональний збір даних. Потрібно реалізувати можливість експорту аналітичних даних для кожного окремого опитування;

5. Безпечність. Необхідно реалізувати можливість налаштування правд доступу для опитувань.

## 1.4 Висновок до розділу 1

Під час роботи над розділом проаналізовано предметну область, розглянуто: основні поняття інформаційної технології організації колективних опитувань і процеси та етапи, які будуть використовуватись при розробці організації колективних опитувань. Проаналізувавши об’єкт дослідження визначено: ключові вимоги до інформаційної технології, вхідні та вихідні дані для програмних засобів та вимоги до програмно-апаратного забезпечення системи.

Аналізу сервісів аналогів допоміг визначити їх характеристики, основні критерії щодо системи, описаної в межах даної магістерської роботи, переваги та недоліки систем-аналогів, область їх використання. Також розглянуто основі проблеми, що виникають при розробці подібних систем.

# 2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ОПИТУВАНЬ

## 2.1 Обґрунтування вибору засобів розробки

Розробка інформаційної технології для колективних опитувань включає створення бекенду та фронтенду. Вибір технологій базується на їхній ефективності, гнучкості та здатності задовольнити потреби проекту.

Фронтенд (або клієнтська частина) – це частина програмного забезпечення, яка відповідає за взаємодію з користувачем та відображення інформації на його пристрої. Основною метою фронтенду є створення інтерфейсу, який дозволяє користувачам взаємодіяти з веб-додатком, а також представлення даних у зручній та привабливій формі. Основні завдання фронтенду включають в себе:

1. Взаємодія з користувачем. Фронтенд забезпечує інтерактивність та відповідає на дії користувача. Це може бути введенням тексту, натисканням кнопок, перетягуванням об'єктів тощо;

2. Відображення інформації. Фронтенд відповідає за виведення даних користувачеві у зрозумілій та естетичній формі. Це може включати в себе роботу з HTML, CSS, та JavaScript для створення веб-сторінок та їхнього дизайну;

3. Забезпечення зручності використання. Фронтенд повинен бути інтуїтивно зрозумілим та забезпечувати зручність взаємодії з користувачем, щоб забезпечити позитивний досвід використання додатку;

4. Взаємодія з бекендом. Фронтенд взаємодіє з бекендом (серверною частиною) для отримання та відправлення даних. Це може включати в себе виконання запитів до сервера та обробку отриманих відповідей;

5. Оптимізація продуктивності. Фронтенд повинен працювати ефективно та швидко, забезпечуючи при цьому зручний та ефективний взаємодію з користувачем;

6. Кросплатформеність. Деякі фронтенд-технології та підходи дозволяють створювати додатки, які працюють на різних платформах та пристроях без значних змін.

Фронтенд може бути реалізований за допомогою різних технологій, таких як HTML, CSS, JavaScript, фреймворки та бібліотеки для розробки інтерфейсів, такі як React, Vue.js, Angular та інші.

Vue.js є основою фронтенду який використовує Options API для структурування компонентів. Це забезпечує чітку організацію коду та спрощує розробку.

Vuex використовується для управління станом додатку, що дозволяє централізовано керувати всіма даними в межах Vue-додатку.

VueRouter використовується для маршрутизації, що забезпечує гнучке управління навігацією.

Axios використовується для виконання HTTP-запитів до бекенду, що дозволяє легко обмінюватися даними з сервером.

Розглядаються як можливість використати Tailwind CSS для стилізації або Vuetify як UI фреймворку, залежно від вимог до дизайну та сумісності.

Vite використовується як інструмент розробки, що забезпечує швидке створення проекту, оптимізацію та hot module reloading.

Бекенд (або серверна частина) – це частина програмного забезпечення, яка відповідає за обробку логіки, зберігання та керування даними, а також за взаємодію з базою даних та іншими зовнішніми ресурсами. Основною метою бекенду є забезпечення функціональності веб-додатка, яка необхідна для коректної роботи, та надання необхідних даних фронтенду для відображення користувачам. Основні завдання бекенду включають в себе:

1. Логіка додатку. Бекенд містить бізнес-логіку, яка визначає, як додаток повинен взаємодіяти з даними, обробляти запити користувачів та виконувати різні операції;

2. Управління даними. Бекенд відповідає за зберігання, оновлення та видалення даних в базі даних або інших системах зберігання;

3. Безпека. Бекенд виконує заходи для захисту даних, оброблює аутентифікацію та авторизацію користувачів, а також забезпечує інші аспекти безпеки додатка;

4. Взаємодія з іншими службами. Бекенд може взаємодіяти з іншими службами, API або зовнішніми ресурсами для отримання додаткової функціональності або інформації;

5. Оптимізація продуктивності. Бекенд забезпечує ефективну роботу додатка, враховуючи оптимізацію запитів до бази даних, кешування та інші аспекти продуктивності;

6. Масштабованість. Бекенд повинен бути готовий масштабуватися для обробки зростаючої кількості користувачів та завдань;

7. Кросплатформеність. Деякі бекенд-технології дозволяють створювати додатки, які працюють на різних платформах та пристроях.

Бекенд може бути реалізований за допомогою різних технологій та мов програмування, таких як Python, Java, Ruby, Node.js, та інші. Часто використовують фреймворки для швидкої та ефективної розробки бекенд-додатків.

Основою бекенду є Django, потужний фреймворк для розробки веб-додатків, що забезпечує швидкість та гнучкість розробки.

Django REST Framework використовується для створення API, що спрощує процес розробки RESTful API.

Allauth використовується для автентифікації користувачів використовується allauth, що дозволяє легко інтегрувати соціальну автентифікацію та інші форми входу.

PostgreSQL використовується для системи управління базами даних, що забезпечує високу продуктивність, надійність та гнучкість.

Обраний набір технологій для фронтенду та бекенду забезпечує баланс між продуктивністю, гнучкістю та зручністю розробки. Vue.js разом з Vuex, VueRouter, Axios та потенційно Tailwind CSS або Vuetify на фронтенді, а також Django з Django REST Framework, allauth та PostgreSQL на бекенді, створюють

міцну основу для розробки ефективної системи колективних опитувань. В таблиці 2.1 наведено опис мов програмування:

Таблиця 2.1 – Мови програмування та їх характеристики

| Мова програмування | Характеристики |
|---|---|
| Python | – об'єктно-орієнтована;<br>– динамічна;<br>– структурована;<br>– має багато бібліотек;<br>– не потребує компілятора |
| Ruby | – зручний синтаксис, схожий на мову;<br>– стислий;<br>– Framework Ruby on Rails;<br>– мультипарадигма |
| PHP | – широко використовується;<br>– багато ресурсів;<br>– динамічна;<br>– гнучке середовище |

Мова програмування PHP існує вже більше двох десятиліть і зарекомендувала себе як потужне і надійне рішення, здобувши армію прихильників і шанувальників. Однак ми повинні визнати, що ця мова кодування поступово втрачає свою популярність на користь новіших і оптимізованих аналогів. Назва PHP означає Hypertext Preprocessor і позначає мову сценаріїв на стороні сервера, що означає, що програми, написані на ньому, працюють на веб-серверах і не залежать від веб-браузера. Проте з роками сфера його використання змінилася, і сьогодні мова кодування PHP входить до числа найкращих і найпопулярніших інструментів програмування для веб-розробки завдяки своїм багатьом перевагам. Це вважається дуже ефективною технологією, яка пропонує

зручний процес розробки з багатьма додатковими інструментами, які допомагають йому. Насправді, згідно з індексом популярності мови програмування (PYPL), PHP є п'ята за популярністю мова кодування в світі.

Мова PHP в основному використовується для веб-розробки, і вона справді хороша у цій області. Хоча спочатку вона використовувалась для створення динамічних веб-сторінок, розробники вважають за краще використовувати цю мову сценаріїв для створення серверної частини веб-додатків. Однак для початку PHP є мовою загального призначення, тому, якщо потрібно, вона може мати інші реалізації. Наприклад, можна створювати настільні програми за допомогою PHP. Більше того, починаючи з версії 5, PHP підтримує об’єктно-орієнтоване програмування, пропонуючи цілий новий набір можливостей [17].

Універсальність цієї мови сценаріїв є результатом її чудової здатності поєднуватися з іншими мовами програмування. Наприклад, розробники можуть писати розширення до PHP, використовуючи мову C, що дозволяє додати ще більше функцій. Крім того, PHP має велику кількість доступних бібліотек і фреймворків, які ще більше розширюють його можливості. Найпопулярнішими прикладами таких фреймворків є Laravel, Symfony, Phalcon, Zend Framework і Yii.

Два з найвідоміших прикладів програмного забезпечення, написаного на PHP – це Facebook і Wordpress. Wordpress є найпопулярнішою системою керування вмістом в Інтернеті: серед усіх веб-сайтів, які використовують такі системи, близько 48% з них використовують Wordpress. Завдяки великій кількості плагінів, як вбудованих, так і сторонніх, Wordpress підходить практично для будь-якої ролі. Ви можете використовувати його для створення блогу, фотогалереї, інтернет-магазину, порталу новин та багатьох інших типів веб-сайтів.

Facebook є ще одним очевидним прикладом того, що може робити PHP. Ця соціальна мережа вже давно перетворилася на складне середовище, яке має різноманітний спектр функцій, включаючи обмін миттєвими повідомленнями,

рекламу, ведення блогів, презентацію новин, відтворення відео та інші, але все ще базується на коді PHP, тісно пов’язаному з іншими веб-технологіями [18].

Популярність мови PHP є логічним результатом її численних переваг, які роблять її потужним та ефективним інструментом розробки. Розглянемо переваги PHP:

1. Великий вибір доступних спеціалістів. Популярність PHP породила численну спільноту розробників, частина з яких може бути потенційними кандидатами на роботу. Велика кількість наявних спеціалістів призводить до високої конкурентоспроможності та меншої затребуваної заробітної плати, що вигідно для зниження витрат на розробку. Крім того, цю мову досить легко вивчити та впровадити, тому навіть молодші розробники зазвичай можуть ефективно реалізувати базову функціональність програми;

2. Велика документація. Доступно багато навчальних посібників, посібників та інших довідкових матеріалів, які полегшують розробку та можуть стати допомогою та джерелом натхнення у важких ситуаціях. Ці матеріали також безцінні для початківців програмістів, що полегшують процес навчання з поступово зростаючими труднощами. І, як було описано вище, вивчення PHP є порівняно легким, хоча й не таким легким, як вивчення Python, наприклад, що ще більше додає його популярності;

3. Покращена швидкість завантаження. Використання PHP прискорює завантаження сторінок веб-сайтів у порівнянні з багатьма іншими технологіями веб-розробки. Наприклад, наразі PHP приблизно втричі швидший за Python для більшості сценаріїв використання. У свою чергу, менший час завантаження є важливим фактором рейтингу SEO, який сприяє подальшому просуванню веб-сайту, забезпечуючи конкурентні переваги. Вища швидкість додатків забезпечує задоволення клієнтів і, у поєднанні з іншими перевагами, допомагає створити та зберегти клієнтську базу;

4. Широкий вибір баз даних. PHP дозволяє підключатися практично до будь-якого типу бази даних. Найпоширенішим вибором є MySQL, головним чином тому, що він безкоштовний, ефективний і популярний серед розробників.

Інші надійні варіанти систем керування базами даних, сумісні з PHP – це mSQL, MS-SQL, SQLite, PostgreSQL тощо. Крім того, PHP можна однаково добре використовувати з ElasticSearch, Redis, MongoDB та іншими нереляційними базами даних. Таким чином, розробники не обмежуються використанням конкретної бази даних і можуть вибрати найбільш оптимальну для майбутнього додатка, враховуючи всі важливі фактори;

5. Недороге програмне забезпечення з відкритим кодом. PHP — це безкоштовна технологія, яка дає значну економію бюджету на розробку. Крім того, більшість інструментів розробки, які зазвичай використовуються в поєднанні з PHP, є програмним забезпеченням з відкритим кодом і можуть використовуватися безкоштовно; тим самим вони додатково знижують вартість проекту. Крім того, існують численні фреймворки, такі як Laravel і CodeIgniter, а також різні CMS, такі як Wordpress і Drupal, наприклад, які розширюють функціональність PHP і роблять процес розробки простішим і ефективнішим;

6. Дешевші послуги хостингу. Найпоширенішим сценарієм роботи веб-сайту PHP є стек LAMP. Це означає, що веб-сайт працює на веб-сервері Apache HTTP, розгорнутому в системі Linux, і використовує MySQL як базу даних. Всі ці компоненти безкоштовні, а стек добре перевірений, що передбачає скорочення часу та коштів на розробку;

7. Відмінна комбінованість з HTML. PHP пропонує вбудоване програмування HTML, що є причиною неймовірної синергії між цими двома технологіями. У більшості випадків PHP-скрипт не втручається в HTML-код веб-сторінки, а завершує його, залишаючись у межах, визначених тегами <?php ?>;

8. Хороша гнучкість. Завдяки гнучкості PHP може ефективно поєднуватися з багатьма іншими мовами програмування, щоб програмний продукт міг використовувати найбільш ефективну технологію для кожної окремої функції. Крім того, PHP є міжплатформною мовою, що означає, що розробники можуть використовувати будь-яку основну операційну систему – Windows, Linux, macOS – для виконання кодування. Така гнучкість значно полегшує процес розробки, роблячи його швидшим і менш дорогим;

9. Сумісність з хмарними сервісами. Нині багато сучасних продуктів, як правило, використовують рішення хмарних обчислень, як-от Amazon Web Services, для різних цілей. Програми, написані на PHP, підтримуються різними хмарними сервісами, такими як AWS Lambda, наприклад. Таким чином, PHP-додаток можна розгорнути на хмарному сервері і досягти відмінної масштабованості та інших корисних ефектів. Більш того, область хмарних обчислень не монополізована іншими мовами кодування, тому PHP зайняв своє місце в таких реалізаціях.

Хоча PHP, безсумнівно, корисний у сфері веб-розробки, він також має кілька недоліків, які не дозволяють йому домінувати в цій області. Розглянемо ці недоліки та дізнаємося, як вони можуть бути шкідливими для майбутнього програмного забезпечення та його бізнес-реалізації. Три основних недоліки PHP:

1. Зниження популярності. Хоча PHP є потужним інструментом, який підтримується великою спільнотою та великою довідковою документацією, існують простіші мови програмування для веб-програм. З цієї причини початківці розробники вважають за краще вивчати Python як свою першу мову і рідко замислюються про додавання PHP до своїх навичок. Зараз PHP домінує в сегменті веб-розробки, але, швидше за все, це зміниться в майбутньому. Згодом кількість фахівців скоротиться, а початківців розробників, які пропонують базові навички за низькою ціною, буде бракувати, тому вартість продуктів, створених на PHP, ймовірно, зросте;

2. Відсутність спеціалізованих бібліотек для сучасних потреб. Наприклад, машинне навчання зараз є гарячою тенденцією, і вона точно збереже свою популярність і в найближчому майбутньому. Незважаючи на те, що PHP має свій набір бібліотек, він не може конкурувати з Python у розробці веб-додатків за допомогою машинного навчання. Наразі PHP не може запропонувати настільки ж швидкі й ефективні альтернативи. Таким чином, якщо вашій програмі потрібна функціональність ML або може знадобитися це в майбутньому, коли ваш бізнес буде розширюватися, PHP не найкращий вибір;

3. Вади безпеки. Протягом багатьох років побоювання щодо безпеки продуктів на основі PHP все ще зберігаються з кількох причин. Однією з них є природа PHP з відкритим вихідним кодом, що означає, що можливі вразливості коду стають загальновідомими після їх виявлення. Таким чином, принаймні теоретично, протягом періоду між їх виявленням і виправленням у нових версіях мови програмування ці вразливості можуть бути використані будь-яким програмістом зі зловмисним умислом і відповідними навичками. Однак цей недолік можна в рівній мірі віднести до інших технологій з відкритим кодом загалом, оскільки він переважно стосується моделі з відкритим кодом, а не мови PHP зокрема.

Python – це дуже потужна мова сценаріїв на стороні клієнта, введена для того, щоб «оживити веб-сторінки». Це дозволяє створювати динамічний вміст для Інтернету. Python – це легка мова з відкритим вихідним кодом і дозволяє використовувати багато платформ. Вона не вимагає компіляції та інтерпретується за допомогою об'єктно-орієнтованих можливостей. Крім того, вона працює з різними іншими мовами програмування. І це є причиною його широкого використання в усьому світі. Багато популярних веб-сайтів та веб-програм, таких як Google, Amazon, PayPal тощо, використовують цю мову. Розширення файлу Python – .js.

Брендан Айх розробив Python у 1995 році, працюючи в Netscape Communications, Java, Scheme і Self, надихнувши його. Оскільки Microsoft стала смертельною загрозою, Netscape розпочав процес стандартизації, щоб запобігти доступу Microsoft до Python. Вони також співпрацювали з Sun Microsystems (зазвичай її називають Sun), щоб зламати монополію Microsoft. Причина, чому партнерство між Sun і Netscape працювало настільки ідеально, полягало в тому, що вони обидва мали одну ціль. Python спочатку був відомий як LiveScript від Netscape і Mocha. Пізніше вони перейменували LiveScript/Mocha на Python. Хоча Java і Python звучать схожі, вони не схожі. Вони мають дуже різний синтаксис, семантику та використання. Єдине спільне те, що обидва є торговими марками або зареєстрованими торговими марками Oracle у США та інших країнах.

Проаналізуємо навіщо потрібен Python. Більшість програм працюють завдяки взаємодії між клієнтом (пристроєм користувача) і віддаленим сервером. Клієнт запитує дані від сервера. Сервер отримує запит, обробляє його, а потім відповідає відповідно. Відповідь, надіслана назад, має формат, зрозумілий користувачеві, і тому є прийнятним для клієнта. Але цей процес вимагає часу та ресурсів. Python дозволяє перевіряти форми без введення сервера, зменшуючи трафік. Він надає чудові інструменти для більш інтерактивного та зручного веб-сайту. Деякі з основних функцій Python:

– Автозаповнення. Вікно пошуку дає пропозиції на основі того, що вже ввів користувач;

– Перевірка форми. якщо користувачі роблять помилку під час заповнення форми, Python негайно повідомляє їх про помилку, уникаючи повторного заповнення;

– Вирішує проблеми з макетом, щоб уникнути накладання елементів на сторінці;

– Додає анімацію на сторінку, щоб зробити її більш привабливою.

Python є однопоточним. Це означає, що інструкції виконуються послідовно, по одній. Це можливо за допомогою наступних компонентів:

1. Механізм Python. Це програма, що відповідає за переклад вихідного коду на машинну мову та його виконання на ЦП. Кожен сучасний браузер оснащений механізмом Python. Таким чином, немає необхідності завантажувати будь-яке додаткове програмне забезпечення. Двигун складається з 2 компонентів: куча пам’яті – тут відбувається виділення пам’яті; стек викликів – коли сценарій викликає функцію, інтерпретатор спочатку додає її до стеку викликів, а потім починає обробляти;

2. Час виконання. Механізм Python працює всередині середовища для додаткових функцій, які використовуються під час виконання. Хоча Python є однопоточним, середовище виконання складається з пулу потоків. Це дозволяє Python одночасно працювати у фоновому режимі, поки користувач переглядає, не перериваючи виконання програми;

3. Петля подій. Цей механізм керує всіма потоками з пулу потоків у порядку їх виконання, тобто обробляє зворотні виклики. Зворотний виклик – це фрагмент коду, який виконується, коли відбувається певна подія, тобто клацання мишею. Якщо відбувається подія, середовище поміщає зворотний виклик в обробник подій у циклі подій. Зворотні виклики завжди виконуються по одному.

Нижче наведено основні функції Python:

– Підтримує концепції об'єктно-орієнтованого програмування;

– Не залежить від платформи та чутливий до регістру;

– Надає різні вбудовані функції, такі як alert(), prompt() тощо;

– Можливість обробки винятків;

– Дозволяє використовувати функції з будь-яким ім'ям або без нього. Функція без імені є анонімною функцією.

Застосування Python:

– Веб-розробка. Веб-розробка є окремою мовою для створення веб-сторінок. Він також підтримує зовнішні програми, такі як документи PDF, запущені віджети тощо. Це також додає різні спеціальні ефекти до сторінки, як-от графіку;

– Веб-додатки. Веб-програми взаємодіють з браузером, не надсилаючи повідомлення між браузером і сервером. За допомогою різних фреймворків, доступних на ринку, дуже легко створювати інтерактивні сторінки. Це економить час і зусилля, необхідні розробнику для створення веб-додатку;

– Презентації. Презентація Python надає зручні бібліотеки та рамки для презентацій. Вона забезпечує вишукані теми, які не надто кричущі;

– Серверні програми. Сервер-додатки Python також дуже корисний для створення серверних програм і швидший, ніж інші серверні мови. NodeJS — це безкоштовне серверне середовище з відкритим вихідним кодом, яке використовується для цього. Воно дозволяє генерувати динамічний веб-контент, змінювати базу даних, збирати дані форм тощо;

– Веб-сервери. Веб-сервери NodeJS має вбудований модуль (набір функцій), який дозволяє створити HTTP-сервер. Незважаючи на те, що Python почався як сценарій на стороні клієнта, тепер Python повністю здатний також виконувати сценарії на стороні сервера;

– Ігри. Разом із HTML5, Python допомагає у розробці ігор. Бібліотека EaselJS надає прості рішення для роботи з насиченою графікою. Вона також має API, знайомий для багатьох розробників.

– Використовуючи Python, малювати графіку за допомогою HTML стало легше на веб-сторінці. Полотно є без кордонів або вмісту, тому користувачі можуть створювати власне мистецтво. Python забезпечив користувача середовищем для різних проектів цифрового мистецтва;

– Програми для розумних годинників. Python містить бібліотеку PebbleJS, яка надає прості рішення для роботи з насиченою графікою. Розробник має доступ до багатьох функцій програми, створених для розумних годинників;

– Мобільні програми. Python забезпечує структуру під назвою PhoneGap, яка підтримує мобільні додатки. React Native сьогодні також служить цій же меті. Маючи пристойні знання HTML, CSS і Python, можна створювати чудові програми;

– Машинне навчання. Python представив машинне навчання разом з усіма попередніми функціями. Різні фреймворки, такі як Keras.js, BrainJS, Compromise і WebDNN, покращують функціональність Python у сфері AI.

Розглянемо переваги Python:

1. Швидкість – Python має тенденцію працювати дуже швидко, оскільки часто запускається безпосередньо в браузері клієнта. Поки він не потребує зовнішніх ресурсів, Python не сповільнюється викликами сервера. Крім того, всі основні браузери підтримують JIT компіляцію для Python, що означає, що немає необхідності компілювати код перед його запуском;

2. Простота – синтаксис Python був натхненний Java і його відносно легко вивчити порівняно з іншими популярними мовами, такими як C++;

3. Популярність – Python є скрізь в Інтернеті, а з появою Node.js все частіше використовується у бекенд. Існує незліченна кількість ресурсів для вивчення Python. Як StackOverflow, так і GitHub демонструють все більшу кількість проектів, які використовують Python, і очікується, що популярність, яку він отримав за останні роки, лише збільшиться;

4. Сумісність – на відміну від PHP або інших мов сценаріїв, Python можна вставити на будь-яку веб-сторінку. Python можна використовувати в багатьох різних програмах завдяки підтримці інших мов, таких як Pearl і PHP;

5. Завантаження сервера – Python є клієнтським, тому він загалом зменшує попит на сервери, і простим додаткам може взагалі не знадобитися сервер;

6. Розширені інтерфейси – Python можна використовувати для створення таких функцій, як перетягування компонентів, таких як повзунки, які значно покращують інтерфейс користувача та роботу сайту;

7. Розширена функціональність – розробники можуть розширити функціональність веб-сторінок, написавши фрагменти Python для сторонніх доповнень, таких як Greasemonkey;

8. Універсальність – існує багато способів використання Python через сервери Node.js. Якщо ви завантажуєте Node.js за допомогою Express, використовуєте базу даних документів, як-от MongoDB, і використовуєте Python на інтерфейсі для клієнтів, можна розробити цілу програму Python від початку до кінця, використовуючи лише Python;

9. Оновлення – з появою ECMAScript 5 (специфікація сценаріїв, яку використовує Python), ECMA International було присвячено оновленню Python щорічно.

Розглянемо недоліки Python:

1. Клієнтська безпека – оскільки код Python виконується на стороні клієнта, помилка і прорахунки можуть іноді бути використані в зловмисних цілях. Через це деякі люди вирішують повністю вимкнути Python;

2. Підтримка браузерів – у той час як серверні скрипти завжди дають той же результат, різні браузери іноді інтерпретувати код Python різному. Ці

відмінності мінімальні, і не доводиться турбуватися про це, поки тестується скрипт у всіх основних браузерах.

Отже, вище було проаналізовано такі мови програмування як PHP, Ruby та Python, описано їх основні характеристики, переваги та недоліки. Згідно цієї інофрмації прийнято рішення використати для розробки інформаційної технології мову програмування Python через її переваги, такі як швидкість, сумісність та універсальність.

## 2.2 Обґрунтування вибору мови програмування

Вибір мови програмування для розробки інформаційної технології є ключовим рішенням, яке впливає на продуктивність, масштабованість, та підтримку проекту. У цій розробці основними мовами програмування є JavaScript для фронтенду та Python для бекенду.

Використання JavaScript для розробки фронтенду обґрунтоване кількома ключовими факторами:

– Клієнтська взаємодія. JavaScript надає можливість створювати динамічний та інтерактивний веб-інтерфейс. Завдяки JavaScript, можна взаємодіяти з користувачем без перезавантаження сторінки, що поліпшує загальний досвід використання.

– Асинхронний код. JavaScript використовує асинхронні запити, що дозволяє виконувати операції без очікування завершення попередніх. Це особливо важливо для здійснення асинхронних запитів до сервера, отримання та обробки даних без блокування інтерфейсу;

– Розширені можливості взаємодії з DOM. JavaScript надає зручні та потужні інструменти для маніпулювання DOM (Document Object Model). Це дозволяє динамічно змінювати вміст сторінки, створювати анімації, реагувати на події користувача та інше;

– Багатофункціональність. Завдяки екосистемі бібліотек та фреймворків (таких як React, Angular, Vue.js), JavaScript дозволяє розробляти

багатофункціональні та масштабовані фронтенд-додатки. Ці інструменти прискорюють розробку та дозволяють ефективно керувати станом додатку;

– Сучасні стандарти. Стандарти мови JavaScript постійно оновлюються, і багато з нововведень спрямовані на поліпшення продуктивності та зручності розробки. Наприклад, введення async/await у JavaScript спрощує асинхронне програмування;

– Кросплатформеність. JavaScript може використовуватися для розробки фронтенду для різних браузерів та платформ. Завдяки цьому, можна створювати веб-додатки, які працюють на різних пристроях та операційних системах;

– Широке сприйняття. JavaScript - одна з найпоширеніших мов програмування для фронтенду. Більшість сучасних браузерів підтримують виконання JavaScript, що робить його досить універсальним та доступним інструментом для розробки веб-додатків.

Враховуючи ці переваги, використання JavaScript для розробки фронтенду стало стандартом у розробці.

Використання мови програмування Python для розробки бекенду (серверної частини розробки) має кілька обґрунтувань:

– Простота та Читабельність. Python відомий своєю простотою та читабельністю коду. Це дозволяє розробникам швидко створювати та обслуговувати код, що є особливо важливим при роботі в команді та підтримці коду;

– Розширюваність та Масштабованість. Python підтримує різні парадигми програмування і має широкий набір бібліотек і фреймворків, які сприяють розширюваності та масштабованості бекенд-додатків. Фреймворки, такі як Django, Flask і FastAPI, дозволяють швидко створювати потужні та ефективні веб-додатки;

– Швидкість розробки. Python відомий своєю високою продуктивністю та швидкістю розробки. Велика кількість готових рішень і

стандартизований синтаксис спрощують створення функціонального бекенду в короткі строки;

– Спільнота та Екосистема. Python має велику та активну спільноту розробників. Це означає, що завжди є доступ до великої кількості ресурсів, документації, форумів та інших джерел підтримки. Багата екосистема бібліотек та фреймворків також полегшує вирішення різних завдань;

– Підтримка асинхронного програмування. Введення асинхронних функцій та корутин в мові Python (зокрема, у версії 3.5 і вище) дозволяє ефективно вирішувати проблеми асинхронного програмування та забезпечує високу продуктивність для веб-додатків, що вимагають багатозадачності;

– Широка підтримка великих компаній. Python є популярним вибором для великих IT-компаній. Багато великих веб-сервісів та додатків, таких як Instagram, Dropbox та Spotify, базуються на Python для своїх бекенд-систем;

– Безпека. Python відомий своєю високою рівнем безпеки. Розробники Python активно вдосконалюють та поновлюють безпеку мови, що робить її надійною для розробки безпечних веб-додатків.

Враховуючи ці переваги, використання Python для розробки бекенду є зручним та популярним вибором у веб-розробці.

Вибір JavaScript та Python для розробки інформаційної технології організації колективних опитувань є обґрунтованим рішенням. Ці мови забезпечують ідеальне поєднання гнучкості, продуктивності, та легкості у використанні, що є критично важливим для створення ефективних та надійних веб-додатків.

## 2.3 Математична модель проведення колективних опитувань

Математична модель проведення колективних опитувань є ключовим інструментом для аналізу, планування та оптимізації процесу збору та аналізу даних. Така модель дозволяє формалізувати процес опитування, визначити

ключові параметри та залежності, а також спрогнозувати результати. Для більшої наочності наведемо UML -діаграму взаємодії користувача із системою


Рисунок 2.1 – Взаємодія користувача із системою

Основні компоненти моделі:

1. Популяція (N). Загальна кількість осіб у групі, яка є предметом опитування;

2. Вибірка (n). Підмножина популяції, яка фактично описується. Розмір вибірки може варіюватися залежно від потреб дослідження та доступних ресурсів;

3. Відповіді (R). Відповіді, отримані від учасників опитування. Вони можуть бути кількісними або якісними;

4. Ймовірність відповіді (P). Ймовірність того, що учасник опитування дасть відповідь. Це важливо для врахування можливих відсутніх даних;

5. Помилка вибірки (E). Помилка, що виникає через використання даних вибірки замість повної популяції. Вона залежить від розміру вибірки та розподілу відповідей.

Математичне Формулювання:

1. Розрахунок розміру вибірки (2.1):

$$n = \frac{N}{1+N(e^2)}, \tag{2.1}$$

де ( e ) - прийнятний рівень помилки вибірки.

2. Оцінка представництва вибірки (2.2):

$$P_r = \frac{n}{N}, \tag{2.2}$$

де ( Pr ) - ймовірність представництва вибірки.

3. Розрахунок стандартної помилки вибірки (SE) (2.3):

$$SE = \sqrt{\frac{p(1-p)}{n}}, \tag{2.3}$$

де ( p ) - ймовірність успішної відповіді.

4. Конфіденційний інтервал (CI) (2.4):

$$CI = \bar{x} \pm Z \times SE, \tag{2.4}$$

де ( $x$ ) - середнє значення відповідей, ( $Z$ ) - значення Z-рахунку, що відповідає бажаному рівню довіри.

Математична модель проведення колективних опитувань дозволяє систематизувати та оптимізувати процес збору даних. Вона включає в себе розрахунок розміру вибірки, оцінку помилки вибірки, а також визначення конфіденційних інтервалів для результатів. Це забезпечує високу якість та надійність отриманих даних, що є критично важливим для ефективного аналізу та прийняття обґрунтованих рішень.

## 2.4 Розробка структури інформаційної технології колективних опитувань

Структура інформаційної технології для колективних опитувань включає в себе декілька ключових компонентів, які забезпечують збір, обробку, аналіз та представлення даних. Ця структура повинна бути гнучкою, масштабованою та ефективною для задоволення різноманітних потреб користувачів.

Основні Компоненти:

1. Інтерфейс користувача (Frontend):

– Веб-інтерфейс. Розроблений з використанням Vue.js, Vuex, VueRouter, і Axios для забезпечення інтерактивного та зручного користувацького досвіду;

– Мобільний інтерфейс. Опціонально, може бути розроблений для забезпечення доступу до опитувань через мобільні пристрої.

2. Серверна частина (Backend):

– Веб-сервер. Використання Django та Django REST Framework для обробки запитів, автентифікації користувачів та взаємодії з базою даних;

– База даних. PostgreSQL для зберігання даних опитувань, відповідей та інформації про користувачів.

3. Модуль аналізу даних:

– Обробка даних. Функціонал для агрегації, фільтрації та аналізу зібраних даних;

– Звітність. Генерація звітів та візуалізацій для представлення результатів опитувань.

4. Інтеграція та Розширення:

– API. REST або GraphQL API для інтеграції з зовнішніми системами та сервісами;

– Плагіни/Розширення. Можливість додавання додаткових модулів або інтеграцій з іншими інструментами.

5. Безпека та Конфіденційність:

– Автентифікація та Авторизація. Забезпечення безпечного доступу до системи;

– Захист даних. Застосування сучасних методів шифрування та захисту даних.

Архітектура Системи:

– Клієнт-серверна архітектура. Розділення функціоналу між клієнтською (фронтенд) та серверною (бекенд) частинами;

– Мікросервісна архітектура. Опціонально, для більшої масштабованості та гнучкості, система може бути розроблена з використанням мікросервісної архітектури. Для більш наглядного приставляння архітектури наведена загальна UML на рис.2.2

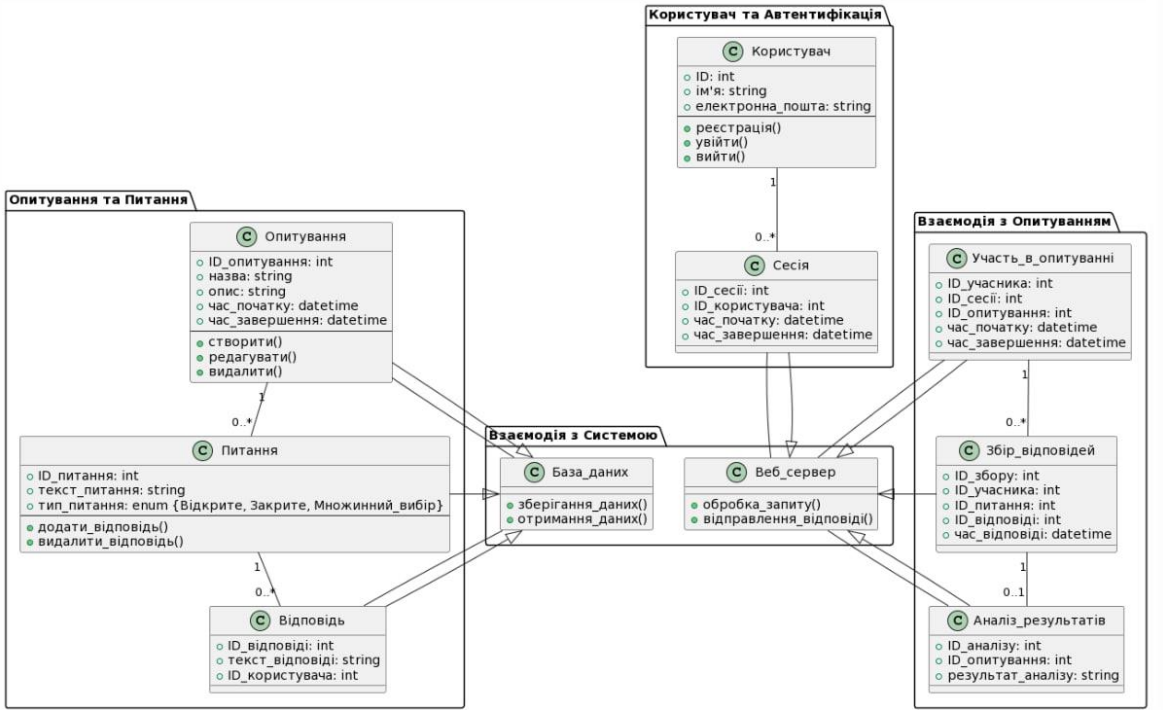


Рисунок 2.2 – Загальна UML-діаграма опитувань

Розроблена структура інформаційної технології для колективних опитувань забезпечує комплексний підхід до збору, обробки, аналізу та представлення даних наведена на рисунку 2.3

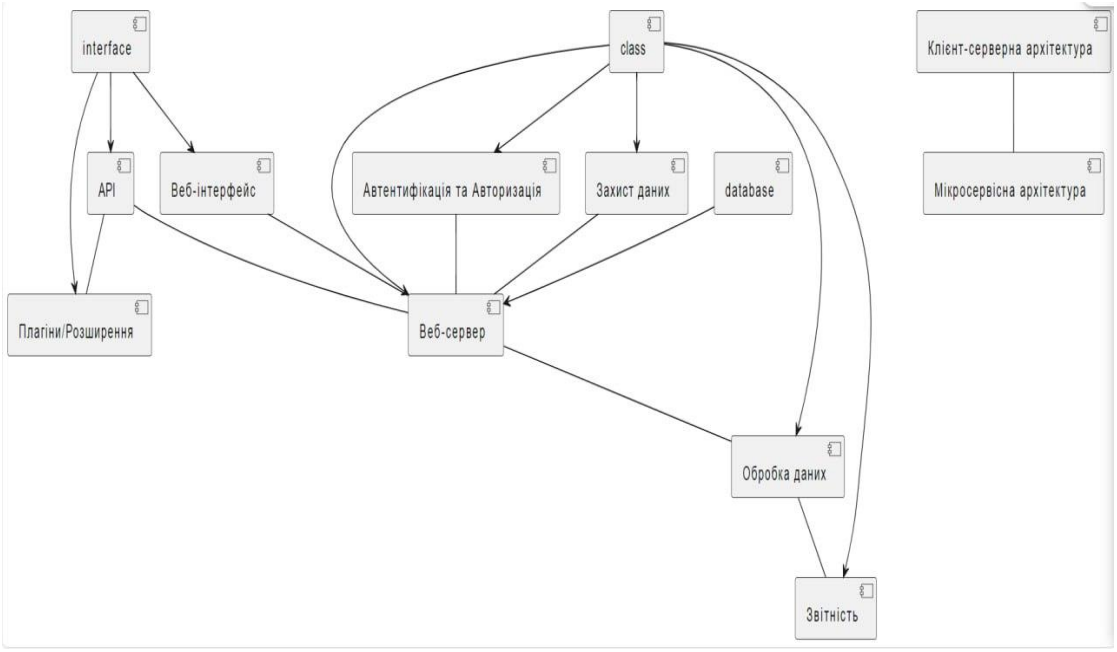


Рисунок 2.3 – Загальна UML-роботи системи

Вона включає в себе гнучкий фронтенд, потужний бекенд, розширений модуль аналізу даних, інтеграційні можливості та високий рівень безпеки, що робить її ефективним інструментом для проведення колективних опитувань.

## 2.5 Розробка алгоритмів функціонування інформаційної технології проведення колективних опитувань

Алгоритми функціонування інформаційної технології для проведення колективних опитувань є ключовими для забезпечення ефективності, точності та надійності процесу збору та аналізу даних. Вони включають механізми створення опитувань, збору відповідей, обробки даних та представлення результатів.

Основні Алгоритми:

1. Алгоритм створення опитування наведено на рисунку 2.4:

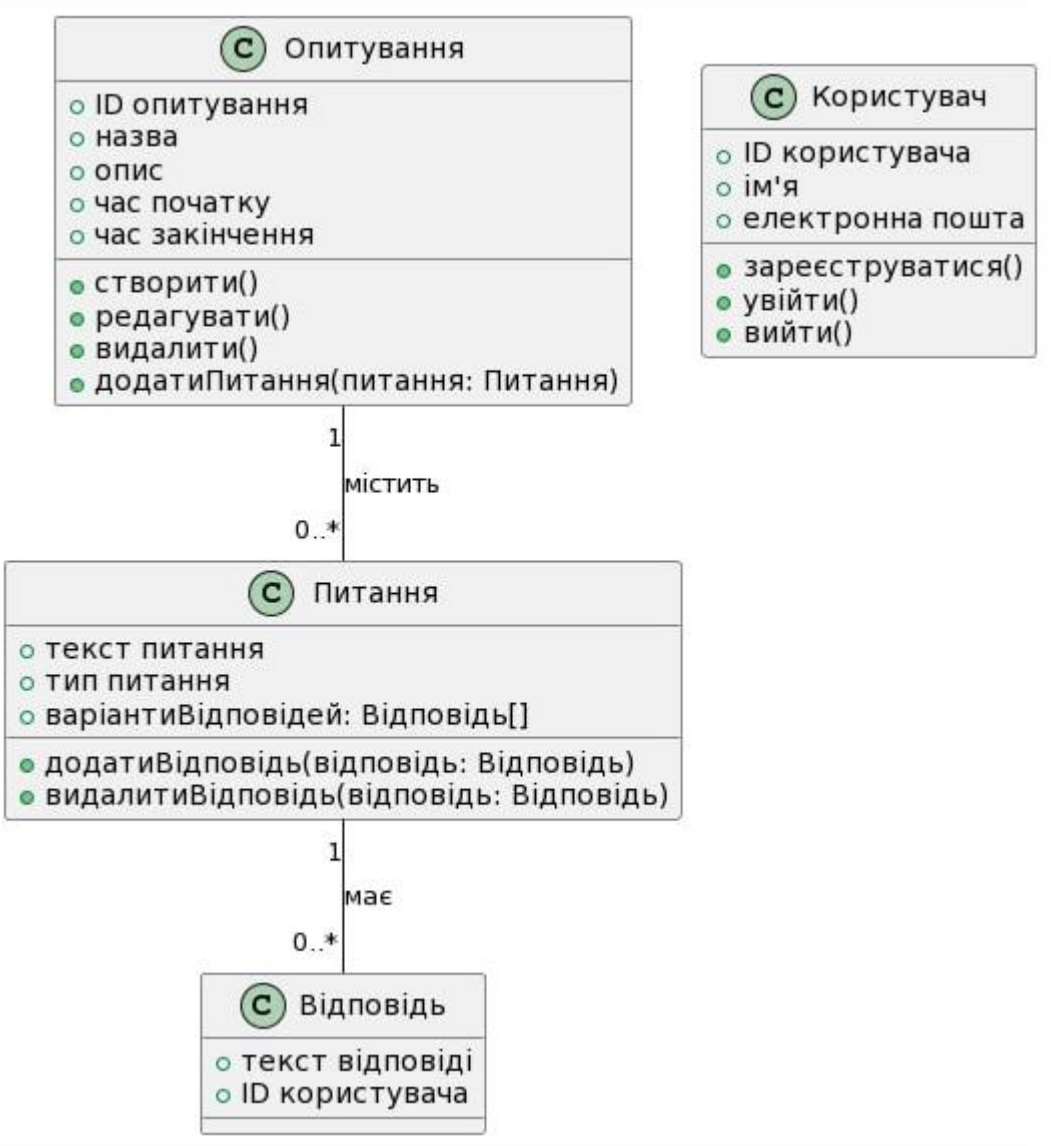


Рисунок 2.4 – Класова структура взаємозвязків

2. Алгоритм розповсюдження опитування наведено на рисунку 2.5:

- Вхід. Ідентифікатор опитування, цільова аудиторія;

- Процес:

2.1. Визначення цільової аудиторії;

2.2. Розсилка запрошень для участі в опитуванні (емейл, соціальні мережі тощо);

- Вихід. Статус розповсюдження.


Рисунок 2.5 – Фрагмент алгоритму проведення опитувань

3. Алгоритм збору відповідей наведено на рисунках з 2.6 по 2.8:

- Вхід. Відповіді учасників;

- Процес:

3.1. Збір відповідей через інтерфейс користувача;

3.2. Валідація та обробка відповідей;

3.3. Збереження відповідей в базі даних;

- Вихід. Зібрані дані.


Рисунок 2.6 – Фрагмент алгоритму збору відповіді учасників


Рисунок 2.7 – Фрагмент алгоритму вибору параметрів опитування


Рисунок 2.8 – Фрагмент головного алгоритму вибору

4. Алгоритм аналізу даних наведено на рисунку 2.9:

- Вхід. Зібрані дані;

- Процес:

4.1. Агрегація даних;

4.2. Статистичний аналіз (середні значення, медіана, мода тощо);

4.3. Виявлення тенденцій та закономірностей;

- Вихід. Аналітичний звіт.


Рисунок 2.9 – Фрагмент алгоритму вибору опитувань

5. Алгоритм представлення результатів:

- Вхід. Аналітичний звіт;

- Процес:

5.1. Вибір формату представлення (текст, графіки, діаграми);

5.2. Генерація візуалізацій;

5.3. Підготовка звіту для публікації або друку;

- Вихід. Готовий звіт.

Розроблені алгоритми забезпечують повний цикл функціонування інформаційної технології для проведення колективних опитувань. Від створення та розповсюдження опитувань до збору відповідей, їх аналізу та представлення результатів - кожен крок оптимізований для забезпечення ефективності та точності процесу. Це дозволяє отримати високоякісні дані для подальшого аналізу та прийняття обґрунтованих рішень.

Створення повноцінної UML (Unified Modeling Language) діаграми в текстовому форматі може бути складним, але я можу надати опис основних компонентів UML діаграми для системи колективних опитувань. Для повноцінного візуального представлення рекомендується використовувати спеціалізоване програмне забезпечення для створення UML діаграм, таке як Lucidchart, Microsoft Visio, або інші аналогічні інструменти. Фрагмент класової взаємодії модулів системи наведено на рисунку 2.10.

Основні Компоненти UML для системи колективних опитувань:

1. Діаграма Випадків Використання (Use Case Diagram):

Актори:

- Користувач (може бути учасником або адміністратором опитування);
- Система (веб-сервер, база даних).

Випадки використання:

- Створення опитування;
- Розповсюдження опитування;
- Участь в опитуванні;
- Збір відповідей;
- Аналіз результатів опитування;
- Перегляд результатів.

2. Діаграма Класів (Class Diagram)

Основні класи:

- Опитування;
- Атрибути: ID опитування, назва, опис, час початку, час закінчення;
- Методи: створити, редагувати, видалити.

Питання:

- Атрибути: текст питання, тип питання (відкрите, закрите, множинний вибір);
- Методи: додати відповідь, видалити відповідь.

Відповідь:

- Атрибути: текст відповіді, ID користувача;

Користувач:

- Атрибути: ID користувача, ім'я, електронна пошта;
- Методи: зареєструватися, увійти, вийти.

3. Діаграма Послідовностей (Sequence Diagram)

Сценарій:

- Користувач входить в систему;
- Користувач вибирає або створює нове опитування;
- Система обробляє запит та відображає форму опитування;
- Користувач заповнює опитування та надсилає відповіді;
- Система зберігає відповіді та надає зворотний зв'язок.

4. Діаграма Активностей (Activity Diagram)

Процес участі в опитуванні:

- Початок;
- Вхід в систему;
- Вибір опитування;
- Відповідь на питання;
- Надсилання відповідей;
- Кінець.

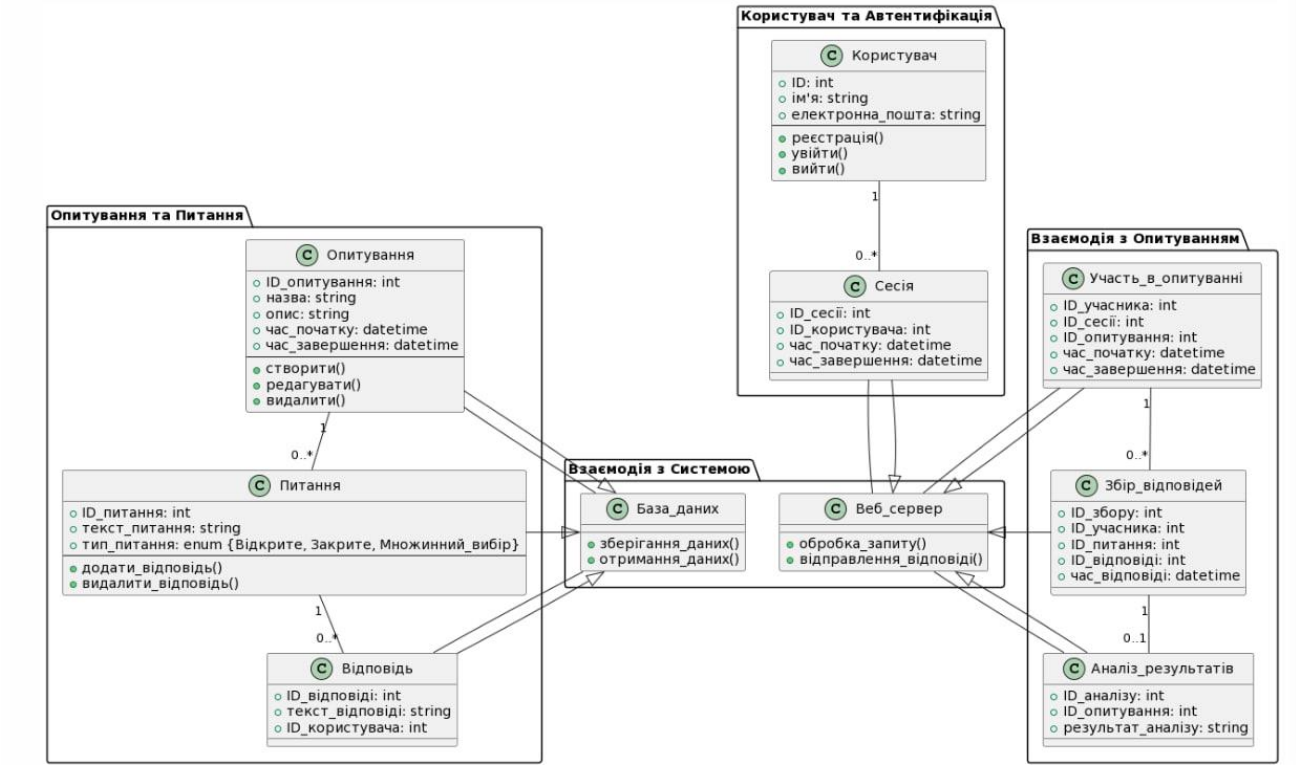


Рисунок 2.10 – Фрагмент класової взаємодії модулів системи

Ці діаграми є лише загальними прикладами того, як можна візуалізувати структуру та процеси системи колективних опитувань. Для детального аналізу та проектування системи, кожна діаграма повинна бути деталізовано розроблена з урахуванням специфіки проекту.

## 2.6 Висновок до розділу 2

У розділі було розглянуто ключові аспекти розробки інформаційної технології для проведення колективних опитувань. Цей розділ охоплює важливі етапи від обґрунтування вибору технологій до розробки конкретних алгоритмів функціонування системи.

Вибір технологій, включаючи Vue.js, Vuex, VueRouter, Axios для фронтенду та Django, Django REST Framework, allauth, PostgreSQL для бекенду, було зроблено на основі їхньої ефективності, гнучкості та здатності

задовольнити вимоги проекту. Використання цих технологій забезпечує створення надійної, масштабованої та легкої у підтримці системи.

JavaScript та Python були обрані за основу фронтенду та бекенду відповідно. Ці мови забезпечують гнучкість, широку підтримку та велику кількість ресурсів для розробки.

Розробка математичної моделі для проведення опитувань дозволяє точно планувати, аналізувати та оптимізувати процес збору даних, включаючи визначення розміру вибірки, оцінку помилок та аналіз результатів.

Розроблена структура включає в себе фронтенд, бекенд, модуль аналізу даних, інтеграційні можливості та забезпечення безпеки. Це забезпечує комплексний підхід до збору, обробки, аналізу та представлення даних.

Розроблені алгоритми охоплюють весь процес проведення опитувань, від створення до аналізу даних та представлення результатів. Це забезпечує ефективність та точність у зборі та обробці інформації.

У підсумку, надається чітке розуміння технічних аспектів розробки інформаційної технології для колективних опитувань. Використання сучасних технологій та ефективних алгоритмів забезпечує створення надійної та гнучкої системи, яка може бути адаптована до різних потреб користувачів та сценаріїв використання.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ОПИТУВАНЬ

## 3.1 Розробка структури бази даних інформаційної технології організації колективних опитувань

Структура бази даних є фундаментальною частиною інформаційної технології для організації колективних опитувань. Вона повинна бути розроблена таким чином, щоб ефективно зберігати, обробляти та надавати доступ до даних опитувань, відповідей учасників, а також іншої важливої інформації.

Основні сутності бази даних:

1. Опитування (Surveys):

- ID опитування;
- Назва;
- Опис;
- Дата створення;
- Дата закінчення (опціонально);
- Статус (активне, закрите тощо).

2. Питання (Questions):

- ID питання;
- ID опитування (зв'язок з таблицею Опитувань);
- Текст питання;
- Тип питання (одиночний вибір, множинний вибір, текстова відповідь тощо).

3. Відповіді (Responses):

- ID відповіді;

- ID питання (зв'язок з таблицею Питань);
- ID учасника (зв'язок з таблицею Учасників);
- Відповідь (текст, вибраний варіант тощо).

4. Учасники (Participants):
- ID учасника;
- Ім'я;
- Електронна пошта;
- Інша інформація (вік, стать, професія тощо).

5. Логи (Logs):
- ID запису;
- Тип події (вхід, вихід, відповідь на питання тощо);
- Час події;
- ID учасника або ID адміністратора.

Схема бази даних повинна відображати відносини між цими сутностями. Наприклад:

- Кожне опитування може містити багато питань.
- Кожне питання може мати багато відповідей.
- Кожен учасник може брати участь у багатьох опитуваннях та давати відповіді на різні питання.

Реалізація:
- Використання PostgreSQL. Ця СУБД підтримує реляційні моделі та забезпечує високу продуктивність та надійність;
- Нормалізація даних. Важливо забезпечити нормалізацію для уникнення дублювання даних та забезпечення цілісності;

- Індексація. Використання індексів для покращення швидкості пошуку та запитів.

Ефективно спроектована структура бази даних є критично важливою для успішної реалізації інформаційної технології організації колективних опитувань. Вона повинна бути гнучкою, масштабованою та здатною забезпечити швидкий доступ до даних, їх обробку та аналіз. Ретельне планування та реалізація бази даних забезпечує міцну основу для всієї інформаційної системи.


Рисунок 3.1 – Головна сторінка

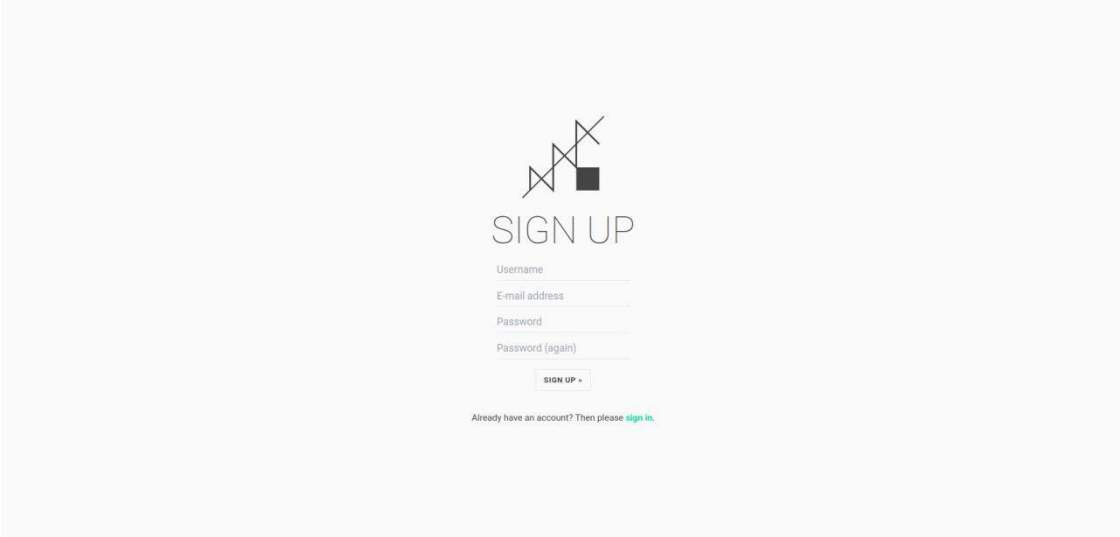


Рисунок 3.2 – Створення акаунту

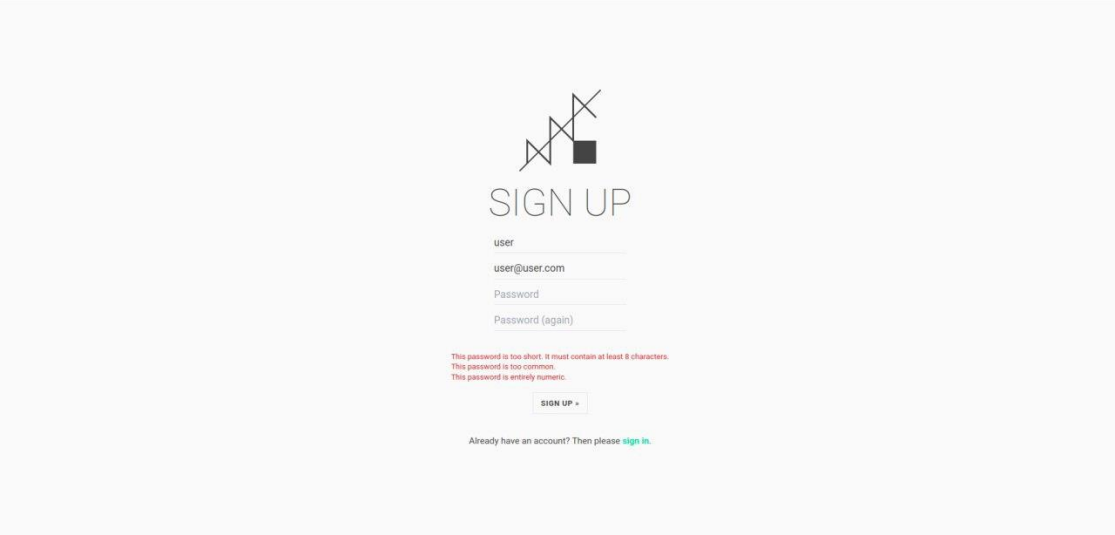


Рисунок 3.3 – Введення даних


Рисунок 3.3 – Підтвердження електронної пошти

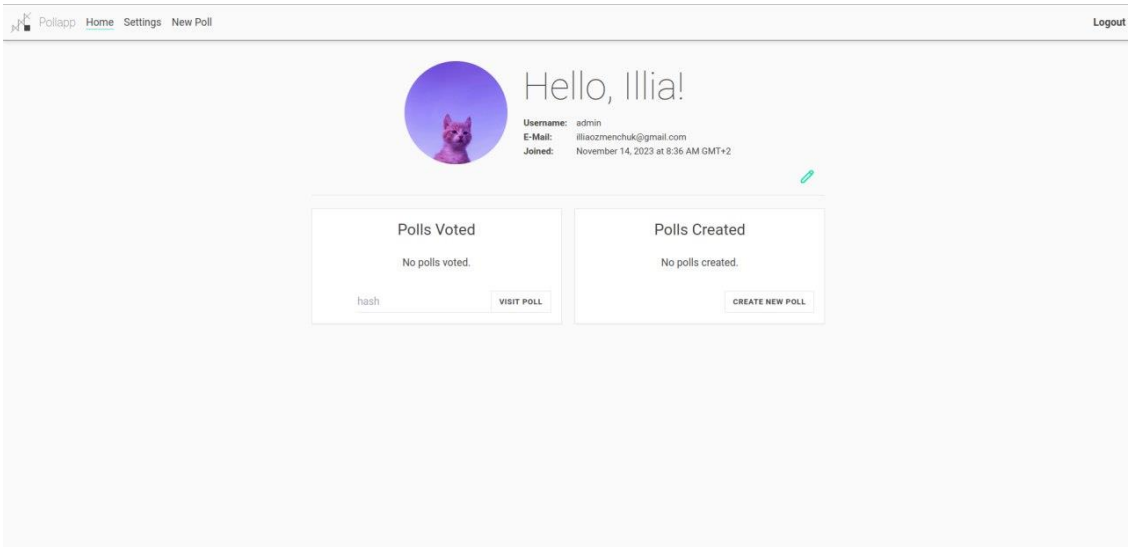


Рисунок 3.4 – Меню користувача

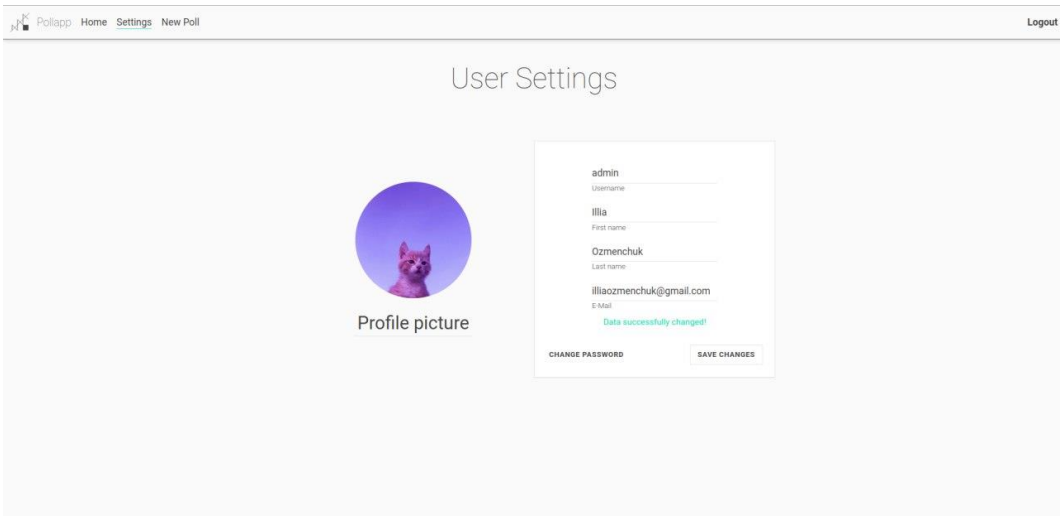


Рисунок 3.5 – Налаштування користувача

## 3.2 Розробка загальної структури програмного забезпечення для колективних опитувань

Загальна структура програмного забезпечення для колективних опитувань включає в себе декілька ключових компонентів, які забезпечують створення, розповсюдження, збір відповідей та аналіз результатів опитувань. Ця структура повинна бути гнучкою, інтуїтивно зрозумілою та ефективною.

Основні Компоненти:

1. Інтерфейс користувача (Frontend):

- Веб-клієнт. Розроблений з використанням Vue.js, Vuex, VueRouter, і Axios для забезпечення інтерактивного та зручного користувацького досвіду;

- Адаптивний дизайн. Забезпечення комфортного відображення на різних пристроях, включаючи мобільні.

2. Серверна частина (Backend):

- API. Розробка RESTful API з використанням Django та Django REST Framework для обробки запитів від фронтенду;

- Логіка обробки даних. Реалізація бізнес-логіки для створення опитувань, збору відповідей, та їх аналізу.

3. База даних:

- Сховище даних. Використання PostgreSQL для зберігання даних опитувань, відповідей, користувачів та іншої важливої інформації.

- Модель даних. Розробка нормалізованої моделі даних, яка відображає відносини між різними сутностями.

4. Безпека:

- Автентифікація та авторизація. Забезпечення безпечного доступу до системи через механізми, такі як OAuth, JWT.

- Захист даних. Застосування сучасних методів шифрування та захисту даних.

5. Інтеграція та Розширення:

- API для зовнішніх інтеграцій. Можливість інтеграції з іншими системами та сервісами через API;

- Модульність. Розробка системи з можливістю легкого додавання нових функцій та компонентів.

Архітектура Системи:

- Мікросервісна або монолітна архітектура. Вибір між мікросервісною та монолітною архітектурою залежить від масштабу проекту та вимог до масштабування;

- Розділення відповідальності. Чітке розділення функціоналу між фронтендом та бекендом.

Розробка загальної структури програмного забезпечення для колективних опитувань вимагає інтеграції різних компонентів та технологій для створення ефективної, гнучкої та безпечної системи

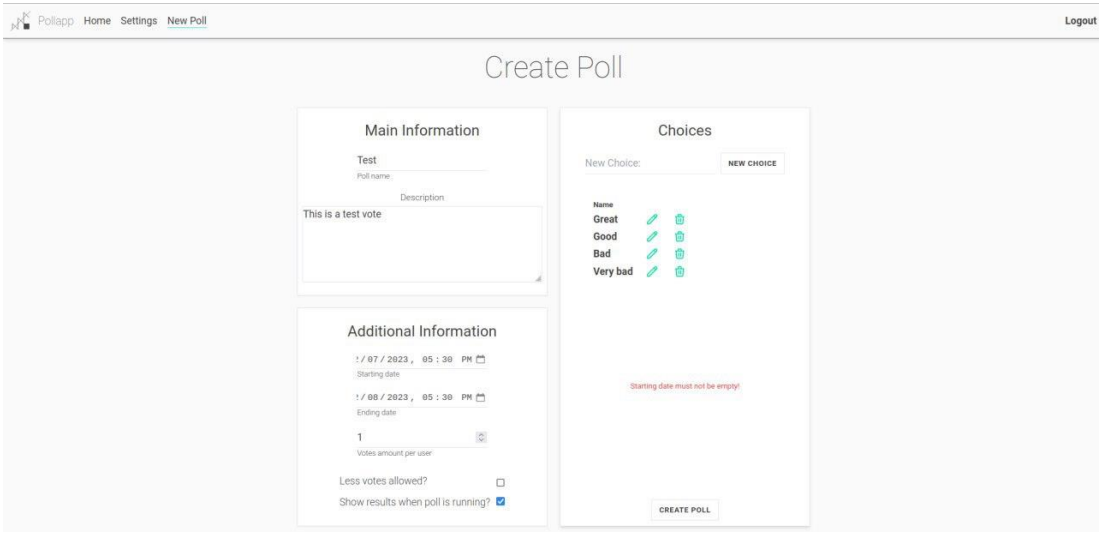


Рисунок 3.6 – Меню створення опитування

Важливо забезпечити зручний інтерфейс для кінцевих користувачів, надійну серверну обробку даних, ефективне зберігання даних, а також високий рівень безпеки. Розробка такої системи вимагає глибокого розуміння потреб користувачів та бізнес-процесів, що стоять за колективними опитуваннями.

## 3.3 Розробка серверної частини веб-застосунку для колективних опитувань

Серверна частина веб-застосунку для колективних опитувань відіграє ключову роль у забезпеченні функціональності, безпеки та ефективності обробки даних. Вона включає в себе API для взаємодії з фронтендом, логіку обробки даних, взаємодію з базою даних, а також забезпечення безпеки.

На початку визначається моделі для збереження інформації про опитування, вибори та виборців у Django-додатку. Розглянемо кожну модель та їхні основні характеристики:

Модель Poll:

```
class Poll(models.Model):
    title = models.CharField(max_length=128)
    description = models.CharField(max_length=1024)
    votes_amt = models.PositiveSmallIntegerField(default=1)
    less_allowed = models.BooleanField(default=False)
    show_while_running = models.BooleanField(default=False)

    date_created = models.DateTimeField(auto_now=True)
    date_to_start = models.DateTimeField(null=True)
    date_to_end = models.DateTimeField(null=True)

    owner = models.ForeignKey(User, on_delete=models.CASCADE, related_name='owner')
    voters = models.ManyToManyField(User, through='Voter', related_name='voters')

    id_hashed = models.CharField(max_length=32, null=True)
```

Поля моделі Poll:

– title. CharField для збереження назви опитування;

– description. CharField для збереження опису опитування;

– votes_amt. PositiveSmallIntegerField для визначення кількості голосів, яку може дати кожен користувач;

– less_allowed. BooleanField, який вказує, чи можна обмежити кількість голосів користувача;

– show_while_running. BooleanField, який вказує, чи опитування видиме під час його проведення;

– date_created. DateTimeField, який автоматично встановлюється при створенні опитування;

– date_to_start та date_to_end. DateTimeField, які вказують на час початку та закінчення опитування;

– owner. ForeignKey для вказання власника опитування (зв'язок з моделлю User);

– voters. ManyToManyField для вказання користувачів, які взяли участь у голосуванні (зв'язок з моделлю User через Voter);

– id_hashed. CharField для збереження хешу, який є унікальним ідентифікатором опитування.

Метод post_create:

Цей метод генерує та зберігає унікальний ідентифікатор опитування (id_hashed), використовуючи дані про його створення. Метод викликається при події post_save для моделі Poll.

Модель Choice:

```
class Choice(models.Model):
    name = models.CharField(max_length=256)
    votes = models.IntegerField(default=0)
    poll = models.ForeignKey(Poll, on_delete=models.CASCADE)
```

Поля моделі Choice:

– name. CharField для збереження назви варіанта вибору;

– votes. IntegerField для збереження кількості голосів, отриманих варіантом вибору;

– poll. ForeignKey для вказання опитування, до якого належить варіант вибору;

Модель Voter:

```
class Voter(models.Model):
    poll = models.ForeignKey(Poll, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
```

Поля моделі Voter:

– poll. ForeignKey для вказання опитування, до якого належить виборець;

– user. ForeignKey для вказання користувача, який проголосував (зв'язок з моделлю User);

Сигнал post_save:

– З'єднання сигналу post_save з методом post_create для автоматичного виклику методу при збереженні нового екземпляра моделі Poll;

– Цей код визначає основну логіку додатку для опитувань, включаючи створення опитувань, варіантів вибору та виборців.

Визначаються серіалізатори для моделі Choice та інший серіалізатор для процесу голосування в Django Rest Framework. Давайте розглянемо кожен серіалізатор та їхню логіку:

Серіалізатор ChoiceSerializer:

```
class ChoiceSerializer(ModelSerializer):
    class Meta:
        model = Choice
```

```
        fields = 'all'
        read_only_fields = ['name', 'poll']
```

Основні характеристики:

– Використовується для серіалізації об'єктів моделі Choice;

– Усі поля моделі включаються (fields = 'all'), щоб автоматично включити всі поля моделі до серіалізації;

– Поля 'name' та 'poll' вказані як тільки для читання (read_only_fields), оскільки ці дані повинні лише читатися, а не змінюватися під час серіалізації.

Серіалізатор PollRunningChoiceSerializer:

```
class PollRunningChoiceSerializer(ModelSerializer):
    class Meta:
        model = Choice
        exclude = ['votes']
        read_only_fields = ['name', 'poll']
```

Основні характеристики:

– Використовується для серіалізації об'єктів моделі Choice, з виключенням поля 'votes';

– Цей серіалізатор призначений для використання тільки в тих випадках, коли опитування активне, тому дані про голоси відсутні.

Серіалізатор VotingSerializer:

```
class VotingSerializer(Serializer):
    votes = serializers.ListField()

    def init(self, instance=None, data=None, poll=None, user=None,
**kwargs):
        self.poll = poll
        self.user = user
        super().init(instance=instance, data=data, **kwargs)
```

```
    def validate(self, attrs):
        # Валідація вхідних даних перед створенням голосу

    def create(self, validated_data):
        # Логіка створення голосу
```

Основні характеристики:

– Використовується для обробки та валідації даних при створенні голосу;

– Містить поле votes, яке є списком обраних варіантів вибору.

Метод validate проводить валідацію вхідних даних:

– Перевіряє, чи обрано хоча б один варіант вибору;

– Перевіряє обмеження на кількість голосів, якщо таке встановлено для опитування;

– Перевіряє наявність обраних варіантів вибору в поточному опитуванні;

– Перевіряє, чи користувач ще не голосував у цьому опитуванні;

– Перевіряє, чи опитування активне, тобто час голосування.

Метод create створює логіку створення голосу:

– Збільшує кількість голосів для обраних варіантів вибору;

– Додає користувача до списку голосуючих;

– Зберігає зміни в базі даних.

Ці серіалізатори визначають способи перетворення об'єктів моделі та вхідних даних для використання їх у Django Rest Framework.

Визначає два декоратори для перевірки аутентифікації користувача в Django-додатку: user_authenticated та user_unauthenticated. Давайте розглянемо кожен декоратор та їхню логіку:

Декоратор user_authenticated:

```
def user_authenticated(view_func):
    def wrapper(request, *args, **kwargs):
        if request.user.is_authenticated:
            return view_func(request, *args, **kwargs)
        else:
            return redirect('index')
    return wrapper
```

Основні характеристики:

– Приймає функцію view_func (вид функції) як аргумент;

– Створює функцію-обгортку wrapper, яка перевіряє, чи користувач аутентифікований (request.user.is_authenticated);

– Якщо користувач аутентифікований, викликає функцію view_func;

– Якщо користувач не аутентифікований, перенаправляє його на сторінку 'index'.

Декоратор user_unauthenticated

```
def user_unauthenticated(view_func):
    def wrapper(request, *args, **kwargs):
        if request.user.is_authenticated:
            return redirect('user')
        else:
            return view_func(request, *args, **kwargs)
    return wrapper
```

Основні характеристики:

– Приймає функцію view_func (вид функції) як аргумент;

– Створює функцію-обгортку wrapper, яка перевіряє, чи користувач не аутентифікований (not request.user.is_authenticated);

– Якщо користувач не аутентифікований, викликає функцію view_func;

– Якщо користувач аутентифікований, перенаправляє його на сторінку 'user'.

Декоратори можуть бути використані для захисту конкретних відображень від неавторизованого доступу або для перенаправлення користувачів залежно від їхнього стану аутентифікації. Наприклад:

```
@user_authenticated
def some_protected_view(request):
    # Код відображення для аутентифікованих користувачів
    pass

@user_unauthenticated
def login_view(request):
    # Код відображення для неаутентифікованих користувачів
    Pass
```

У цьому випадку, якщо користувач спробує отримати доступ до some_protected_view і не буде аутентифікований, його буде перенаправлено на сторінку 'index'. Аналогічно, якщо користувач спробує отримати доступ до login_view і вже аутентифікований, його буде перенаправлено на сторінку 'user'.

Далі визначаються маршрути (URL patterns) для Django-додатку і вказує, які відображення (views) пов'язані з кожним URL. Давайте розглянемо кожний маршрут та його призначення:

Основні Маршрути:

1. Індексна Сторінка

```
path('', views.index, name='index'),
```

Цей маршрут вказує на відображення index (ймовірно, головної сторінки або стартової сторінки).

2. Створення користувача (UserView)

```
url(r'^home.*$', views.UserView.as_view(), name='user'),
```

Цей маршрут вказує на відображення UserView, яке може бути пов'язане з URL-ами у вигляді '/home', '/home123', і т.д.

3. API Маршрути

Отримання Даних Користувача (UserAPI):

```
path('api/user/', views.UserAPI.as_view(), name='user_data'),
```

Цей маршрут слугує для отримання даних користувача через API.

4. Створення Опитування (CreatePollAPI)

```
path('api/create_poll/',views.CreatePollAPI.as_view(),
name='create_poll'),
```

Цей маршрут дозволяє створювати нові опитування через API.

5. Деталі Опитування (PollAPI):

```
path('api/poll/<str:id_hashed>/',views.PollAPI.as_view(),
name='poll'),
```

Цей маршрут дозволяє отримувати деталі опитування через API, де <str:id_hashed> - унікальний хеш опитування.

6. Подання Голосу (SubmitVoteAPI):

```
path('api/vote/<str:id_hashed>/',     views.SubmitVoteAPI.as_view(),
name='vote'),
```

Цей маршрут дозволяє користувачам подавати голоси в опитування через API.

Опитування Користувача (PollReprAPI):

```
path('api/user/polls/',views.PollReprAPI.as_view(),
name='user_polls'),
```

Цей маршрут дозволяє отримувати список опитувань, пов'язаних із певним користувачем через API.

Ці URL маршрути визначають, як Django має обробляти HTTP-запити та які відображення відповідають за ці запити в вашому Django-додатку.

Далі визначаються абстрактні класи для валідації даних в Django-додатку, а також реалізації декількох конкретних валідаторів. Давайте розглянемо основні частини цього коду:

Клас Validator:

```
class Validator(ABC):
    fields = {}

    def init(self, data, *args, **kwargs):
        self.data = data
        self.validated_data = {}

    def validation(self, *args, **kwargs):
        pass

    def is_valid(self, *args, **kwargs):
        # Перевірка наявності всіх полів та їх типів
        for name, data_type in self.fields.items():
            if name in self.data.keys():
                try:
                    self.validated_data[name]                    =
data_type(self.data[name])
                except:
```

```
                        self.error    =    '"{name}"    is    not    type
{data_type}'.format(name=name, data_type=data_type)
                        return False
                else:
                    self.error      =      '"{name}"      does      not
exist!'.format(name=name)
                    return False

            # Власна валідація
            try:
                self.error = self.validation(*args, **kwargs)
            except:
                self.error = 'An error occurred, probably because the
submission is invalid.'
                return False

            if self.error:
                return False

            return True
```

Клас Validator є базовим абстрактним класом для валідаторів:

fields: Словник, що містить опис всіх полів та їх типів, які очікується отримати валідатор;

init: Метод ініціалізації, який приймає дані для валідації;

validation: Метод для реалізації індивідуальної валідації;

is_valid: Метод для виклику валідації та перевірки на валідність.

Клас Validation:

```
class Validation(ABC):
    def init(self, data, *args, **kwargs):
        self.data = data
```

```
    def validation(self, *args, **kwargs):
        pass

    def is_valid(self, *args, **kwargs):
        try:
            self.validated_data = self.validation(*args, **kwargs)
            return True
        except:
            return False
```

Клас Validation є абстрактним класом для реалізації індивідуальних правил валідації.

Клас ListValidation:

```
class ListValidation(Validation):
    def validation(self, *args, **kwargs):
        return None
```

Клас ListValidation успадкований від Validation та реалізує валідацію списків.

```
Клас PollValidator:
python
Copy code
class PollValidator(Validator):
    fields = {
        'title': str,
        'description': str,
        'votes_amt': int,
        'less_allowed': bool,
        'show_while_running': bool,
        'date_to_start': str,
        'date_to_end': str,
```

```
            'choices': ListValidation,
        }

        def validation(self, *args, **kwargs):
            data = self.data
            if not data.title.isspace():
                return 'Title must contain at least one non-whitespace
character!'
            if not data.description.isspace():
                return 'Description must contain at least one non-
whitespace character!'
            if int(data.votes_amt) < 1:
                return 'Votes amount must be higher than 0!'
```

Клас PollValidator успадкований від Validator та конкретизує очікувані поля та їхні типи для опитувань.

Реалізує індивідуальні правила валідації для опитувань, такі як перевірка непорожніх рядків, валідація кількості голосів і т.д.

Ці класи можуть бути використані для валідації вхідних даних у додатку, наприклад, під час створення опитувань або інших дій, де потрібна перевірка коректності та відповідності типам даних.

Визначаються відображення для додатку, яке взаємодіє з клієнтськими запитами та виконує деякі дії. Давайте розглянемо основні частини цього коду:

Функція index:

```
@user_unauthenticated
def index(request):
    return render(request, 'index.html', {})
```

index є функцією для відображення головної сторінки. Застосований декоратор @user_unauthenticated, який перенаправляє аутентифікованих користувачів на іншу сторінку (можливо, на сторінку користувача).

Клас UserView:

```
@method_decorator(user_authenticated, name='dispatch')
class UserView(View):
    def get(self, request, *args, **kwargs):
        return render(request, 'user.html', self.get_context_data(*args, **kwargs))

    def get_context_data(self, *args, **kwargs):
        return {
            'user': UserSerializer(self.request.user).data,
        }
```

UserView є класом, який успадкований від View. Для доступу до сторінки користувача. Використовує декоратор @user_authenticated для перевірки аутентифікації.

Клас UserAPI:

```
@method_decorator(user_authenticated, name='dispatch')
class UserAPI(APIView):
    def get(self, request, *args, **kwargs):
        return JsonResponse(UserSerializer(self.request.user).data)

    def post(self, request, *args, **kwargs):
        serializer = UserForm(request.POST, request.FILES, instance=self.request.user)
        if serializer.is_valid():
            user = serializer.save()
            return Response('Successfully changed user profile.', status=status.HTTP_200_OK)
        return JsonResponse(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

UserAPI - клас, що успадковує APIView для обробки запитів, пов'язаних із користувачем. Обробляє GET-запит для отримання даних користувача та POST-запит для зміни профілю користувача.

Клас CreatePollAPI

```
@method_decorator(user_authenticated, name='dispatch')
class CreatePollAPI(APIView):
    def post(self, request, *args, **kwargs):
        serializer = PollCreationSerializer(
            data=self.request.POST, user=request.user)

        if serializer.is_valid():
            poll = serializer.save()
            return Response('Poll successfully created.',
status=status.HTTP_200_OK)
        else:
            return JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
```

CreatePollAPI - клас для створення нового опитування через POST-запит.

Клас PollAPI:

```
@method_decorator(user_authenticated, name='dispatch')
class PollAPI(APIView):
    def dispatch(self, request, *args, **kwargs):
        self.poll =
Poll.objects.filter(id_hashed=kwargs['id_hashed'])
        if self.poll.exists():
            self.poll = self.poll[0]
        else:
            return Response('Poll not existing.',
status=status.HTTP_404_NOT_FOUND)
```

```
        return super().dispatch(request, *args, **kwargs)

    def get(self, request, *args, **kwargs):
        # Логіка отримання даних про опитування
    ...

    def post(self, request, *args, **kwargs):
        # Логіка оновлення даних опитування
    ...

    def delete(self, request, *args, **kwargs):
        # Логіка видалення опитування
    ...
```

PollAPI - клас для отримання, оновлення та видалення даних опитувань.

Це основні частини коду для обробки різних запитів, пов'язаних із користувачами та опитуваннями у додатку.

Клас SubmitVoteAPI:

```
@method_decorator(user_authenticated, name='dispatch')
class SubmitVoteAPI(APIView):
    def post(self, request, *args, **kwargs):
        user = request.user
        poll = Poll.objects.filter(id_hashed=kwargs['id_hashed'])

        # Перевірка існування опитування
        if poll.exists():
            poll = poll[0]
        else:
            return Response('Poll not existing.',
status=status.HTTP_404_NOT_FOUND)

        # Валідація голосу
```

```
            serializer = VotingSerializer(
                data=request.POST, poll=poll, user=request.user)

            if serializer.is_valid():
                serializer.save()
                return Response('Vote successfully submitted!',
status=status.HTTP_200_OK)
            else:
                return JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
```

SubmitVoteAPI - клас для обробки POST-запитів на голосування у відповідь на конкретне опитування.

Клас PollReprAPI:

```
@method_decorator(user_authenticated, name='dispatch')
class PollReprAPI(APIView):
    def get(self, request, *args, **kwargs):
        polls_created =
Poll.objects.filter(owner=self.request.user.id)
        polls_voted = self.request.user.voters.all()

        return JsonResponse({
            'polls_created': PollSerializer(polls_created,
many=True, context={
                'fields': {'title', 'id_hashed'}
            }).data,
            'polls_voted': PollSerializer(polls_voted, many=True,
context={
                'fields': {'title', 'id_hashed'}
            }).data,
        })
```

PollReprAPI - клас для отримання даних про опитування, які створив користувач або в яких він взяв участь.

Особливості коду:

SubmitVoteAPI відповідає за обробку голосів у відповідь на конкретне опитування. Голосування перевіряється за допомогою VotingSerializer, який містить власні правила валідації.

PollReprAPI відповідає за надання інформації про опитування, які були створені або в яких брав участь користувач. Інформація повертається у форматі JSON, де наводяться заголовки та хеш-ідентифікатори опитувань.

Ці класи розширюють функціональність додатку, дозволяючи користувачам створювати, переглядати, голосувати та видаляти опитування.

Файлом міграції Django, який визначає моделі даних для вашого додатку, а також створює таблиці у базі даних.

Опис структури міграції:

– Poll це модель, що представляє опитування. Містить різні поля, такі як title, description, votes_amt, less_allowed, show_while_running, date_created, date_to_start, date_to_end, і owner. Останнє поле owner вказує на користувача, який створив опитування, і здійснює зв'язок один до одного;

– Voter це модель, яка представляє відношення між користувачем і опитуванням, в якому він прийняв участь. Зберігає зовнішні ключі для вказання на poll (опитування) і user (користувач);

– Choice це модель, що представляє варіант відповіді для конкретного опитування. Зберігає інформацію про назву (name), кількість голосів (votes), і посилається на відповідне опитування (poll).

Ці міграції генеруються автоматично при використанні команди python manage.py makemigrations та python manage.py migrate для створення та застосування міграцій відповідно.

Цей код містить Django-views для обробки HTTP-запитів, пов'язаних із створенням, відображенням та зміною опитувань, а також профілю користувача.

index - Функція для обробки головної сторінки, яка відображається користувачам, які не увійшли в систему (user_unauthenticated).

UserView - Клас-відображення для обробки запитів, пов'язаних із відображенням профілю користувача.

UserAPI - Клас-відображення для надання та обробки інформації про користувача у форматі JSON.

CreatePollAPI - Клас-відображення для створення нового опитування на основі надісланих даних.

PollAPI - Клас-відображення для опитувань. Використовується для отримання, оновлення та видалення конкретного опитування.

Далі відбувається обробка HTTP-запитів, пов'язаних із відображенням інформації про опитування, які створив користувач, та відправленням голосувань за конкретні опитування.

PollReprAPI: get метод який відображає інформацію про опитування, створені користувачем та в яких він взяв участь. Використовується JsonResponse для повернення відповіді у форматі JSON.

SubmitVoteAPI: post метод який обробляє HTTP POST-запити для відправлення голосів за опитування. Перевіряє, чи існує опитування з вказаним id_hashed, а потім використовує VotingSerializer для перевірки та збереження голосів.

Основні Компоненти:

1. API (Application Programming Interface):

– RESTful API. Розробка API з використанням Django REST Framework для обробки HTTP запитів від фронтенду;

– Маршрутизація. Визначення маршрутів для різних функцій, таких як створення опитувань, отримання відповідей, управління користувачами.

2. Логіка обробки даних:

– Обробка запитів. Реалізація логіки для обробки запитів від користувачів, включаючи валідацію даних;

– Управління опитуваннями. Логіка для створення, редагування, видалення та отримання опитувань;

– Обробка відповідей. Механізми для збору та обробки відповідей учасників.

3. Взаємодія з базою даних:

– ORM (Object-Relational Mapping). Використання Django ORM для взаємодії з базою даних PostgreSQL;

– Міграції. Управління структурою бази даних через міграції.

4. Автентифікація та Авторизація:

– Система автентифікації. Впровадження системи автентифікації, наприклад, з використанням Django allauth;

– JWT або сесійні токени. Забезпечення безпечного доступу до API.

5. Безпека:

– Захист від загроз. Реалізація заходів безпеки, таких як захист від SQL-ін'єкцій, XSS, CSRF;

– Шифрування. Застосування шифрування для захисту чутливих даних.

Розробка серверної частини інформаційної технології для організації колективних опитувань вимагає глибокого розуміння бізнес-логіки, безпеки та взаємодії з фронтендом. Використання Django та Django REST Framework дозволяє створити надійний, безпечний та ефективний серверний бекенд, який забезпечує всі необхідні функції для проведення колективних опитувань(рис.3.7).

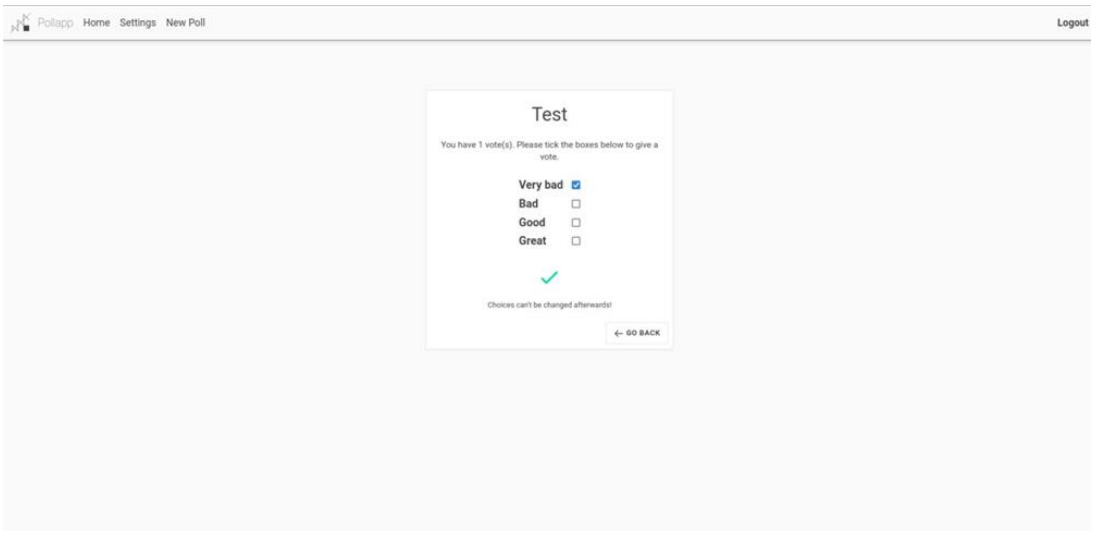


Рисунок 3.7 – Вибір варіанта відповіді

## 3.4 Розробка клієнтської частини веб-застосунку для колективних опитувань

Клієнтська частина (фронтенд) інформаційної технології для організації колективних опитувань є ключовим інтерфейсом для кінцевих користувачів. Вона повинна бути інтуїтивно зрозумілою, зручною у використанні та ефективно взаємодіяти з серверною частиною для забезпечення гладкого процесу проведення опитувань.

Для цієї частини використовується JavaScript та бібліотеку Axios та Vue Composition API для взаємодії із сервером. Основні функції цього коду - надання можливості робити HTTP-запити та спрощення процесу отримання даних і відправлення форм.

makeRequest:

– Ця функція призначена для виконання HTTP-запитів за допомогою бібліотеки Axios.

– Використовується метод axios для відправлення запиту з параметрами method, data та url.

– Додає заголовок X-CSRFToken для захисту від CSRF-атак, який витягується з куки csrftoken.

– xsrfCookieName та xsrfHeaderName вказують на ім'я кукі та заголовку для CSRF-токена.

useFetching:

– Ця функція використовує Vue Composition API для створення реактивних змінних та методів для взаємодії з сервером.

– notFound та fetching - реактивні змінні для відстеження статусу запиту (чи дані знайдені, чи триває процес запиту).

– fetchData - метод для відправлення GET-запиту за вказаним шляхом (path). Використовує makeRequest та обробляє помилки.

useSubmit:

– Ця функція також використовує Vue Composition API для створення реактивних змінних та методів для взаємодії з сервером.

– response, error, loading - реактивні змінні для відстеження результатів запиту (відповідь сервера, помилка, чи процес завантаження).

– submit - метод для відправлення POST-запиту за вказаним шляхом (path) та зазначеними даними. Використовує makeRequest та обробляє результати запиту.

Цей набір функцій використаний у Vue-компонентах для спрощення взаємодії з сервером та обробки запитів.

Далі визначаються стилі для веб-сторінки, написані з використанням препроцесора стилів Tailwind CSS та анімацій для Vue.js.

1. Імпорт шрифту:

```
@import
url('https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap');
```

Тут імпортується шрифт Roboto з Google Fonts.

2. Базові стилі та Tailwind CSS:

```
@tailwind base;
@tailwind components;
```

```
@tailwind utilities;
```

Застосовуються базові стилі та Tailwind CSS.

3. Глобальні стилі для елементів:

```
* {
    @apply text-font-dark font-body;
}
```

Всі елементи отримують основний текстовий колір та шрифт.

4. Стилі для інпутів та кнопок:

```
input, button {
    @apply bg-transparent focus:outline-none;
}
```

Визначається стиль для інпутів та кнопок.

5. Стилі для тіла сторінки:

```
body {
    @apply bg-backg-light;
}
```

Встановлюється фон тіла сторінки.

6. Стилі для елементу з ефектом тинту:

```
.tinted {
    backdrop-filter: saturate(180%) blur(10px);
}
```

Визначається ефект тинту для певного елементу.

7. Стилі для посилень та кнопок:

```
.link {
    @apply font-bold transition ease-out duration-200 text-primary-
500;
}

.btn, .btn-primary {
    // ...
}
```

Встановлюються стилі для посилень та кнопок.

8. Стилі для інших елементів:

```
.inpt, .tbox, .card, nav {
    // ...
}
```

Визначаються стилі для інших елементів, таких як поля вводу, блоки та навігаційна панель.

Анімації для Vue.js:

```
/* vue animations */
.appear-enter-active, .appear-leave-active {
    @apply transition ease-out duration-500;
}
// ...
```

Встановлюються анімації для входу/виходу та інші анімаційні класи Vue.js.

Ці стилі використовуються для вигляду та поведінки елементів на сторінці веб-застосунку.

Основні Компоненти:

1. Інтерфейс користувача:

- Структура сторінок: Розробка структури основних сторінок (головна сторінка, сторінка опитування, адміністративна панель тощо);

- Компоненти UI: Використання компонентів UI для створення зручного та сучасного інтерфейсу.

2. Взаємодія з сервером:

- Axios для HTTP запитів: Використання Axios для відправлення та отримання даних від сервера;

- Обробка відповідей сервера:** Відображення даних опитувань, обробка відповідей користувачів.

3. Маршрутизація та навігація:

- Vue Router: Використання Vue Router для управління навігацією всередині застосунку.

4. Стан застосунку:

- Vuex: Управління станом застосунку, зберігання даних користувачів, опитувань та відповідей.

5. Адаптивний дизайн:

- Responsive Web Design: Забезпечення коректного відображення на різних пристроях (десктопи, планшети, мобільні телефони).

Розробка:

1. Настройка проекту:

– Створення нового проекту Vue.js;

– Встановлення та налаштування необхідних залежностей та плагінів.

2. Розробка компонентів UI:

– Створення компонентів для різних частин інтерфейсу: навігаційна панель, форми опитувань, списки опитувань тощо.

3. Маршрутизація та навігація:

– Визначення маршрутів та налаштування Vue Router для управління переходами між сторінками.

4. Взаємодія з сервером:

– Реалізація логіки для відправлення запитів до сервера та обробки отриманих відповідей.

5. Управління станом:

– Використання Vuex для управління глобальним станом застосунку (наприклад, даними користувача, станом опитувань).

6. Тестування та оптимізація:

– Проведення тестування компонентів, маршрутизації та взаємодії з сервером;

– Оптимізація швидкодії та відповідності до стандартів веб-доступності.

Розробка клієнтської частини веб-застосунку для колективних опитувань вимагає зосередження на створенні інтуїтивно зрозумілого, зручного для користувача інтерфейсу, а також ефективної взаємодії з серверною частиною. Використання сучасних фреймворків та бібліотек, таких як Vue.js, Vuex, Vue Router, Axios, дозволяє створити динамічний та відгуковий фронтенд, який забезпечує високу продуктивність та зручність використання.

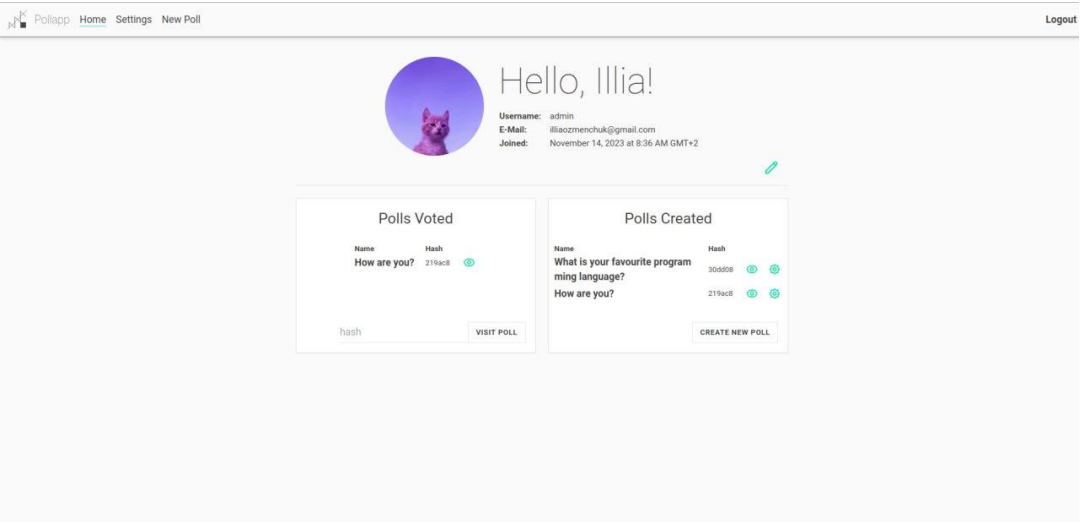


Рисунок 3.8 – Акаунт користувача

## 3.5 Тестування веб-застосунку для колективних опитувань

Тестування веб-застосунку для колективних опитувань є критично важливим етапом розробки, який забезпечує якість, надійність, безпеку та коректну функціональність продукту. Цей процес включає різні види тестування, від юніт-тестування до приймального тестування.

Види Тестування:

1. Юніт-тестування (Unit Testing):

– Мета Перевірка окремих модулів або компонентів застосунку на коректність роботи;

– Інструменти. Для Python/Django можна використовувати `unittest` або `pytest`; для JavaScript/Vue.js - `Jest` або `Mocha`.

2. Функціональне тестування (Functional Testing):

– Мета. Перевірка різних функцій веб-застосунку (наприклад, створення опитування, відправлення відповідей);

– Інструменти. Selenium, TestCafe для автоматизації браузерних тестів.

3. Інтеграційне тестування (Integration Testing):

– Мета. Перевірка взаємодії між різними компонентами системи (наприклад, фронтенд-бекенд інтеграція);

– Інструменти. Cypress, Postman.

4. Навантажувальне тестування (Load Testing):

– Мета. Перевірка здатності застосунку витримувати велике навантаження (багато користувачів, запитів);

– Інструменти. JMeter, LoadRunner.

5. Тестування безпеки (Security Testing):

– Мета: Виявлення потенційних вразливостей у застосунку (наприклад, SQL ін'єкції, XSS);

– Інструменти:** OWASP ZAP, Burp Suite.

6. Тестування користувацького інтерфейсу (UI Testing):**

– Мета: Перевірка елементів інтерфейсу на відповідність вимогам та зручність використання;

– Інструменти: Puppeteer, Selenium.

7. Приймальне тестування (Acceptance Testing):

– Мета: Оцінка застосунку кінцевими користувачами на відповідність їхнім очікуванням та потребам;

– Формат: Зазвичай виконується вручну.

Процес Тестування:

1. Планування:

– Визначення обсягу тестування, вибір методів та інструментів;

– Розробка тестових сценаріїв.

2. Реалізація тестів:

– Написання тестових скриптів та випробувань;

– Налаштування тестового середовища.

3. Виконання тестів:

– Проведення тестів, збір результатів;

– Аналіз результатів, виявлення помилок та проблем.

4. Звітність та вдосконалення:

– Складання звітів про результати тестування;

– Внесення правок та оптимізація коду.

Тестування є невід'ємною частиною процесу розробки веб-застосунку для колективних опитувань. Воно допомагає забезпечити високу якість продукту, виявити та виправити помилки на ранніх етапах, а також підвищити надійність та безпеку застосунку. Використання різноманітних видів тестування та інструментів дозволяє всебічно оцінити різні аспекти системи. Результат зображено на рисунку 3.9.

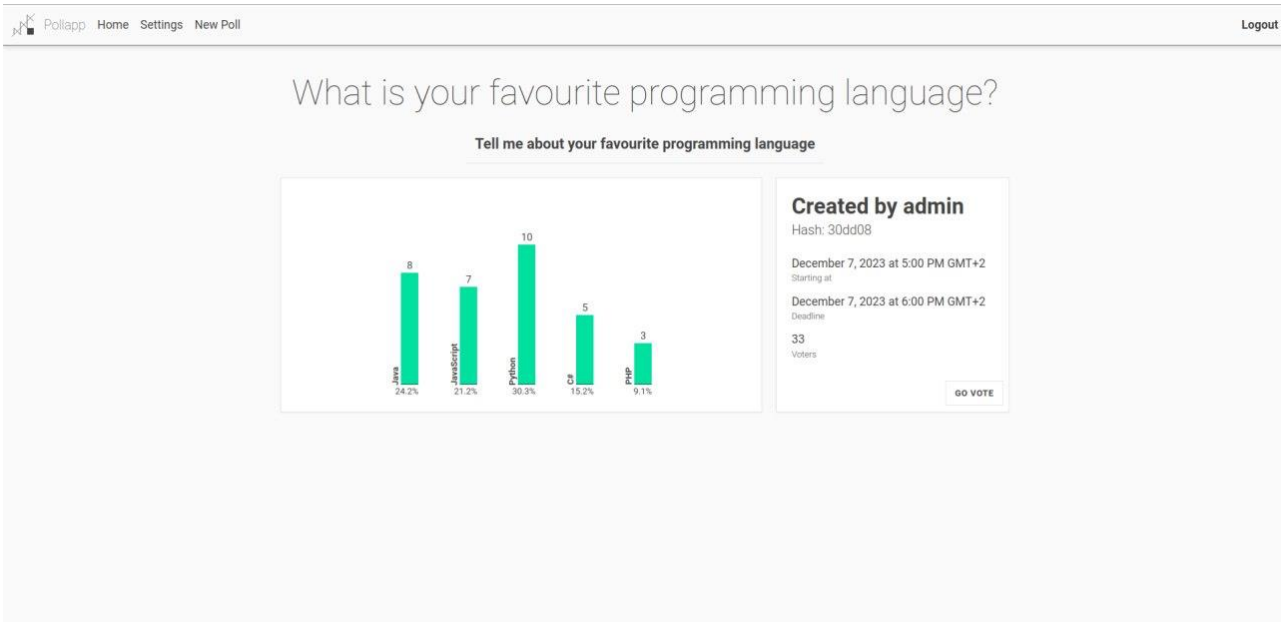


Рисунок 3.9 – Статистика опитувань

Розроблений додаток має функціонал, наведений в таблиці 3.1, який було порівняно із функціоналом популярних застосунків: Google Forms, Typeform, Qualtrics.

Таблиця 3.1 – Порівняльна характеристика розробленого ПЗ

| Назва програми | Вигляд форми | Аналітика | Вимоги безпеки | Додаткові функції |
|---|---|---|---|---|
| Google Forms | +/- | - | +/- | +/- |
| Typeform | + | - | - | + |
| Qualtrics | - | - | + | +/- |
| Розроблений додаток | + | + | + | + |

Отже, розроблений додаток для було протестовано на відповідність меті дослідження. Функціонал програмного забезпечення є розширеним по відношенню до існуючих сучасних рішень та відповідає завданню.

### 3.6 Висновок до розділу 3

Розділ присвячений програмній реалізації інформаційної технології організації колективних опитувань, охоплює важливі аспекти розробки веб-застосунку, включаючи структуру бази даних, загальну структуру програмного забезпечення, розробку серверної та клієнтської частин, а також тестування застосунку. Кожен з цих етапів відіграє ключову роль у створенні ефективного, надійного та зручного у використанні інструменту для проведення колективних опитувань.

Структура бази даних забезпечує надійне зберігання та швидкий доступ до даних опитувань, відповідей та інформації про користувачів.

Розробка веб-застосунку включає інтеграцію серверної та клієнтської частин, що забезпечує гнучкість, масштабованість та високу продуктивність системи.

Використання Django та Django REST Framework для розробки серверної частини дозволяє створити надійний бекенд з потужними можливостями обробки та управління даними.

Розробка фронтенду на основі Vue.js, Vuex та інших сучасних технологій забезпечує інтуїтивно зрозумілий та зручний інтерфейс для кінцевих користувачів.

Ретельне тестування всіх компонентів системи, від юніт-тестів до приймального тестування, гарантує високу якість продукту, виявлення та виправлення помилок на ранніх стадіях розробки.

Розробка веб-застосунку для колективних опитувань має велике значення у сучасному цифровому світі. Це не тільки забезпечує ефективний інструмент для збору та аналізу думок широкої аудиторії, але й відкриває нові можливості для досліджень, маркетингу, соціальних опитувань та інших сфер, де важливо знати та розуміти громадську думку.

Проект має потенціал для подальшого розвитку та вдосконалення. Можливі напрямки включають інтеграцію з додатковими сервісами, розширення функціональності для специфічних потреб користувачів, підвищення безпеки та приватності даних, а також оптимізація для мобільних пристроїв.

У цілому, розробка веб-застосунку для колективних опитувань є складним, але водночас надзвичайно важливим процесом. Він вимагає глибокого розуміння потреб користувачів, високого рівня технічної експертизи та уваги до деталей на кожному етапі розробки. Завдяки цьому процесу створюється потужний інструмент, який може бути використаний

# 4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження «Інформаційна технологія організації колективних опитувань» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

## 4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія організації колективних опитувань» є

оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [19].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

| Бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено працездатність продукту в реальних |
| Ринкові переваги (недоліки) | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші, | Технічні та споживчі влас-тивості продукту трохи гірші, ніж в | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в | Технічні та споживчі властивості продукту значно кращі, |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної | Ринок малий, але має позитивну | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |

Продовження таблиці 4.1

| Практична здійсненність | | | | | |
|---|---|---|---|---|---|
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовуються у військово промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві |
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту |

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

| Критерії | Експерт (ПІБ, посада) | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| | Бали: | | |
| 1. Технічна здійсненність концепції | 5 | 4 | 5 |
| 2. Ринкові переваги (наявність аналогів) | 3 | 4 | 3 |
| 3. Ринкові переваги (ціна продукту) | 4 | 4 | 3 |
| 4. Ринкові переваги (технічні властивості) | 3 | 3 | 4 |
| 5. Ринкові переваги (експлуатаційні витрати) | 3 | 3 | 3 |
| 6. Ринкові перспективи (розмір ринку) | 4 | 3 | 3 |
| 7. Ринкові перспективи (конкуренція) | 3 | 2 | 2 |
| 8. Практична здійсненність (наявність фахівців) | 5 | 5 | 5 |
| 9. Практична здійсненність (наявність фінансів) | 2 | 3 | 2 |
| 10. Практична здійсненність (необхідність нових матеріалів) | 4 | 5 | 5 |
| 11. Практична здійсненність (термін реалізації) | 5 | 4 | 5 |
| 12. Практична здійсненність (розробка документів) | 5 | 5 | 4 |
| Сума балів | 46 | 45 | 44 |
| Середньоарифметична сума балів $СБ_с$ | 45,0 | | |

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [19].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

| Середньоарифметична сума балів СБ, розрахована на основі висновків експертів | Науково-технічний рівень та комерційний потенціал розробки |
|---|---|
| 41…48 | Високий |
| 31…40 | Вище середнього |
| 21…30 | Середній |
| 11…20 | Нижче середнього |
| 0…10 | Низький |

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія організації колективних опитувань» становить 45,0 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість

проведення даних досліджень (рівень комерційного потенціалу розробки високий).

### 4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [20]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

$\alpha_i$ – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт $\alpha_i$ визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

$\beta_i$ – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення $\beta_i$ для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{аi}} \,, \tag{4.2}$$

де $I_{нi}$ та $I_{на}$ – чисельні значення конкретного *і*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \quad (4.3)$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

| Показники (параметри) | Одиниця вимірю-вання | Аналог | Проектований пристрій | Відношення параметрів нової розробки до аналога | Питома вага показника |
|---|---|---|---|---|---|
| Швидкість запуску | с | 7 | 1,5 | 4,66 | 0,35 |
| Швидкість доступу до даних | мкс | 210 | 140 | 1,5 | 0,3 |
| Кількість запитів за секунду (qps) | qps | 70 | 231 | 3,3 | 0,1 |
| Місце на диску | MB | 890 | 240 | 3,7 | 0,1 |
| Потреба в оперативній пам’яті | GB | 8 | 4 | 2 | 0,15 |

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 4{,}66 \cdot 0{,}35 + 1{,}5 \cdot 0{,}3 + 3{,}3 \cdot 0{,}1 + 3{,}7 \cdot 0{,}1 + 2 \cdot 0{,}15 = 3{,}08.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,08 рази.

## 4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія організації колективних опитувань», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

### 4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [19]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де $k$ – кількість посад дослідників залучених до процесу досліджень;

$M_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

$t_i$ – число днів роботи конкретного дослідника, дн.;

$T_p$ – середнє число робочих днів в місяці, $T_p$=23 дні.

$$З_о = 17500{,}00 \cdot 46 / 23 = 35000{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці 4.5.

Таблиця 4.5 – Витрати на заробітну плату дослідників

| Найменування посади | Місячний посадовий оклад, грн | Оплата за робочий день, грн | Число днів роботи | Витрати на заробітну плату, грн |
|---|---|---|---|---|
| Керівник проекту | 17500,00 | 760,87 | 46 | 35000,00 |
| Інженер-програміст | 16500,00 | 717,39 | 40 | 28695,65 |
| Консультант (аналітик-соціолог) | 17300,00 | 752,17 | 7 | 5265,22 |
| Технік | 8500,00 | 369,57 | 44 | 16260,87 |
| Всього | | | | 85221,74 |

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія організації колективних опитувань» відсутні.

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

*$З_{дод}$ = (85221,74 + 0,00) · 12 / 100% = 10226,61 грн.*

### 4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

*Зн = (85221,74 + 0,00 + 10226,61) · 22 / 100% = 20998,64 грн.*

### 4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія організації колективних опитувань».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} , \qquad (4.7)$$

де $H_j$ – норма витрат матеріалу *j*-го найменування, кг;

*n* – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$B_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$M_1$ = *2,0 · 210,00 · 1,1 - 0 · 0 = 462,00 грн.*

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на матеріали

| Найменування матеріалу, марка, тип, сорт | Ціна за од, грн | Норма витрат, од | Величина відходів, од | Ціна відходів, грн/од | Вартість витраченого матеріалу, грн |
|---|---|---|---|---|---|
| Папір офісний А4 500/80 | 210,00 | 2,0 | 0 | 0 | 462,00 |
| Приладдя канцелярське BAMBOO КХ-200 | 240,00 | 3,0 | 0 | 0 | 792,00 |
| Картридж принтера EPSON JT-500 | 1820,00 | 1,0 | 0 | 0 | 2002,00 |
| Всього | | | | | 3256,00 |

### 4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія організації колективних опитувань» відсутні.

### 4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.8)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

$K_i$ – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i$ = *1,10…1,12);*

$k$ – кількість найменувань устаткування.

$B_{спец}$ = *35699,00 · 1 · 1,1 = 39268,90 грн.*

Отримані результати зведемо до таблиці 4.7:

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

| Найменування устаткування | Кількість, шт | Ціна за одиницю, грн | Вартість, грн |
|---|---|---|---|
| Стаціонарний комп’ютер на базі Ryzen 5 3600 у якості виділеного сервера (dedicated server) | 1 | 35699,00 | 39268,90 |
| Система мережевого обладнання передачі даних | 1 | 25800,00 | 28380,00 |
| Всього | | | 67648,90 |

### 4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.9)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$K_i$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i$ = *1,10…1,12);*

*k* – кількість найменувань програмних засобів.

$B_{прг} = 8400{,}00 \cdot 1 \cdot 1{,}1 = 9240{,}00$ *грн.*

Отримані результати зведемо до таблиці 4.8:

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

| Найменування програмного засобу | Кількість, шт | Ціна за одиницю, грн | Вартість, грн |
|---|---|---|---|
| Прикладне середовище розробки ПЗ VSCode | 1 | 8400,00 | 9240,00 |
| Програмне забезпечення SQLite server | 1 | 9300,00 | 10230,00 |
| Програмне забезпечення Vite server | 1 | 8650,00 | 9515,00 |
| Всього | | | 28985,00 |

### 4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.10)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (8560{,}00 \cdot 2) / (5 \cdot 12) = 285{,}33$ *грн.*

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

| Найменування обладнання | Балансова вартість, грн | Строк корисного використання, років | Термін використання обладнання, місяців | Амортизаційні відрахування, грн |
|---|---|---|---|---|
| Офісне обладнання робочого місця керівника проекту | 8560,00 | 5 | 2 | 285,33 |
| Робоче місце інженера - розробника Ноутбук Lenovo ThinkPad T480 | 23899,00 | 3 | 2 | 1327,72 |
| Пристрій виводу інформації EPSON Jl | 6520,00 | 4 | 2 | 271,67 |
| Тестовий термінал вводу запиту | 6800,00 | 3 | 2 | 377,78 |
| Лабораторія розробки ПЗ | 456000,00 | 25 | 2 | 3040,00 |
| ОС Windows 11 | 5700,00 | 3 | 2 | 316,67 |

Продовження таблиці 4.9

| | | | | |
|---|---|---|---|---|
| Прикладний пакет Microsoft Office 2019 | 5120,00 | 3 | 2 | 284,44 |
| Обладнання офісної оргтехніки | 9600,00 | 5 | 2 | 320,00 |
| Всього | | | | 6223,61 |

### 4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_e$) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.11)$$

де $W_{yi}$ – встановлена потужність обладнання на визначеному етапі розробки, кВт;

$t_i$ – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

$\eta_i$ – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,10 \cdot 350,0 \cdot 7,50 \cdot 0,95 / 0,97 = 262,50$ *грн.*

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Витрати на електроенергію

| Найменування обладнання | Встановлена потужність, кВт | Тривалість роботи, год | Сума, грн |
|---|---|---|---|
| Офісне обладнання робочого місця керівника проекту | 0,10 | 350,0 | 262,50 |
| Робоче місце інженера - розробника Ноутбук Lenovo ThinkPad T480 | 0,08 | 320,0 | 192,00 |
| Пристрій виводу інформації EPSON Jl | 0,11 | 10,0 | 8,25 |
| Тестовий термінал вводу запиту | 0,07 | 25,0 | 13,13 |

Продовження таблиці 4.10

| | | | |
|---|---|---|---|
| Стаціонарний комп'ютер на базі Ryzen 5 3600 у якості виділеного сервера (dedicated server) | 0,32 | 320,0 | 768,00 |
| Система мережевого обладнання передачі даних | 0,20 | 320,0 | 480,00 |
| Обладнання офісної оргтехніки | 0,45 | 2,5 | 8,44 |
| Всього | | | 1732,31 |

### 4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія організації колективних опитувань» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (4.12)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

*$B_{св}$ = (85221,74 + 0,00) · 20 / 100% = 17044,35 грн.*

### 4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.13)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

*$B_{сп}$ = (85221,74 + 0,00) · 30 / 100% = 25566,52 грн.*

### 4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.14)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

*$І_в$ = (85221,74 + 0,00) · 50 / 100% = 42610,87 грн.*

### 4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків;

витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (4.15)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

*$В_{нзв}$ = (85221,74 + 0,00) · 100 / 100% = 85221,74 грн.*

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія організації колективних опитувань» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}\ . \qquad (4.16)$$

*$В_{заг}$= 85221,74 +0,00 +10226,61 +20998,64 +3256,00 +0,00 +67648,90 + 28985,00 + 6223,61 +1732,31 +17044,35 +25566,52 +42610,87 +85221,74 = 394736,29 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}\ , \qquad (4.17)$$

де $\eta$ - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо $\eta$=0,9.

*ЗВ = 394736,29 / 0,9 = 438595,87 грн.*

## 4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів

тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія організації колективних опитувань» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

$\Delta N$ – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

| Показник | 1-й рік | 2-й рік | 3-й рік | 4-й рік |
|---|---|---|---|---|
| Збільшення кількості споживачів, осіб | 12000 | 15000 | 16000 | 10000 |

$N$ – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 165000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 40,00 грн;

$\pm\Delta Ц_o$ – зміна вартості послуги від впровадження результатів, приймемо 13,24 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [19]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.18)$$

де $\lambda$ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

$\rho$ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho$ =41%;

$\vartheta$ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta$ =18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(13,24·165000,00+53,24·12000)·0,83·0,41·(1-0,18/100%)=787880,80 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(13,24·165000,00+53,24·27000)·0,83·0,41·(1-0,18/100%)=1010726,94 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(13,24·165000,00+53,24·43000)·0,83·0,41·(1-0,18/100%)=1248429,48 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(13,24·165000,00+53,24·53000)·0,83·0,41·(1-0,18/100%)=1396993,57 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.19)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

$T$ – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau$ =0,15;

$t$ – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =787880,80/(1+0,15)[1]+1010726,94/(1+0,15)[2]+1248429,48/(1+0,15)[3]+
+1396993,57/(1+0,15)[4]=685113,74+764254,77+820862,65+798735,61=3068966,77 грн.

Величина початкових інвестицій $PV$, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.20}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 438595,87 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 438595,87 = 877191,75 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.21}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 3068966,77 грн;

$PV$ – теперішня вартість початкових інвестицій, 877191,75 грн.

$E_{абс} = ПП - PV$ = 3068966,77 - 877191,75 = 2191775,02 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.22}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 2191775,02 грн;

$PV$ – теперішня вартість початкових інвестицій, 877191,75 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+2191775,02/877191,75)^{1/4} = 0,37.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.23}$$

де $d$ – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні $d$ =0,12;

$f$ – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,2.

$\tau_{мін}$ = 0,12+0,2 = 0,32 < 0,37 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія організації колективних опитувань» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{4.24}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,37 = 2,72 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

**Висновок до розділу 4**

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія організації колективних опитувань» становить 45,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,08 рази.

Також термін окупності становить 2,72 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія організації колективних опитувань».

# ВИСНОВКИ

При виконанні магістерської кваліфікаційної роботи розв’язано задачу розробки інформаційної технології організації колективних опитувань.

Здійснений аналіз предметної області дозволив отримати глибше розуміння вимог та функціональності систем для колективних опитувань. Вивчення програм-аналогів визначило їхні переваги та недоліки, що вказало на потребу розробки нового рішення, адаптованого до конкретних вимог та цілей.

Розроблена структура інформаційної технології визначає основні компоненти та зв'язки між ними. Вона враховує вимоги користувачів і дозволяє ефективно взаємодіяти з системою для проведення та участі в колективних опитуваннях.

Розроблена база даних оптимізована для зберігання та взаємодії з інформацією про опитування, користувачів та їхні відповіді. Архітектура серверної частини підтримує ефективний обмін даними між клієнтами та сервером.

Програмна реалізація системи для колективних опитувань враховує функціональні вимоги та забезпечує їх ефективне виконання. Використані технології та інструменти відповідають вибору мови програмування та фреймворків.

Проведене тестування підтверджує функціональність та надійність програмного додатку. Виявлені і виправлені помилки дозволяють забезпечити стабільну роботу системи в реальних умовах використання.

Виконані економічні розрахунки визначають вартість розробки та впровадження системи. Було проведено дослідження, в якому розробка інформаційної технології для організації колективних опитувань має високий комерційний потенціал, оцінений на рівні 45,0 бала. Технічно ця розробка переважає існуючі аналоги приблизно в 3,08 рази за узагальненим коефіцієнтом якості розробки. Термін окупності становить 2,72 р., що свідчить про її

комерційну привабливість і можливість залучення інвестицій для впровадження та виведення на ринок.

Всі завдання магістерської кваліфікаційної роботи виконані. Даний комплексний підхід дозволив використовувати розроблений додаток для організації та проведення колективних опитувань в різних областях.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна технологія організації колективних опитувань / І.С. Озменчук, В.В. Колодний, – "Молодь в науці: дослідження, проблеми, перспективи (МН-2024)" [Електронний ресурс]. Режим доступу – https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/19798/16387

2. Колодний В. В. Некритеріальне оцінювання альтернатив/ В. В. Колодний, В. В. Зубко // “ІНТЕРНЕТОСВІТА-НАУКА-2016”: Збірник матеріальів конференції . Вінниця : ВНТУ, 2016. - С. 43-44.

3. L. Bianchi, L. M. Gambardella et M. Dorigo, An ant colony optimization approach to the probabilistic traveling salesman problem, PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Springer Verlag, Berlin, Allemagne, 2002.

4. Data visualization beginner's guide: a definition, examples, and learning resources [Електронний ресурс] – Режим доступу до ресурсу: https://www.tableau.com/learn/articles/data-visualization.

5. The Top 10 Types of Data Visualization Made Simple [Електронний ресурс] – Режим доступу до ресурсу: https://boostlabs.com/10-types-of-data-visualization-tools/#The-15-Most-Common-Types-of-Data-Visualization-Formats.

6. Юхимчук С.В., Колодний В.В., Зарезенко Д.П. Програмна реалізація системи підтримки прийняття рішень, що базується на методі дискретно-неперервного аналізу розподілу корисності. - Вісник ВПІ. – 2009. – №2. – С. 70-77.

7. Колодний В.В. Комп'ютерна програма «Система візуалізації колективної експертизи CollExpert5» / В.В. Колодний, С.М. Мельник, А.А. Яцько // Свідоцтво про реєстрацію авторського права на твір №96492. – Рішення від 03.03.2020.

8. Berson A. Client/server architecture / Alex Berson. – New York, NY: McGraw-Hill, Inc.Professional Book Group, 1992. – 452 с.

9. S. Iredi, D. Merkle et M. Middendorf, Bi-Criterion Optimization with Multi Colony Ant Algorithms, Evolutionary Multi-Criterion Optimization, First International Conference (EMO'01), Zurich, Springer Verlag, pages 359 — 372, 2001.

10. Колодний В. В. Мобільний застосунок візуального порівняння альтернатив VisPA / В. В. Колодний, П. О. Зоря // "Інформаційні технології та комп'ютерна інженерія". Вінниця: ВНТУ, 2019. - С.3

11. Розробка інтелектуальних систем підтримки прийнятих рішень для діагностики, керування та оптимізації технічних та біологічних обїєктів: звіт про НДР (заключний) / Наук. кер. В.Д. Дмитрієнко. – Х.:НТУ «ХПІ», 2012

12. PHP (Hypertext Preprocessor) Definition [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://techterms.com/definition/php.

13. Solutions M. Advantages and Disadvantages of PHP Frameworks [Електронний ресурс] / Mindfire Solutions. – 2018. – Режим доступу до ресурсу: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-php-frameworks-c046d50754e5.

14. TechTarget Contributor. What is Ruby? - Definition from WhatIs.com [Електронний ресурс] / TechTarget Contributor. – 2019. – Режим доступу до ресурсу: https://whatis.techtarget.com/definition/Ruby.

15. Nick Gorbikoff. What are the advantages and disadvantages of Ruby? [Електронний ресурс] / Nick Gorbikoff. – 2018. – Режим доступу до ресурсу: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-Ruby.

16. What is Python? Python - Overview. [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/Python/Python_overview.htm.

17. An Introduction to Python [Електронний ресурс] – Режим доступу до ресурсу: https://Python.info/intro.

18. Колодний В. В. Інформаційна технологія виявлення переважань в неструктурованих задачах прийняття рішень / В. В. Колодний, В. В. Зубко // "Інформаційні технології та комп'ютерна інженерія". Вінниця: ВНТУ, 2015. - С.3.

19. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

20. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

# ДОДАТКИ

## Додаток А (обов'язковий)

## Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія організації колективних опитувань

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність 86,97% Схожість 13,03%

**Аналіз звіту подібності** (відмітити потрібне):

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

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

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

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

Автор роботи _____ Озменчук І.С.

Керівник роботи _____ Колодний В.В.

**Опис прийнятого рішення**

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

## Додаток Б (обов'язковий)

## Лістинг серверної частини додатку

```
from django.db import models
from django.db.models.signals import post_save
from django.core.exceptions import ValidationError

# Create your models here.

from accounts.models import User
from django.utils import timezone

class Poll(models.Model):
    title = models.CharField(max_length=128)
    description = models.CharField(max_length=1024)
    votes_amt = models.PositiveSmallIntegerField(default=1)
    less_allowed = models.BooleanField(default=False)
    show_while_running = models.BooleanField(default=False)

    date_created = models.DateTimeField(auto_now=True)
    date_to_start = models.DateTimeField(null=True)
    date_to_end = models.DateTimeField(null=True)

    owner = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='owner')
    voters = models.ManyToManyField(User, through='Voter',
related_name='voters')

    id_hashed = models.CharField(max_length=32, null=True)

    @classmethod
    def post_create(cls, sender, instance, created, *args, **kwargs):
        if not created:
            return
```

```
        # Creating poll identification "hash"
        # Made out of creation year, month and poll id % 999999 (I don't
think more than 1,000,000 polls will be created per month...)
        # Then everything will be encoded to hexadecimal

        id_hashed = hex(int('{id}{year}{month}'.format(
            year=instance.date_created.year,
            month=instance.date_created.month,
            id=instance.id % 999999,
        )))[2:]

        instance.id_hashed = id_hashed
        instance.save()

post_save.connect(Poll.post_create, sender=Poll)


class Choice(models.Model):
    name = models.CharField(max_length=256)
    votes = models.IntegerField(default=0)
    poll = models.ForeignKey(Poll, on_delete=models.CASCADE)


class Voter(models.Model):
    poll = models.ForeignKey(Poll, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

from rest_framework.serializers import ModelSerializer, Serializer
from rest_framework import serializers

from django.utils import timezone
from django.forms import ModelForm
```

```
from accounts.models import User
from .models import Poll, Choice


class UserSerializer(ModelSerializer):
    class Meta:
        model = User
        fields = ['username', 'first_name',
                  'last_name', 'email', 'date_joined', 'profile_pic']
        read_only_fields = ['date_joined']
        extra_kwargs = {'profile_pic': {'required': False}}


class UserForm(ModelForm):
    class Meta:
        model = User
        fields = ['username', 'first_name', 'last_name',
                  'email', 'profile_pic']


class PollSerializer(ModelSerializer):
    class Meta:
        model = Poll
        fields = 'all'
        read_only_fields      =      ['votes_amt',      'less_allowed',
'show_while_running',
                            'date_created',             'date_to_start',
'date_to_end', 'owner', 'voters', 'id_hashed']


class PollCreationSerializer(ModelSerializer):

    choices = serializers.ListField()
```

```
    class Meta:
        model = Poll
        fields = 'all'
        read_only_fields    =    ['date_created',    'owner',    'voters',
'id_hashed']
        extra_kwargs = {
            "title":  {"error_messages":  {"blank":  "Title  must  not  be
empty!"}},
            "description": {"error_messages": {"blank": "Description must
not be empty!"}},
            "date_to_start": {"error_messages": {"invalid": "Starting date
must not be empty!"}},
            "date_to_end":  {"error_messages":  {"invalid":  "Ending  date
must not be empty!"}},
        }

    def init(self, instance=None, data=None, user=None, **kwargs):
        self.user = user
        self.fields['choices'].error_messages.update(
            {"required": "There must be at least two choices!"})
        super().init(instance=instance, data=data, **kwargs)

    def validate(self, attrs):
        if attrs['title'].isspace():
            raise serializers.ValidationError(
                'Title   must   contain   at   least   one   none-whitespace
character!')
        if attrs['description'].isspace():
            raise serializers.ValidationError(
                'Description must contain at least one none-whitespace
character!')
        if attrs['votes_amt'] < 1:
            raise serializers.ValidationError(
                'Votes amount must be higher than 0!')
```

```
        if attrs['date_to_start'] >= attrs['date_to_end']:
            raise serializers.ValidationError(
                'Ending date must occur after starting date!')
        if attrs['less_allowed'] and attrs['votes_amt'] == 1:
            raise serializers.ValidationError(
                'Less votes are only allowed if votes amount is higher
than one!')
        if len(attrs['choices']) < 2:
            raise serializers.ValidationError(
                'There must be at least two choices!')
        if attrs['votes_amt'] > len(attrs['choices']):
            raise serializers.ValidationError(
                'Votes amount per user must not be higher than the amount
of choices!')

        # check if date_to_start and date_to_end is not in the past
        return super().validate(attrs)

    def create(self, validated_data):
        poll = Poll(
            title=validated_data['title'],
            description=validated_data['description'],
            votes_amt=validated_data['votes_amt'],
            less_allowed=validated_data['less_allowed'],
            show_while_running=validated_data['show_while_running'],
            date_to_start=validated_data['date_to_start'],
            date_to_end=validated_data['date_to_end'],
            owner=self.user,
        )

        poll.save()

        for item in self.validated_data['choices']:
            choice = Choice(name=item, poll=poll)
```

```
            choice.save()

        return poll

    def update(self, instance, validated_data):
        return super().update(instance, validated_data)

class ChoiceSerializer(ModelSerializer):
    class Meta:
        model = Choice
        fields = 'all'
        read_only_fields = ['name', 'poll']


class PollRunningChoiceSerializer(ModelSerializer):
    class Meta:
        model = Choice
        exclude = ['votes']
        read_only_fields = ['name', 'poll']


class VotingSerializer(Serializer):
    votes = serializers.ListField()

    def  init(self,  instance=None,  data=None,  poll=None,  user=None,
**kwargs):
        self.poll = poll
        self.user = user
        super().init(instance=instance, data=data, **kwargs)

    def validate(self, attrs):
        if len(attrs['votes']) == 0:
            raise serializers.ValidationError(
                'You must at least select one choice!')
```

```
        if self.poll.less_allowed:
            if len(attrs['votes']) > self.poll.votes_amt:
                raise serializers.ValidationError(
                    'More     than     {amt}     choices     are     not
allowed!'.format(amt=self.poll.votes_amt))
        elif not len(attrs['votes']) == self.poll.votes_amt:
            raise serializers.ValidationError(
                'You             must             select             {amt}
choice(s)!'.format(amt=self.poll.votes_amt))

        for vote in attrs['votes']:
            choice = self.poll.choice_set.filter(id=int(vote))
            if not choice.exists():
                raise serializers.ValidationError(
                    'Choices don\'t refer to this poll!')

        if self.poll.voters.filter(id=self.user.id).exists():
            raise serializers.ValidationError(
                'You already voted for this poll.')

        if self.poll.date_to_start > timezone.now():
            raise serializers.ValidationError('Poll has not started yet!')

        if self.poll.date_to_end < timezone.now():
            raise serializers.ValidationError('Deadline is over!')

        return super().validate(attrs)

    def create(self, validated_data):

        for vote in validated_data['votes']:
            choice = Choice.objects.get(id=vote)
            choice.votes += 1
```

```
            choice.save()

        self.poll.voters.add(self.user)
        self.poll.save()

        return self.poll
```

Мельник Іван, [19.12.2023 22:57]

```
from django.shortcuts import render, redirect
from django.contrib import messages

def user_authenticated(view_func):
    def wrapper(request, *args, **kwargs):
        if request.user.is_authenticated:
            return view_func(request, *args, **kwargs)
        else:
            return redirect('index')
    return wrapper

def user_unauthenticated(view_func):
    def wrapper(request, *args, **kwargs):
        if request.user.is_authenticated:
            return redirect('user')
        else:
            return view_func(request, *args, **kwargs)
    return wrapper

from django.urls import path, include
from django.conf.urls import url

from . import views

urlpatterns = [
    path('', views.index, name='index'),
```

```
    url(r'^home.*$', views.UserView.as_view(), name='user'),

    # apis
    path('api/user/', views.UserAPI.as_view(), name='user_data'),
    path('api/create_poll/',               views.CreatePollAPI.as_view(),
name='create_poll'),
    path('api/poll/<str:id_hashed>/',            views.PollAPI.as_view(),
name='poll'),
    path('api/vote/<str:id_hashed>/',      views.SubmitVoteAPI.as_view(),
name='vote'),
    path('api/user/polls/',                  views.PollReprAPI.as_view(),
name='user_polls'),
]

import json

from rest_framework.serializers import ListField

class Validator(ABC):

    fields = {} # All fields in format 'name': [type] e.g. 'title': str

    def init(self, data, *args, **kwargs):
        self.data = data
        self.validated_data = {}

    def validation(self, *args, **kwargs):
        ''' Individual validation here '''
        pass

    def is_valid(self, *args, **kwargs):
        print(self.data['choices'])

        # validate if data contains all fields
```

```
        for name, data_type in self.fields.items():
            if name in self.data.keys():
                # validate if data types are correct
                try:
                    self.validated_data[name]                                =
data_type(self.data[name])
                except:
                    self.error     =     '"{name}"     is     not     type
{data_type}'.format(name=name, data_type=data_type)
                    return False

            else:
                self.error = '"{name}" does not exist!'.format(name=name)
                return False

        # validate validation
        try:
            self.error = self.validation(self, *args, **kwargs)
        except:
            self.error = 'An error occured, probably because the submission
is invalid.'
            return False
        if error:
            return False

        return True

class Validation(ABC):
    def init(self, data, *args, **kwargs):
        self.data = data

    def validation(self, *args, **kwargs):
        ''' Individual validation here '''
        pass
```

```
    def is_valid(self, *args, **kwargs):
        try:
            self.validated_data = self.validation(self, *args, **kwargs)
            return True
        except:
            return False

class ListValidation(Validation):
    def validation(self, *args, **kwargs):
        return None



##### Custom validators #####

class PollValidator(Validator):

    fields = {
        'title': str,
        'description': str,
        'votes_amt': int,
        'less_allowed': bool,
        'show_while_running': bool,
        'date_to_start': str,
        'date_to_end': str,
        'choices': ListValidation,
    }

    def validation(self, *args, **kwargs):
        data = self.data
        if not data.title.isspace():
            return 'Title must contain at least one none-whitespace
character!'
```

```
        if not data.description.isspace():
            return 'Description must contain at least one none-whitespace
character!'
        if int(data.votes_amt) < 1:
            return 'Votes amount must be higher than 0!'


from django.shortcuts import render, redirect
from django.utils.decorators import method_decorator
from django.views import View
from django.http import JsonResponse
from django.utils import timezone
from rest_framework import status
from rest_framework.response import Response
from rest_framework.views import APIView

# Create your views here.

from .decorators import user_authenticated, user_unauthenticated
from     .serializers     import     UserSerializer,     PollSerializer,
PollCreationSerializer,  ChoiceSerializer,  PollRunningChoiceSerializer,
VotingSerializer, UserForm
from .validators import PollValidator
from .models import Poll, Choice



@user_unauthenticated
def index(request):
    return render(request, 'index.html', {})



@method_decorator(user_authenticated, name='dispatch')
class UserView(View):
    def get(self, request, *args, **kwargs):
```

```
        return render(request, 'user.html', self.get_context_data(*args,
**kwargs))

    def get_context_data(self, *args, **kwargs):
        return {
            'user': UserSerializer(self.request.user).data,
        }


@method_decorator(user_authenticated, name='dispatch')
class UserAPI(APIView):
    def get(self, request, *args, **kwargs):
        return JsonResponse(UserSerializer(self.request.user).data)

    def post(self, request, *args, **kwargs):
        serializer      =      UserForm(request.POST,      request.FILES,
instance=self.request.user)
        if serializer.is_valid():
            user = serializer.save()
            return   Response('Successfully   changed   user   profile.',
status=status.HTTP_200_OK)
        return                              JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


@method_decorator(user_authenticated, name='dispatch')
class CreatePollAPI(APIView):
    def post(self, request, *args, **kwargs):

        serializer = PollCreationSerializer(
            data=self.request.POST, user=request.user)

        if serializer.is_valid():
            poll = serializer.save()
```

```
            return      Response('Poll      successfully      created.',
status=status.HTTP_200_OK)
        else:
            return                          JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


@method_decorator(user_authenticated, name='dispatch')
class PollAPI(APIView):

    def dispatch(self, request, *args, **kwargs):
        self.poll = Poll.objects.filter(id_hashed=kwargs['id_hashed'])
        if self.poll.exists():
            self.poll = self.poll[0]
        else:
            return         Response('Poll         not         existing.',
status=status.HTTP_404_NOT_FOUND)
        return super().dispatch(request, *args, **kwargs)

    def get(self, request, *args, **kwargs):
        choices = None
        if          self.poll.show_while_running          or          (not
self.poll.show_while_running and self.poll.date_to_end <= timezone.now()):
            print(True)
            choices = ChoiceSerializer(
                self.poll.choice_set.all(), many=True).data
        else:
            choices = PollRunningChoiceSerializer(
                self.poll.choice_set.all(), many=True).data

        return JsonResponse({
            'poll': PollSerializer(self.poll).data,
            'choices': choices,
            'poll_owner': UserSerializer(self.poll.owner).data,
```

```
            'user_is_owner': self.poll.owner.id == request.user.id,
        })


    def post(self, request, *args, **kwargs):
        if not self.poll.owner.id == self.request.user.id:
            return Response('You are not permitted to change this poll!',
status=status.HTTP_401_UNAUTHORIZED)

        serializer = PollSerializer(self.poll, data=request.POST)

        if serializer.is_valid():
            serializer.save()
            return      Response('Poll      successfully      changed!',
status=status.HTTP_200_OK)
        else:
            return                          JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request, *args, **kwargs):
        if not self.poll.owner.id == self.request.user.id:
            return Response('You are not permitted to delete this poll!',
status=status.HTTP_401_UNAUTHORIZED)

        self.poll.delete()

return Response('Poll successfully deleted!', status=status.HTTP_200_OK)


@method_decorator(user_authenticated, name='dispatch')
class PollReprAPI(APIView):
    def get(self, request, *args, **kwargs):
        polls_created = Poll.objects.filter(owner=self.request.user.id)
        polls_voted = self.request.user.voters.all()
```

```
        return JsonResponse({
            'polls_created':   PollSerializer(polls_created,   many=True,
context={
                'fields': {'title', 'id_hashed'}
            }).data,
            'polls_voted':     PollSerializer(polls_voted,     many=True,
context={
                'fields': {'title', 'id_hashed'}
            }).data,
        })


@method_decorator(user_authenticated, name='dispatch')
class SubmitVoteAPI(APIView):
    def post(self, request, *args, **kwargs):
        user = request.user
        poll = Poll.objects.filter(id_hashed=kwargs['id_hashed'])

        # check if poll exists
        if poll.exists():
            poll = poll[0]
        else:
            return         Response('Poll         not         existing.',
status=status.HTTP_404_NOT_FOUND)

        # validate vote
        serializer = VotingSerializer(
            data=request.POST, poll=poll, user=request.user)

        if serializer.is_valid():
            serializer.save()
            return      Response('Vote      successfully      submitted!',
status=status.HTTP_200_OK)
```

```
        else:
            return                          JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

# Generated by Django 3.1.3 on 2021-04-17 15:57

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Poll',
            fields=[
                ('id',                models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('title', models.CharField(max_length=256)),
                ('description', models.CharField(max_length=1024)),
                ('votes_amt',
models.PositiveSmallIntegerField(default=1)),
                ('less_allowed', models.BooleanField(default=False)),
                ('show_while_running',
models.BooleanField(default=False)),
                ('date_created', models.DateTimeField(auto_now=True)),
                ('date_to_start', models.DateTimeField(null=True)),
```

```
                ('date_to_end', models.DateTimeField(null=True)),
                ('owner',
models.OneToOneField(on_delete=django.db.models.deletion.CASCADE,
related_name='owner', to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Voter',
            fields=[
                ('id',                models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('poll',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='user.poll')),
                ('user',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.AddField(
            model_name='poll',
            name='voters',
            field=models.ManyToManyField(related_name='voters',
through='user.Voter', to=settings.AUTH_USER_MODEL),
        ),
        migrations.CreateModel(
            name='Choice',
            fields=[
                ('id',                models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=256)),
                ('votes', models.IntegerField(default=0)),
```

```
                ('poll',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='user.poll')),
            ],
        ),
    ]

from django.shortcuts import render, redirect
from django.utils.decorators import method_decorator
from django.views import View
from django.http import JsonResponse
from django.utils import timezone
from rest_framework import status
from rest_framework.response import Response
from rest_framework.views import APIView

# Create your views here.

from .decorators import user_authenticated, user_unauthenticated
from     .serializers     import     UserSerializer,     PollSerializer,
PollCreationSerializer,  ChoiceSerializer,  PollRunningChoiceSerializer,
VotingSerializer, UserForm
from .validators import PollValidator
from .models import Poll, Choice


@user_unauthenticated
def index(request):
    return render(request, 'index.html', {})


@method_decorator(user_authenticated, name='dispatch')
class UserView(View):
    def get(self, request, *args, **kwargs):
```

```
        return render(request, 'user.html', self.get_context_data(*args,
**kwargs))

    def get_context_data(self, *args, **kwargs):
        return {
            'user': UserSerializer(self.request.user).data,
        }


@method_decorator(user_authenticated, name='dispatch')
class UserAPI(APIView):
    def get(self, request, *args, **kwargs):
        return JsonResponse(UserSerializer(self.request.user).data)

    def post(self, request, *args, **kwargs):
        serializer      =      UserForm(request.POST,      request.FILES,
instance=self.request.user)
        if serializer.is_valid():
            user = serializer.save()
            return   Response('Successfully   changed   user   profile.',
status=status.HTTP_200_OK)
        return                              JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


@method_decorator(user_authenticated, name='dispatch')
class CreatePollAPI(APIView):
    def post(self, request, *args, **kwargs):

        serializer = PollCreationSerializer(
            data=self.request.POST, user=request.user)

        if serializer.is_valid():
            poll = serializer.save()
```

```
            return      Response('Poll      successfully      created.',
status=status.HTTP_200_OK)
        else:
            return                          JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


@method_decorator(user_authenticated, name='dispatch')
class PollAPI(APIView):

    def dispatch(self, request, *args, **kwargs):
        self.poll = Poll.objects.filter(id_hashed=kwargs['id_hashed'])
        if self.poll.exists():
            self.poll = self.poll[0]
        else:
            return         Response('Poll         not         existing.',
status=status.HTTP_404_NOT_FOUND)
        return super().dispatch(request, *args, **kwargs)

    def get(self, request, *args, **kwargs):
        choices = None
        if          self.poll.show_while_running          or          (not
self.poll.show_while_running and self.poll.date_to_end <= timezone.now()):
            print(True)
            choices = ChoiceSerializer(
                self.poll.choice_set.all(), many=True).data
        else:
            choices = PollRunningChoiceSerializer(
                self.poll.choice_set.all(), many=True).data

        return JsonResponse({
            'poll': PollSerializer(self.poll).data,
            'choices': choices,
            'poll_owner': UserSerializer(self.poll.owner).data,
```

```
            'user_is_owner': self.poll.owner.id == request.user.id,
        })


    def post(self, request, *args, **kwargs):
        if not self.poll.owner.id == self.request.user.id:
            return Response('You are not permitted to change this poll!',
status=status.HTTP_401_UNAUTHORIZED)

        serializer = PollSerializer(self.poll, data=request.POST)

        if serializer.is_valid():
            serializer.save()
            return      Response('Poll      successfully      changed!',
status=status.HTTP_200_OK)
        else:
            return                          JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request, *args, **kwargs):
        if not self.poll.owner.id == self.request.user.id:
            return Response('You are not permitted to delete this poll!',
status=status.HTTP_401_UNAUTHORIZED)

        self.poll.delete()

return Response('Poll successfully deleted!', status=status.HTTP_200_OK)

@method_decorator(user_authenticated, name='dispatch')
class PollReprAPI(APIView):
    def get(self, request, *args, **kwargs):
        polls_created = Poll.objects.filter(owner=self.request.user.id)
        polls_voted = self.request.user.voters.all()
```

```
        return JsonResponse({
            'polls_created':   PollSerializer(polls_created,   many=True,
context={
                'fields': {'title', 'id_hashed'}
            }).data,
            'polls_voted':     PollSerializer(polls_voted,     many=True,
context={
                'fields': {'title', 'id_hashed'}
            }).data,
        })


@method_decorator(user_authenticated, name='dispatch')
class SubmitVoteAPI(APIView):
    def post(self, request, *args, **kwargs):
        user = request.user
        poll = Poll.objects.filter(id_hashed=kwargs['id_hashed'])

        # check if poll exists
        if poll.exists():
            poll = poll[0]
        else:
            return         Response('Poll         not         existing.',
status=status.HTTP_404_NOT_FOUND)

        # validate vote
        serializer = VotingSerializer(
            data=request.POST, poll=poll, user=request.user)

        if serializer.is_valid():
            serializer.save()
            return     Response('Vote     successfully     submitted!',
status=status.HTTP_200_OK)
        else:
```

```
            return                          JsonResponse(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
```

## Додаток В (обов'язковий)

## Лістинг клієнтської частини додатку

```
import axios from "axios"
import { ref } from "vue"

export function makeRequest(method, data, url) {
    return axios({
        method: method,
        url: url,
        headers: {
            'X-CSRFToken': 'csrftoken'
        },
        data: data,
        xsrfCookieName: 'csrftoken',
        xsrfHeaderName: 'X-CSRFToken'
    });
}

export function useFetching() {
    const notFound = ref(false);
    const fetching = ref(true);

    const fetchData = (path) => {
        notFound.value = false;
        fetching.value = true;

        return new Promise((resolve, reject) => {
            makeRequest("GET", "", path)
                .then((res) => {
                    resolve(res);
                })
                .catch((err) => {
                    notFound.value = true;
```

```
                    reject(err);
                })
                .finally(() => {
                    fetching.value = false;
                });
        });
    };

    return {
        notFound,
        fetching,
        fetchData,
    };
}

export function useSubmit() {
    const response = ref('');
    const error = ref('');
    const loading = ref(false);

    const submit = (path, data) => {
        loading.value = true;
        response.value = '';
        error.value = '';

        return new Promise((resolve, reject) => {
            makeRequest("POST", data, path)
                .then(res => {
                    response.value = res;
                    resolve(res);
                })
                .catch(err => {
                    error.value = err;
                    reject(err);
```

```
                })
                .finally(() => {
                    loading.value = false;
                });
        });
    };
    return {
        response,
        error,
        loading,
        submit,
    };
}

@import
url('https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,30
0;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swa
p');

@tailwind base;
@tailwind components;
@tailwind utilities;

* {
    @apply text-font-dark font-body;
}

input {
    @apply bg-transparent focus:outline-none;
}

button {
    @apply bg-transparent focus:outline-none;
}
```

```
body {
    @apply bg-backg-light;
}

.tinted {
    backdrop-filter: saturate(180%) blur(10px);
}

.link {
    @apply font-bold transition ease-out duration-200 text-primary-500;
}

.btn {
    @apply py-2 px-3 uppercase text-xs font-bold cursor-pointer tracking-
wider;
}

.btn-primary {
    @apply btn border transition ease-out duration-200 bg-transparent
hover:border-primary-400 hover:shadow;
}

.inpt {
    @apply border-b w-52 py-1 hover:bg-backg-light focus:bg-backg-light
focus:border-primary-400 transition ease-out duration-200;
}

.tbox {
    @apply border hover:bg-backg-light focus:bg-backg-light focus:border-
primary-400 transition ease-out duration-200;
}

.card {
```

```
    @apply flex flex-col border bg-white shadow-sm hover:shadow-md
transition ease-out duration-500;
}

nav {
    @apply sticky top-0;
}

/* vue animations */

.appear-enter-active,
.appear-leave-active {
    @apply transition ease-out duration-500;
}
.appear-enter-from,
.appear-leave-to {
    transform: scale(.75);
    @apply opacity-0;
}

.appear-enter-to,
.appear-leave-from {
    @apply opacity-100 scale-100;
}

.rise-enter-active,
.rise-leave-active {
    @apply transition ease-out duration-700;
}

.rise-enter-from,
.rise-leave-to {
    @apply transform scale-y-0 origin-bottom
}
```

```
.rise-enter-to,
.rise-leave-from {
    @apply transform scale-y-100 origin-bottom
}
```

**Додаток Г (обов'язковий)**

# ІЛЮСТРАТИВНА ЧАСТИНА

## ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ОПИТУВАНЬ

Виконав: студент 2-го курсу,
групи 3КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Озменчук І. С.
(прізвище та ініціали)

Керівник: к.т.н., доц. кафедри КН
_____ Колодний В.В.
(прізвище та ініціали)
«07» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Фрагменти алгоритму розробленого застосунку


Рисунок Г.1 – Графічне представлення фрагменту  алгоритму вибору параметрів  оптувань


Рисунок Г.2 – Графічне представлення фрагменту  алгоритму вибору опитувань


Рисунок Г.3 – Графічне представлення фрагменту  збору відповідей учасників


Рисунок Г.4 – Графічне представлення фрагменту  алгоритму вибору доступного опитування


Рисунок Г.5– Графічне представлення фрагменту  алгоритму вибору опитування для поширення


Рисунок Г.6– Графічне представлення фрагменту алгоритму вибору та визначення параметрів оптування


Рисунок Г.7– Графічне представлення фрагменту  алгоритму вибору опитування для поширення


Рисунок Г.8– Графічне представлення фрагменту  алгоритму вибору опитування для поширення

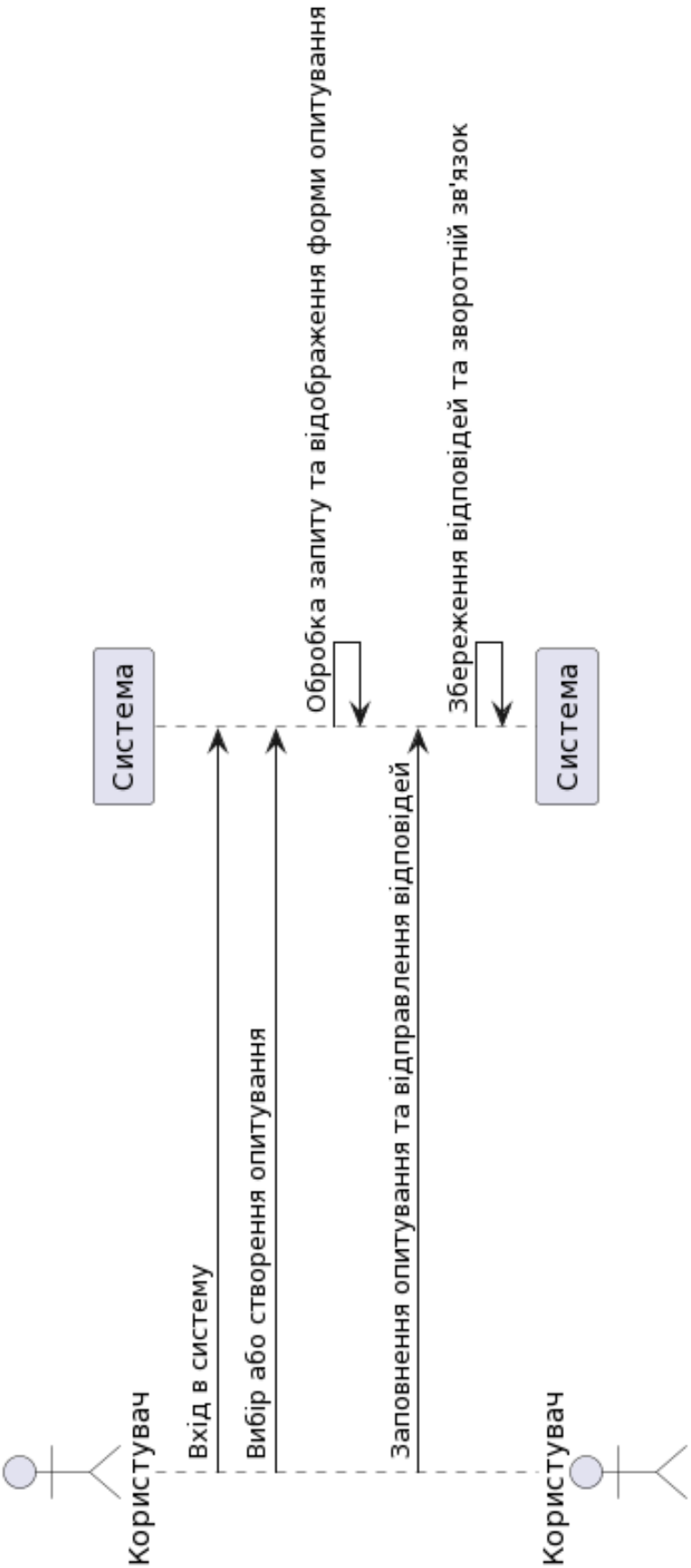


Рисунок Г.9 – UML -Діаграми процесу проведення опитувань

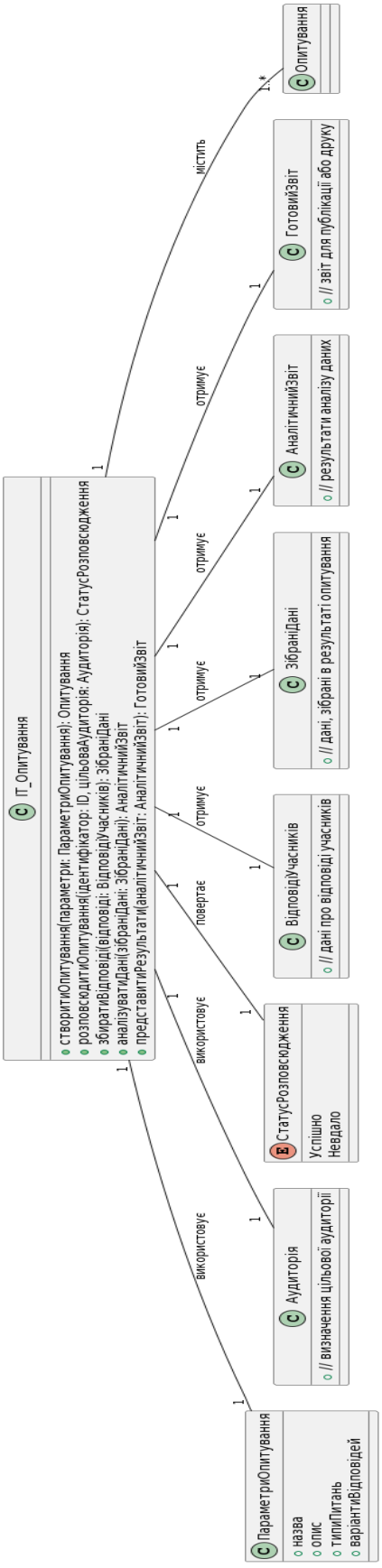


Рисунок Г.10 – UML -Діаграми взаємодії класових модулів системи

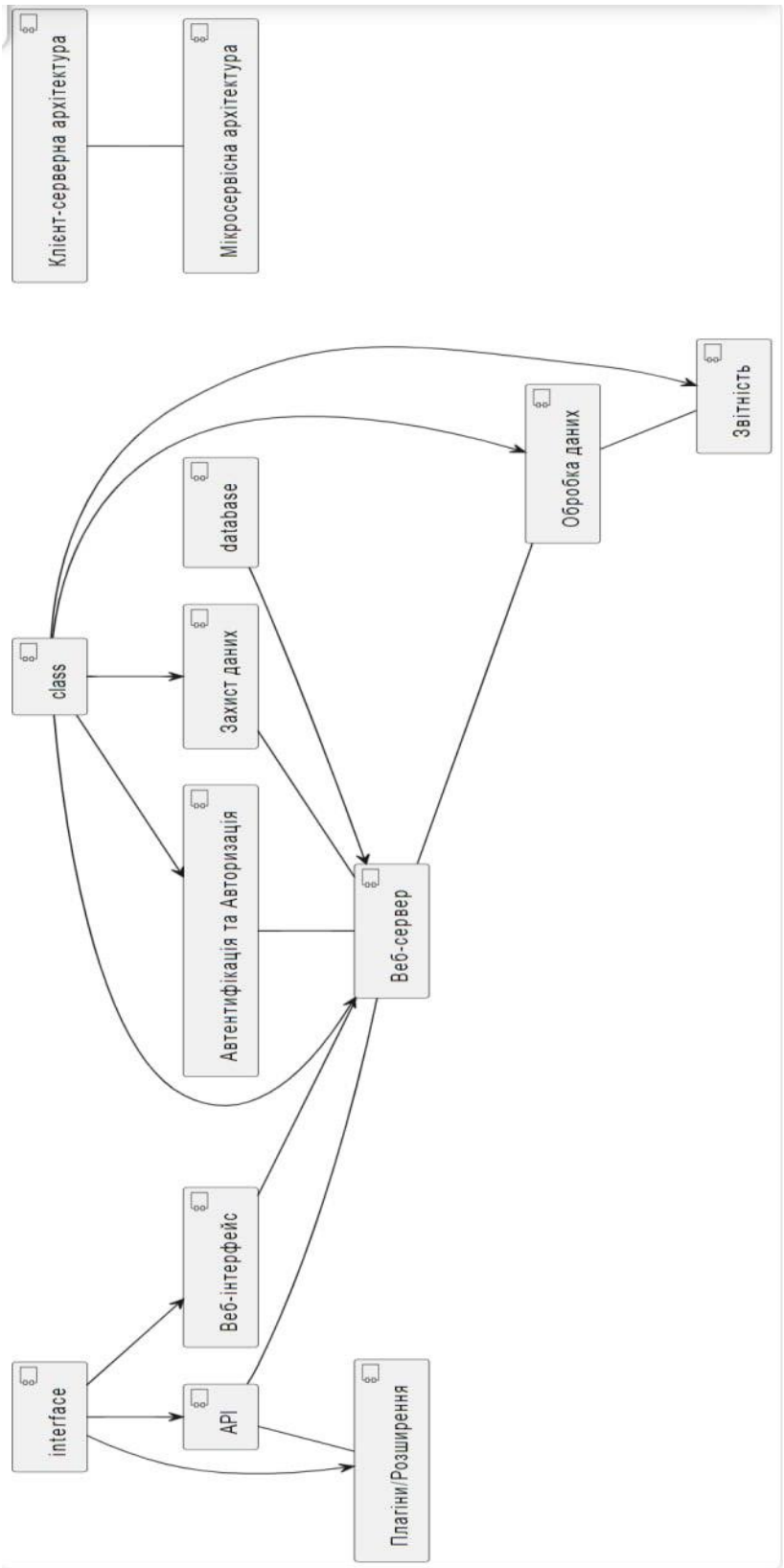


Рисунок Г.11 – Графічне приставляння роботи застосунку

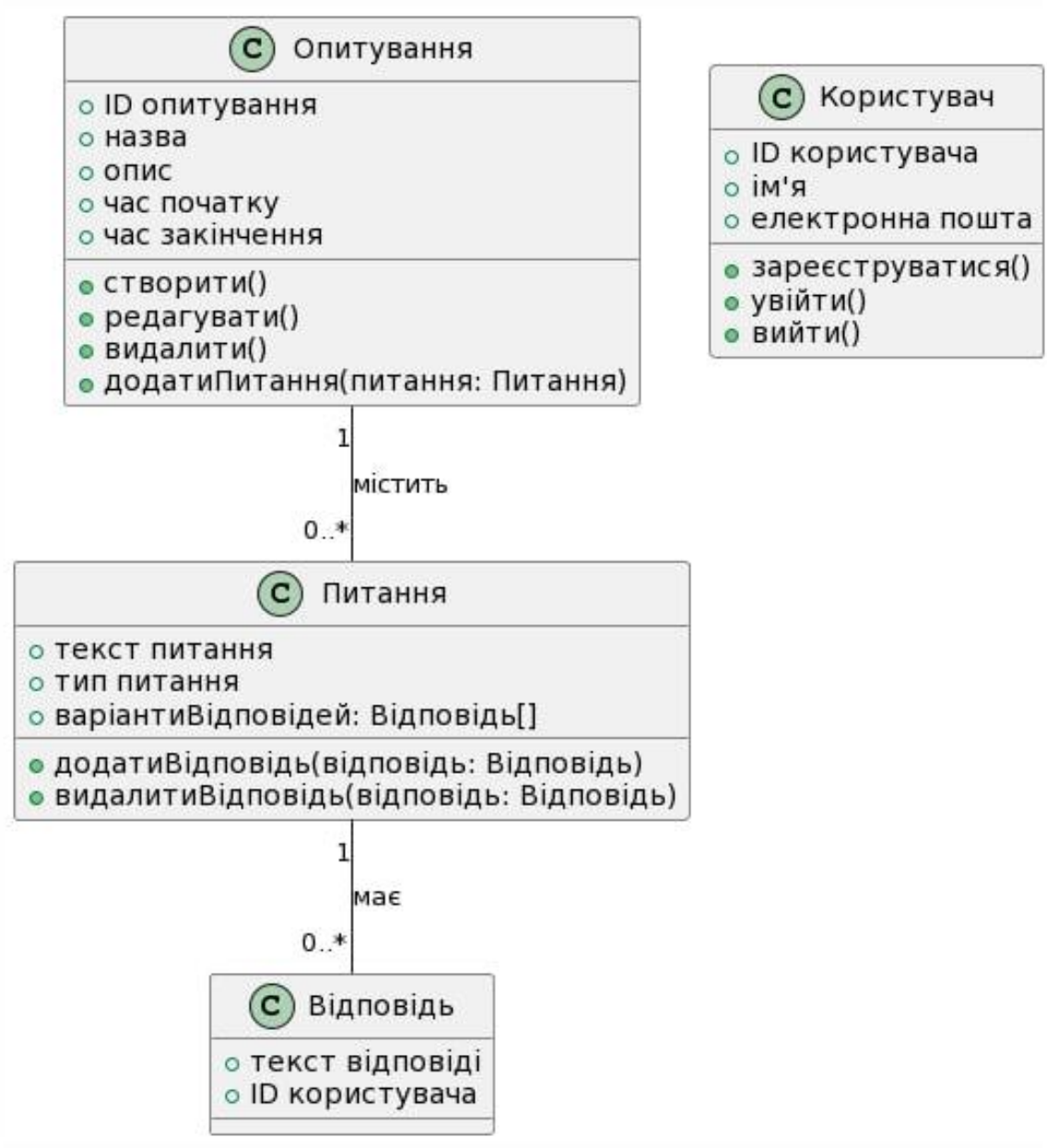


Рисунок Г.12 – Графічне представлення роботи

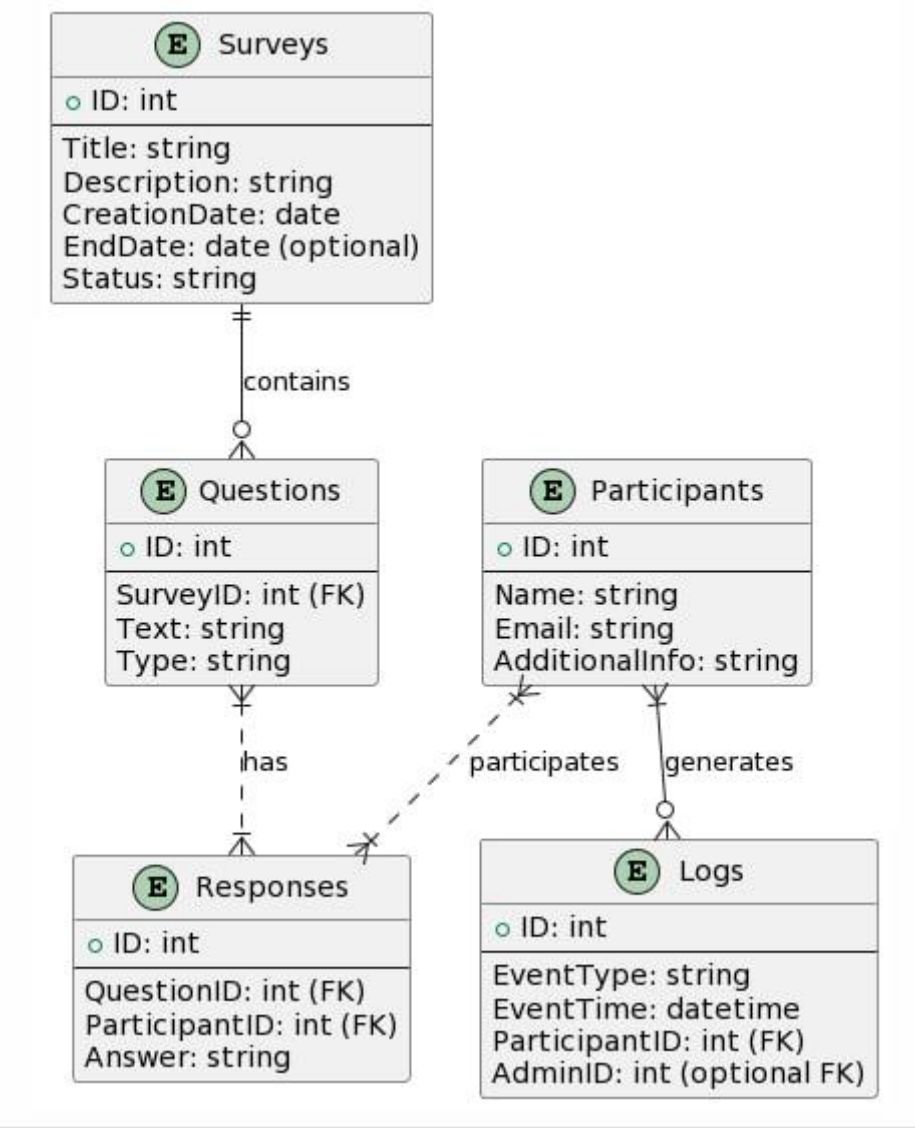


Рисунок Г.13 – Графічне представлення роботи модулів системи частина 1

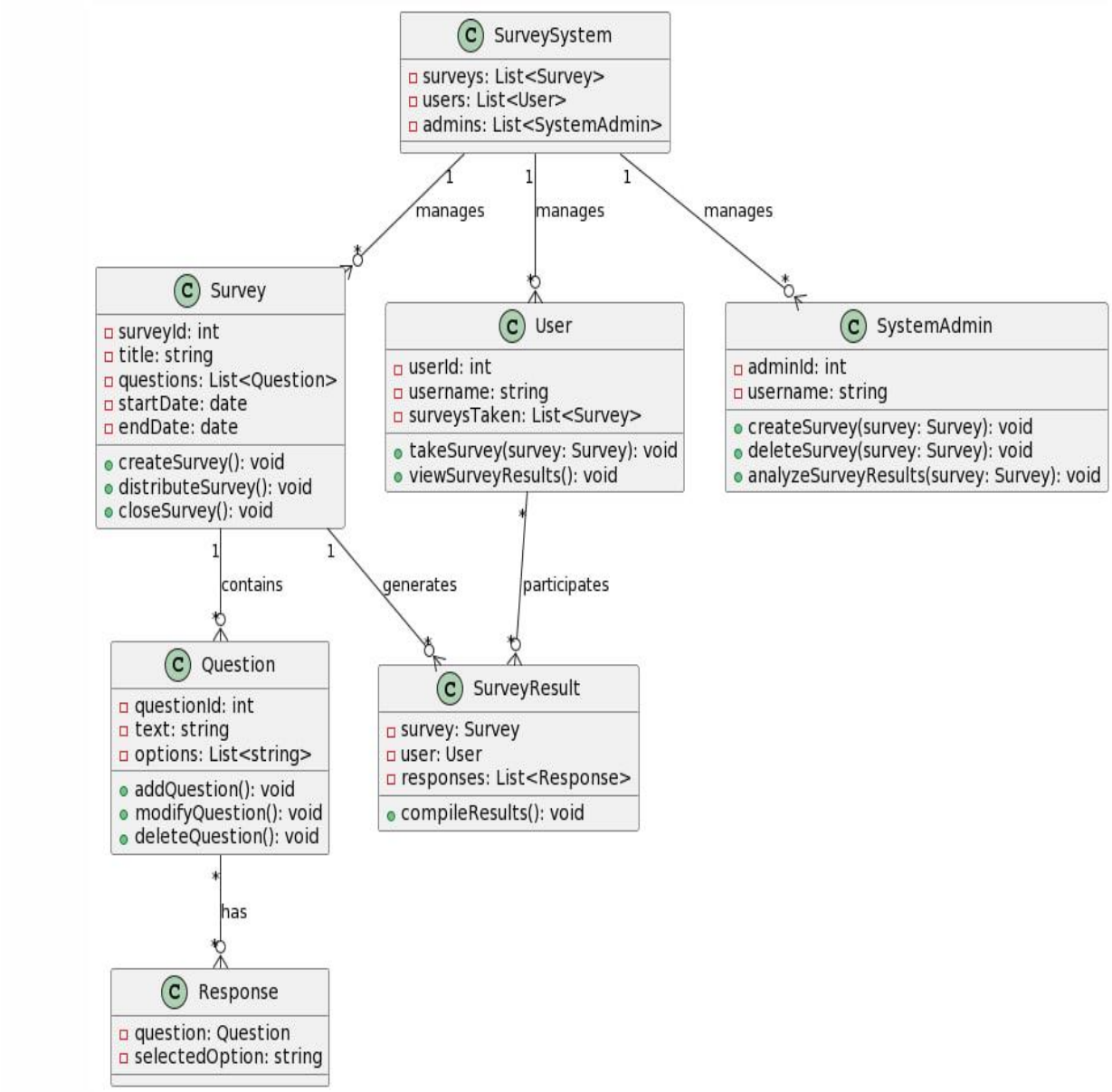


Рисунок Г.14 – Графічне представлення роботи модулів системи частина 2