ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка онлайн-платформи для організації соціологічних опитувань студентів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерел

_____ Микола ТИМОШЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-42
Микола ТИМОШЕНКО

Керівник: Ігор ГАМАНЮК
ст. викладач

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____Тимошенку Миколі Олексійовичу_____

1. Тема кваліфікаційної роботи: «Розробка онлайн-платформи для організації соціологічних опитувань студентів»
керівник кваліфікаційної роботи старший викладач кафедри ІПЗ Ігор ГАМАНЮК, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи проведення соціологічних опитувань, принципи формування питань і варіантів відповідей, документація до фреймворку Django, опис побудови клієнт-серверної архітектури веб-додатків, приклади реалізації онлайн-систем для збору, зберігання та обробки відповідей респондентів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз існуючих онлайн-платформ для проведення опитувань і методів збору соціологічних даних серед студентської аудиторії.
2. Проєктування веб-застосунку для створення, адміністрування та проходження опитувань з урахуванням вимог зручності та автоматизації.

3. Програмна реалізація та опис функціонування системи, зокрема: створення опитувань, додавання питань, збереження відповідей, відображення статистики, експорт результатів.
4. Тестування роботи платформи, перевірка її функціональності, стійкості до помилок і правильності обробки відповідей.
5. Реалізація базових механізмів захисту даних та ролей користувачів (доступ лише для адміністратора до керування опитуваннями й статистикою).

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма діяльності
6. Діаграма Станів
7. Діаграма предметної галузі
8. Екранні форми.
9. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Формування вимог до функціональності платформи	14.03-20.03.2025	
4	Проєктування структури даних та логіки взаємодії	21.03-28.03.2025	
5	Реалізація основних модулів: створення, проходження, таймер, обмеження	29.03-18.04.2025	
6	Тестування платформи та виправлення помилок	19.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Микола ТИМОШЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: _48_ стор., _3_ табл., _5_ рис., _17_ джерел.

Мета роботи – спрощення та оптимізація процесу збору та аналізу даних студентських опитувань

Об'єкт дослідження – процес організації та проведення опитувань серед студентів із використанням веб-сервісів для автоматизації збору даних

Предмет дослідження розробка програмного забезпечення для автоматизації збору, обробки та аналізу даних у рамках системи опитувань для організаторів освітнього процесу.

Короткий зміст роботи: У роботі розроблено веб-платформу для створення, проходження та аналізу соціологічних опитувань серед студентської аудиторії. Проаналізовано сучасні підходи до автоматизації опитувань, збору та обробки відповідей. В якості інструментів реалізації використано фреймворк Django для серверної частини та HTML/CSS/JavaScript — для формування інтерфейсу. Реалізовано функціонал таймера, обмеження повторного проходження за іменем, виведення детальної статистики й експорт результатів у форматах CSV та Excel. Проведено функціональне та модульне тестування системи.

Сферою використання застосунку є організація та автоматизація процесу соціологічного опитування студентів у закладах вищої освіти

КЛЮЧОВІ СЛОВА: ОПИТУВАННЯ, DJANGO, СОЦІОЛОГІЧНЕ ДОСЛІДЖЕННЯ, ТАЙМЕР,ВІДПОВІДІ КОРИСТУВАЧІВ

ЗМІСТ

ВСТУП 17
1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 19
1.1 Сучасні підходи до організації онлайн-опитувань 19
1.2 Аналіз функціоналу існуючих платформ 20
1.3 Обґрунтування потреби у створенні власної системи 23
1.4 Вимоги до систем автоматизованого опитування 24
1.5. Вибір технологій розробки 26
2 ПРОЄКТУВАННЯ СИСТЕМИ ОПИТУВАНЬ 29
2.1 Постановка завдання 29
2.2 Опис структури платформи та логіки взаємодії 30
2.3 Моделювання структури даних 31
2.4. Опис ролей користувачів та сценаріїв використання 32
2.6 Формування вимог до функціональності системи 36
2.7. Ризики проєктування та способи їх мінімізації 37
3 РЕАЛІЗАЦІЯ ТА ІНТЕГРАЦІЯ СИСТЕМИ 39
3.1. Вибір середовища розробки 39
3.2. Реалізація моделей і бази даних 41
3.3. Обробка форм, збереження результатів, таймер і перевірка за іменем 43
3.4. Створення та управління опитуваннями в інтерфейсі адміністратора 44
3.5. Експорт відповідей у CSV та Excel 46
3.6. Інтерфейс користувача: основні сторінки, шаблони 48
3.7. Адміністративна частина платформи 50
4. ТЕСТУВАННЯ ТА ОЦІНКА СИСТЕМИ 53
4.1. Опис методу тестування 53
4.2. Функціональне тестування основних можливостей 55
4.3. Приклади тестів 56
4.4. Аналіз результатів роботи системи 59
4.5. Порівняння з аналогами 60
ВИСНОВОК 63
ПЕРЕЛІК ПОСИЛАНЬ 65
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 68
ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ 76

ВСТУП

Сучасний освітній процес неможливо уявити без якісного зворотного зв’язку між студентами та викладачами, адміністрацією та учасниками освітнього середовища. Університети дедалі частіше звертаються до цифрових інструментів для організації внутрішнього моніторингу, опитувань щодо якості навчання, оцінки задоволеності студентів змістом освітніх програм, умовами навчання тощо. Онлайн-опитування стали зручним та гнучким інструментом, що дозволяє швидко отримувати статистично значущу інформацію в електронному форматі.

Однак більшість існуючих платформ для створення опитувань (наприклад, Google Forms, SurveyMonkey) мають низку обмежень, що ускладнює їх адаптацію до потреб закладів вищої освіти. Серед них — відсутність гнучкої логіки контролю за проходженням опитування, неможливість встановити таймер чи обмежити повторне проходження, обмеження у збереженні та аналізі даних. Окрім цього, такі платформи зазвичай не дозволяють локальне розгортання або зберігання даних на власному сервері, що є важливим у контексті захисту персональної інформації.

В умовах цифровізації та необхідності дотримання норм академічної доброчесності виникає потреба у створенні адаптивного інструменту, що дозволяє не лише проводити опитування, а й автоматично аналізувати відповіді, формувати звіти, здійснювати фільтрацію, перевірку правильності відповідей тощо. Важливо також забезпечити інтуїтивний інтерфейс як для користувача (респондента), так і для особи, що адмініструє опитування (викладача, соціолога, працівника деканату тощо).

У зв’язку з цим було обґрунтовано необхідність створення нової веб-платформи UniSurvey, орієнтованої на потреби студентської аудиторії та освітніх установ.

Метою кваліфікаційної роботи є розробка функціонального веб-застосунку для організації, проходження та аналізу онлайн-опитувань із можливістю налаштування ключових параметрів: обмеження повторного

проходження, встановлення таймера, виведення статистики та експорту результатів.

Основні завдання дослідження:

- проаналізувати сучасні підходи до автоматизації збору даних в освітньому середовищі;
- дослідити функціонал аналогічних систем та виявити їхні недоліки;
- обґрунтувати вибір технологій реалізації;
- спроєктувати структуру бази даних та логіку взаємодії користувачів із системою;
- реалізувати ключові модулі: таймер, обробка форм, експортування у CSV/Excel;
- провести тестування функціональності платформи та оцінити її ефективність.

Об’єктом дослідження є процес автоматизації організації онлайн-опитувань у студентському середовищі.

Предметом дослідження виступає веб-платформа UniSurvey, її структура, логіка взаємодії та функціональні можливості.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1Сучасні підходи до організації онлайн-опитувань

Онлайн-опитування стали невід'ємною частиною сучасного освітнього процесу, особливо в умовах цифровізації та дистанційного навчання. Вони дозволяють швидко та ефективно збирати зворотний зв'язок від учасників освітнього процесу, що є важливим для підвищення якості освіти.

У 2024 році Міністерство освіти і науки України ініціювало всеукраїнське опитування для збору даних про якість освіти, що підкреслює значущість таких інструментів для прийняття управлінських рішень у сфері освіти [1]. Подібні ініціативи не є поодинокими — вже у 2025 році МОН анонсувало запуск нової хвилі масштабного онлайн-опитування серед школярів, що демонструє сталість цифрових практик в освітньому середовищі [2].Сучасні підходи до організації онлайн-опитувань передбачають:

- швидке розгортання опитування без потреби в глибоких технічних знаннях;
- гнучкість у створенні структури опитувальника, включаючи різні типи питань (відкриті, множинний вибір, шкала тощо);
- захист даних респондентів, зокрема через авторизацію та обмеження доступу;
- автоматичну обробку результатів, генерацію статистики та візуалізацій;
- адаптивність інтерфейсу для використання на різних пристроях (ПК, смартфони, планшети);
- інтеграцію з іншими системами або можливість експорту у формати CSV/Excel.

Застосування онлайн-опитувань у навчальному процесі дозволяє адміністрації закладів освіти отримувати оперативний зворотний зв'язок від студентів щодо якості навчання, викладання окремих дисциплін або проведення внутрішніх досліджень. Такі системи мають на меті не лише збирання

відповідей, а й їх автоматичну обробку, збереження в базі даних та надання адміністративного доступу для перегляду результатів.

Крім того, сучасні платформи дедалі частіше використовують принципи UX/UI-дизайну, що підвищує зручність взаємодії користувачів із системою опитувань та позитивно впливає на повноту і достовірність зібраних даних.

1.2 Аналіз функціоналу існуючих платформ

Для обґрунтування необхідності створення власної платформи для онлайн-опитувань доцільно провести аналіз найбільш поширених існуючих систем, зокрема таких як Google Forms, SurveyMonkey та LimeSurvey. Кожна з них має певні переваги та обмеження, що варто враховувати при проєктуванні нової системи.

Google Forms — безкоштовний інструмент для створення форм і опитувань, інтегрований у Google Workspace. Має простий інтерфейс, підтримку різних типів питань (текстові, вибір одного чи кількох варіантів, шкала), можливість автоматичного збору відповідей у Google Sheets та побудови графіків. Серед обмежень — неможливість встановлення таймера, обмеження повторного проходження без Google-авторизації, обмежені функції

статистики [3].

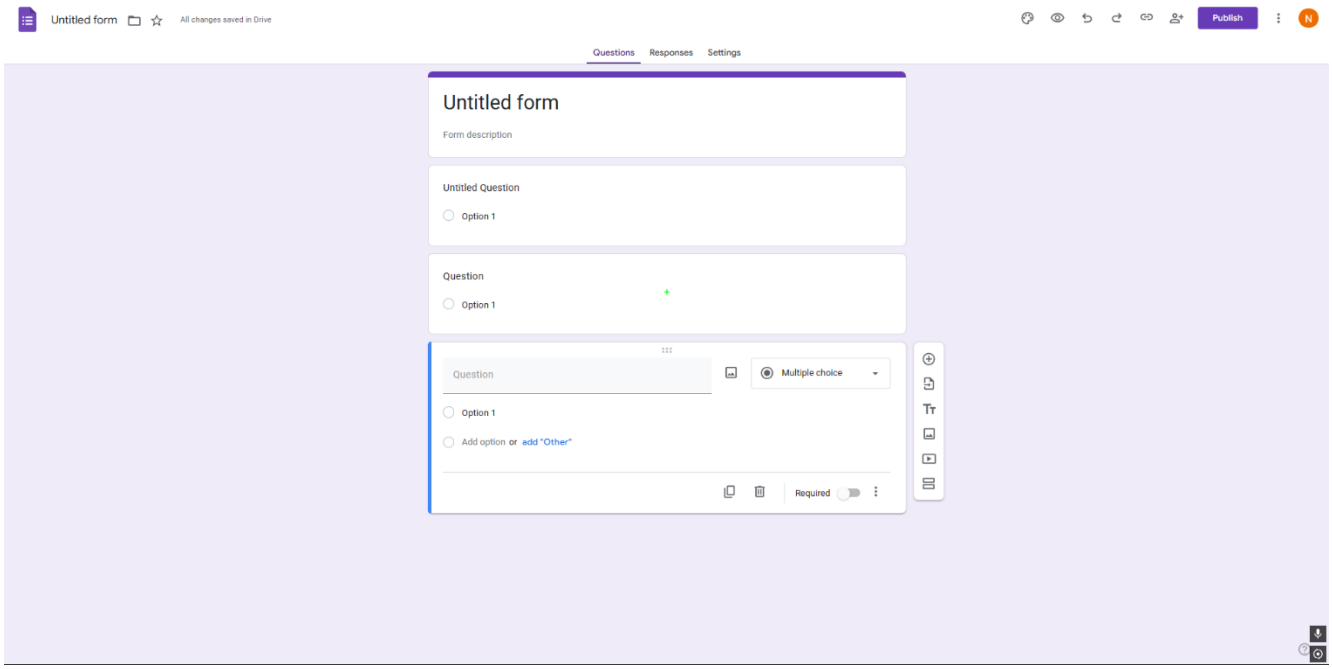


Рис. 1.1 Google Forms (створення опитування)

SurveyMonkey — комерційна платформа для створення складних опитувань, орієнтована на бізнес та дослідницькі потреби. Підтримує логіку переходів між питаннями, аналітику відповідей, захист паролем, різні типи звітів. Основним недоліком є обмеження функціоналу у безкоштовній версії та складніша інтеграція з іншими системами.

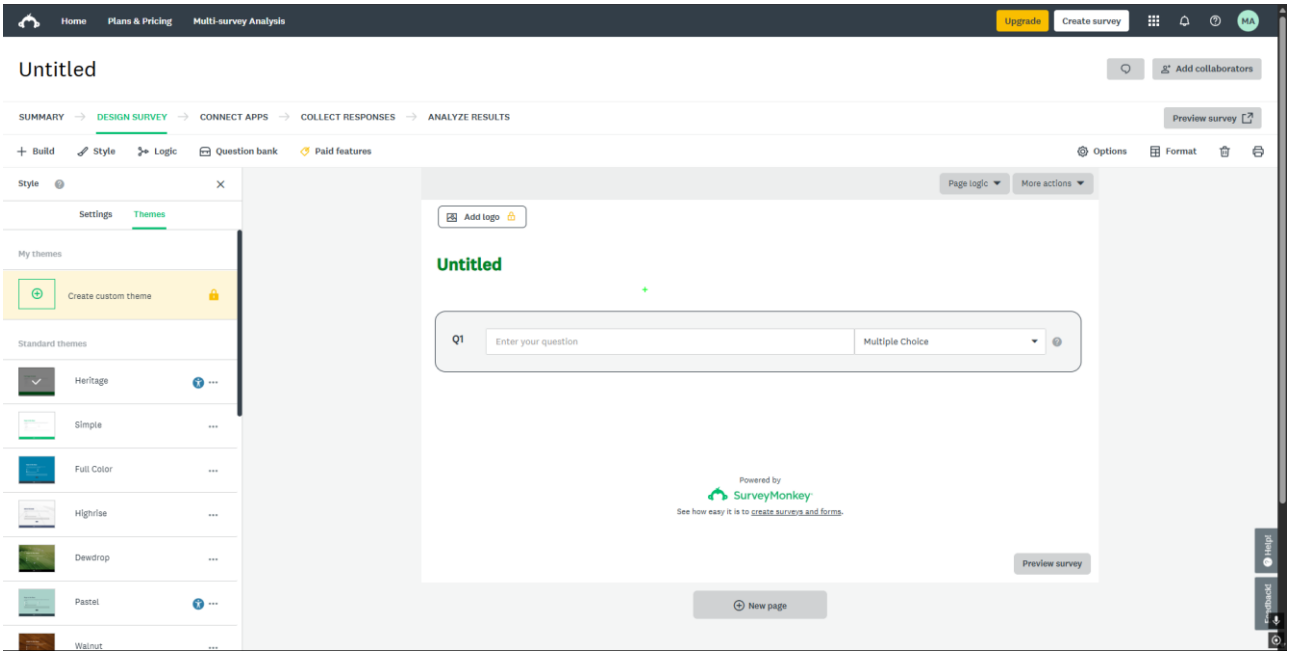


Рис. 1.2 SurveyMonkey (створення опитування)

LimeSurvey — опенсорс-платформа, яка дозволяє розгортати власний сервер для проведення опитувань. Підтримує багато типів питань, створення шаблонів, контроль за сесією користувача, розгалуження сценаріїв. Хоча LimeSurvey має широкий функціонал, її інтерфейс менш інтуїтивний, ніж у комерційних аналогів, і потребує технічної підтримки для налаштування.

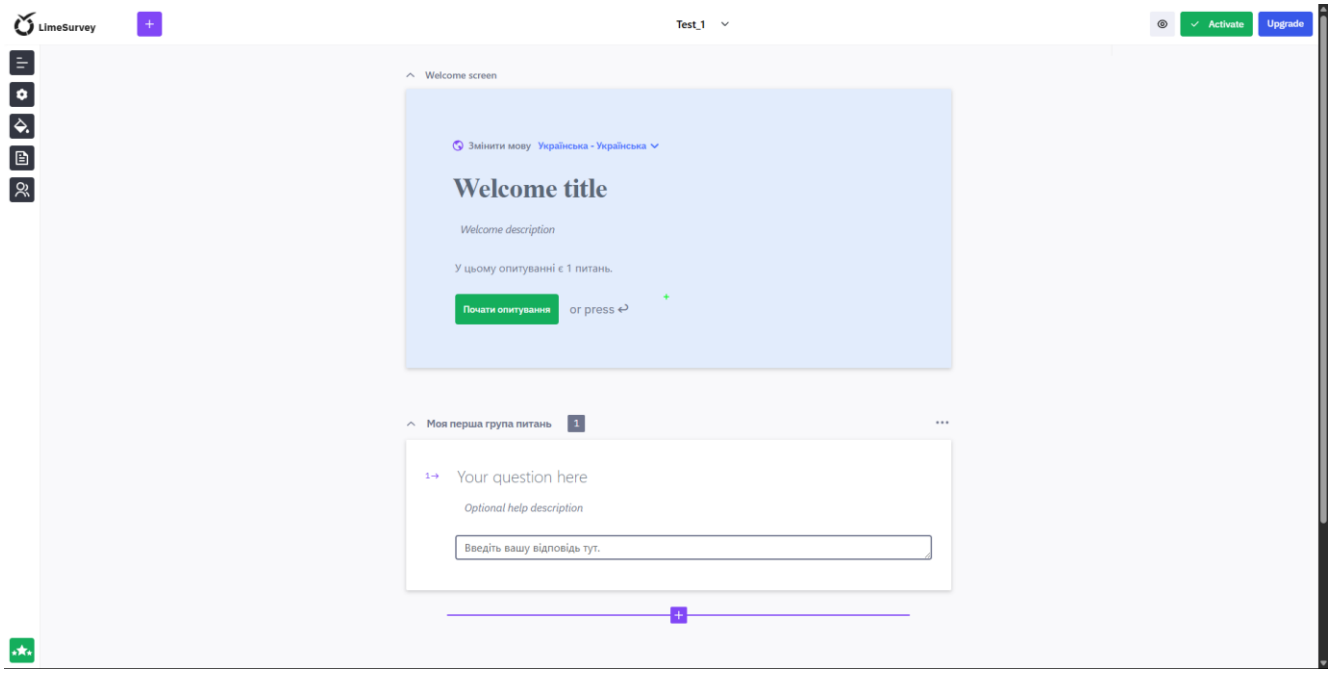


Рис. 1.3 LimeSurvey (створення опитування)

У таблиці нижче наведено порівняння ключових функцій згаданих платформ:

Таблиця 1.1

Порівняльна характеристика функціональних можливостей платформ

	Google Forms	SurveyMonkey	LimeSurvey
Завантаження результатів стороннього опитування	Ні	Ні	Ні
Імпорт результатів з Excel/CSV	Ні	Ні	Ні
Формування звітів	Ні	Так	Ні
Статистика	Ні	Ні	Так
Автоматичне відображення результатів	Так	Ні	
Обмеження за іменем	Ні	Ні	Ні

Таким чином, жодна з існуючих систем не поєднує в собі одночасно простоти, контролю проходження (таймер, обмеження) та повноцінного аналізу даних безкоштовно або в межах відкритого коду. Це підтверджує актуальність розробки власного застосунку з адаптованим функціоналом для потреб закладів освіти.

1.3 Обґрунтування потреби у створенні власної системи

Проведений аналіз існуючих онлайн-платформ для створення опитувань, таких як Google Forms, SurveyMonkey та LimeSurvey, дозволив виявити низку спільних обмежень, які ускладнюють або унеможливлюють їх ефективне

використання в освітньому середовищі. Незважаючи на широке розповсюдження та базову функціональність, жодна з розглянутих систем не надає користувачу повного контролю над процесом проходження опитування, обробкою результатів та безпечністю даних.

Серед основних обмежень варто виділити:

- відсутність механізмів ідентифікації респондентів (захист від повторного проходження за іменем);
- неможливість встановлення таймера, що особливо важливо для обмеження часу на виконання контрольних чи дослідницьких анкет [4], [5];
- обмеження у вивантаженні зовнішніх результатів опитування або автоматизованому формуванні звітів;
- складність розміщення та адміністрування платформи у межах інфраструктури університету.

Дані спостереження підтверджуються як аналізом документації платформ, так і зовнішніми дослідженнями щодо їх використання в освітньому середовищі [6].

З урахуванням зазначених недоліків постає завдання розробити власне рішення — веб-застосунок, який поєднує простоту користування з широкими можливостями керування опитуваннями та аналізом відповідей. Така система має бути адаптована до потреб закладів вищої освіти, з можливістю її локального розміщення, відкритої модифікації та гнучкої настройки логіки опитування.

Отримані висновки формують підґрунтя для подальшого проєктування структури системи, що детально розглядається у наступному розділі.

1.4 Вимоги до систем автоматизованого опитування

Після аналізу функціональних можливостей існуючих рішень та виявлення їх обмежень, сформовано перелік вимог до системи автоматизованого

опитування, яка має бути адаптована для використання у вищих навчальних закладах. Ці вимоги охоплюють як функціональні, так і нефункціональні аспекти, що є критично важливими для забезпечення ефективної, зручної та безпечної роботи платформи.

Функціональні вимоги:

- Можливість створення опитувань з різними типами запитань (текстові, з вибором однієї або кількох відповідей);
- Додавання таймера для обмеження часу проходження опитування;
- Контроль повторного проходження опитування на основі імені користувача;
- Відображення результатів після завершення опитування (за вибором адміністратора);
- Формування статистичних звітів для кожного опитування;
- Експорт результатів у формати CSV та Excel;
- Імпорт результатів;
- Адміністративний інтерфейс для створення, редагування, публікації та аналізу опитувань.

Нефункціональні вимоги:

- Веб-доступність платформи з будь-якого пристрою (адаптивність);
- Простота інтерфейсу для студентів і викладачів;
- Безпечне збереження результатів у базі даних;
- Можливість розгортання на локальному сервері університету;
- Незалежність від платних ліцензій (використання вільного програмного забезпечення);
- Можливість масштабування та розширення функціоналу в майбутньому.

Сформовані вимоги стали основою для подальшого проєктування архітектури системи, визначення її компонентів та способу взаємодії між користувачами і серверною частиною.

1.5. Вибір технологій розробки

Розробка сучасного веб-застосунку вимагає обґрунтованого вибору технологій, які забезпечать надійність, масштабованість, безпеку, зручність у використанні та подальший розвиток. На етапі планування реалізації системи автоматизованого опитування було проаналізовано популярні інструменти веб-розробки, серед яких — мови програмування (PHP, Java, C#, Python), фреймворки (Laravel, Spring Boot, ASP.NET Core, Flask, Django), системи керування базами даних (MySQL, PostgreSQL, SQLite) та засоби побудови інтерфейсу користувача.

Python

Основною мовою програмування було обрано Python — високоуровневу, зручну та зрозумілу мову з широким набором бібліотек. Python активно використовується у сфері веб-розробки, аналізу даних, штучного інтелекту та автоматизації. Його простий синтаксис і активна спільнота дозволяють швидко створювати і підтримувати надійні програмні продукти.

Особливістю Python є можливість реалізації навіть складної логіки з мінімальним обсягом коду.

Django

Для побудови серверної частини застосунку обрано фреймворк Django, який побудований на мові Python. Django є одним із найпопулярніших веб-фреймворків з відкритим кодом і забезпечує високу швидкість розробки. Його ключові переваги — вбудована ORM (Object-Relational Mapping), наявність адміністративної панелі "з коробки", реалізовані засоби аутентифікації та захисту, а також модульність. Django дотримується принципів DRY (Don't Repeat Yourself) і MTV (Model-Template-View), що дозволяє ефективно структурувати код і пришвидшити реалізацію функціоналу.

Саме Django забезпечує реалізацію логіки опитувань, збереження результатів, таймерів, експорту у формати CSV/Excel та обмеження за іменем.

PostgreSQL

Серед багатьох систем управління базами даних (СКБД) перевага була надана PostgreSQL. Це потужна об’єктно-реляційна СКБД, що підтримує складні запити, транзакції, цілісність даних та масштабування. PostgreSQL відзначається стабільною роботою з великими обсягами інформації, гнучкістю налаштування та активною підтримкою розробників.

У порівнянні з MySQL та SQLite, PostgreSQL забезпечує кращу підтримку складних зв’язків між таблицями, що є важливим у контексті опитувань із багатьма варіантами відповідей та відповідями користувачів.

HTML / CSS / JavaScript

Клієнтська частина системи реалізована з використанням стандартних технологій HTML, CSS та JavaScript.

- HTML відповідає за структуру веб-сторінок.
- CSS забезпечує візуальне оформлення та адаптивність інтерфейсу.
- JavaScript додає інтерактивність — наприклад, динамічне додавання варіантів відповіді, індикатор таймера тощо.

Застосування цього стека дозволяє забезпечити адаптивний дизайн, сумісність із сучасними браузерами та зручність використання платформи як для студентів, так і для адміністраторів.

Альтернативи, які розглядалися

Під час планування архітектури платформи були розглянуті й інші популярні технології веб-розробки, такі як PHP з Laravel, Java з Spring Boot, JavaScript з Node.js, а також Firebase від Google як серверless-рішення.

Однак кожна з цих альтернатив має обмеження у контексті освітнього проєкту:

- Laravel (PHP) потребує більше налаштувань для зв’язку між ORM та логікою опитувань, а екосистема PHP має нижчу актуальність у порівнянні з Python у сфері освітніх проєктів.
- Java + Spring Boot — потужний варіант, але занадто громіздкий для невеликої системи з простою логікою. До того ж, час налаштування значно більший, ніж у Django.
- Node.js (Express) забезпечує гнучкість, але вимагає додаткових модулів для реалізації ORM, шаблонізації, захисту і перевірок.
- Firebase — зручне рішення для швидкого MVP, однак не дозволяє легко реалізувати складні структури з логікою ManyToMany, обмеженнями за іменем та експортом у Excel, без значної ручної роботи з Cloud Functions.

Таким чином, Django забезпечив баланс між швидкою розробкою, логікою моделей, зручністю форм та вбудованим адмініструванням, а PostgreSQL — масштабовану і стабільну роботу з даними.

2 ПРОЄКТУВАННЯ СИСТЕМИ ОПИТУВАНЬ

2.1 Постановка завдання

Проблема ефективного збору, обробки та аналізу відповідей у межах освітнього процесу залишається актуальною для багатьох закладів вищої освіти України. Більшість доступних онлайн-сервісів для створення опитувань не забезпечують необхідного рівня контролю над проходженням опитувань, гнучкості в налаштуваннях, функцій автоматичного обмеження доступу або формування детальних звітів. Крім того, залежність від сторонніх платформ унеможливлює адаптацію системи до конкретних вимог навчального закладу або локальне розміщення на внутрішніх серверах.

Завданням даної кваліфікаційної роботи є розробка веб-платформи для створення, проходження та адміністрування онлайн-опитувань, яка:

- забезпечує зручний та інтуїтивний інтерфейс для кінцевих користувачів;
- підтримує різні типи питань (відкрите, вибір одного, вибір кількох);
- дозволяє адміністратору створювати опитування з таймером проходження;
- забезпечує контроль повторного проходження за іменем;
- здійснює автоматичне збереження результатів у базі даних;
- дозволяє експортувати результати у форматах CSV та Excel;
- забезпечує відображення статистики у вигляді таблиць.

Система повинна мати адаптивний веб-інтерфейс, можливість локального розгортання, простий механізм внесення змін до структури опитувань та бути побудована на основі відкритих технологій (Python, Django, PostgreSQL, HTML/CSS/JS).

Результатом роботи має стати універсальний веб-застосунок, орієнтований на потреби університетів, з потенціалом подальшого масштабування і розширення функціональності.

Це завдання повністю відповідає стратегічним напрямам цифрової трансформації освіти, визначеним Міністерством освіти і науки України [6].

2.2 Опис структури платформи та логіки взаємодії

Розроблена платформа для онлайн-опитувань базується на модульній архітектурі, що забезпечує гнучкість, масштабованість та зручність у підтримці. Основні компоненти системи включають:

- Клієнтський інтерфейс (Frontend): забезпечує взаємодію користувача з системою через веб-браузер. Реалізований з використанням HTML, CSS та JavaScript.
- Серверна частина (Backend): відповідає за обробку запитів від клієнта, бізнес-логіку та взаємодію з базою даних. Реалізована за допомогою фреймворку Django на мові Python.
- База даних: зберігає інформацію про користувачів, опитування, відповіді та інші необхідні дані. Використовується система управління базами даних PostgreSQL.
- Адміністративна панель: надає можливість адміністраторам системи створювати та редагувати опитування, переглядати результати та керувати користувачами.

Взаємодія між компонентами системи відбувається за наступною логікою:

1. Користувач через веб-браузер надсилає запит до серверної частини для отримання сторінки опитування.
2. Серверна частина обробляє запит, звертається до бази даних для отримання необхідної інформації та формує відповідь у вигляді HTML-сторінки.
3. Користувач заповнює опитування та надсилає відповіді назад на сервер.
4. Серверна частина зберігає отримані відповіді в базі даних та може надавати адміністраторам доступ до зведених результатів.

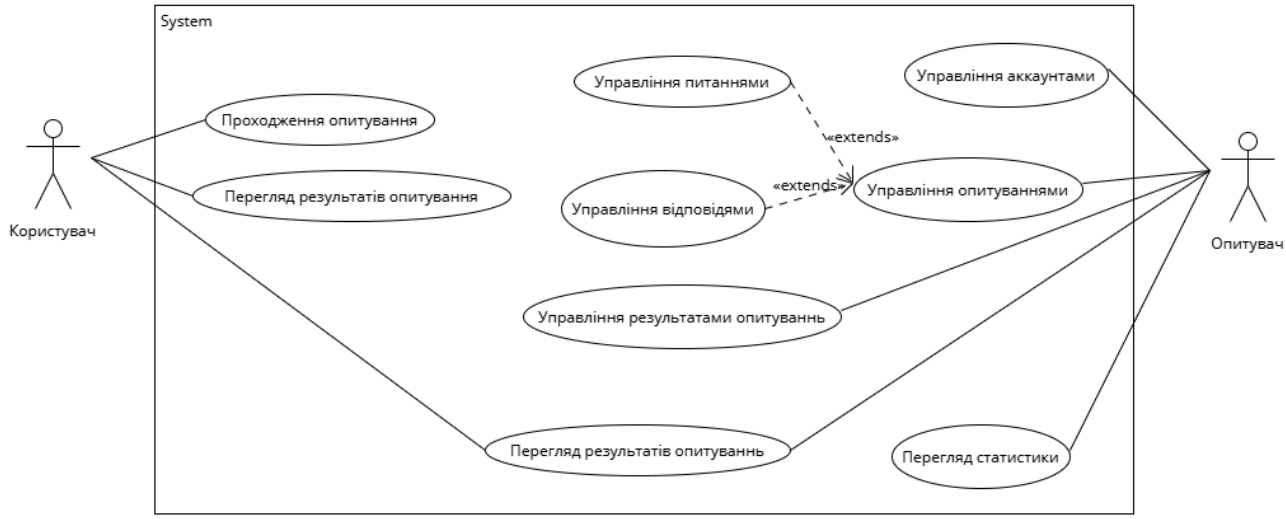


Рис 2.1 (діаграма варіантів використання)

2.3 Моделювання структури даних

Для ефективного зберігання та обробки даних у системі було змодельовано логічну структуру бази даних із використанням діаграми класів UML. Це дозволяє візуалізувати взаємозв’язки між основними об'єктами системи: опитуваннями, питаннями, варіантами відповідей, відповідями студентів тощо. Система складається з таких основних сутностей:

- Survey – модель опитування, що містить загальну інформацію про анкету, включаючи назву, опис, статус активності, наявність таймера та обмеження за іменем.
- Question – питання, які належать до конкретного опитування та можуть мати різні типи (відкрите, один варіант, кілька варіантів).
- AnswerOption – варіанти відповідей до питання (для типів radio/checkbox).
- StudentResponse – відповідь студента на опитування. Зберігає ім’я респондента та дату проходження.
- Answer – окрема відповідь на кожне питання, що належить до конкретної сесії StudentResponse. Вона може містити текстову відповідь або обрані варіанти відповідей (AnswerOption).

Ці класи пов’язані між собою наступним чином:

- Одне опитування (Survey) може містити багато питань (Question).
- Одне питання може мати багато варіантів відповідей (AnswerOption).
- Одна відповідь студента (StudentResponse) пов’язана з багатьма відповідями на питання (Answer).
- Кожна відповідь (Answer) стосується одного питання, а також належить до одного StudentResponse.
- У разі питань з множинними варіантами, Answer пов’язується з кількома AnswerOption через зв’язок ManyToMany.

Зображення цієї структури представлено на UML-діаграмі класів (рисунок 2.2), створеній відповідно до логіки Django моделей, що реалізують ці сутності.

Використання UML-діаграм класів дозволяє створити чітке логічне уявлення про структуру об’єктів у системі, полегшуючи реалізацію, тестування та супровід програмного забезпечення [7].

2.4. Опис ролей користувачів та сценаріїв використання

Платформа UniSurvey орієнтована на дві основні ролі користувачів:

- Опитувач: відповідає за створення та керування опитуваннями, редагування питань, перегляд відповідей, експорт результатів, активацію таймера та обмеження за іменем.
- Користувач**:** має доступ до проходження активних опитувань та перегляду результатів (якщо дозволено опитувачем).

Модель ролей і взаємодії користувачів із функціоналом платформи було представлено раніше у вигляді діаграми варіантів використання (див. рисунок 2.1).

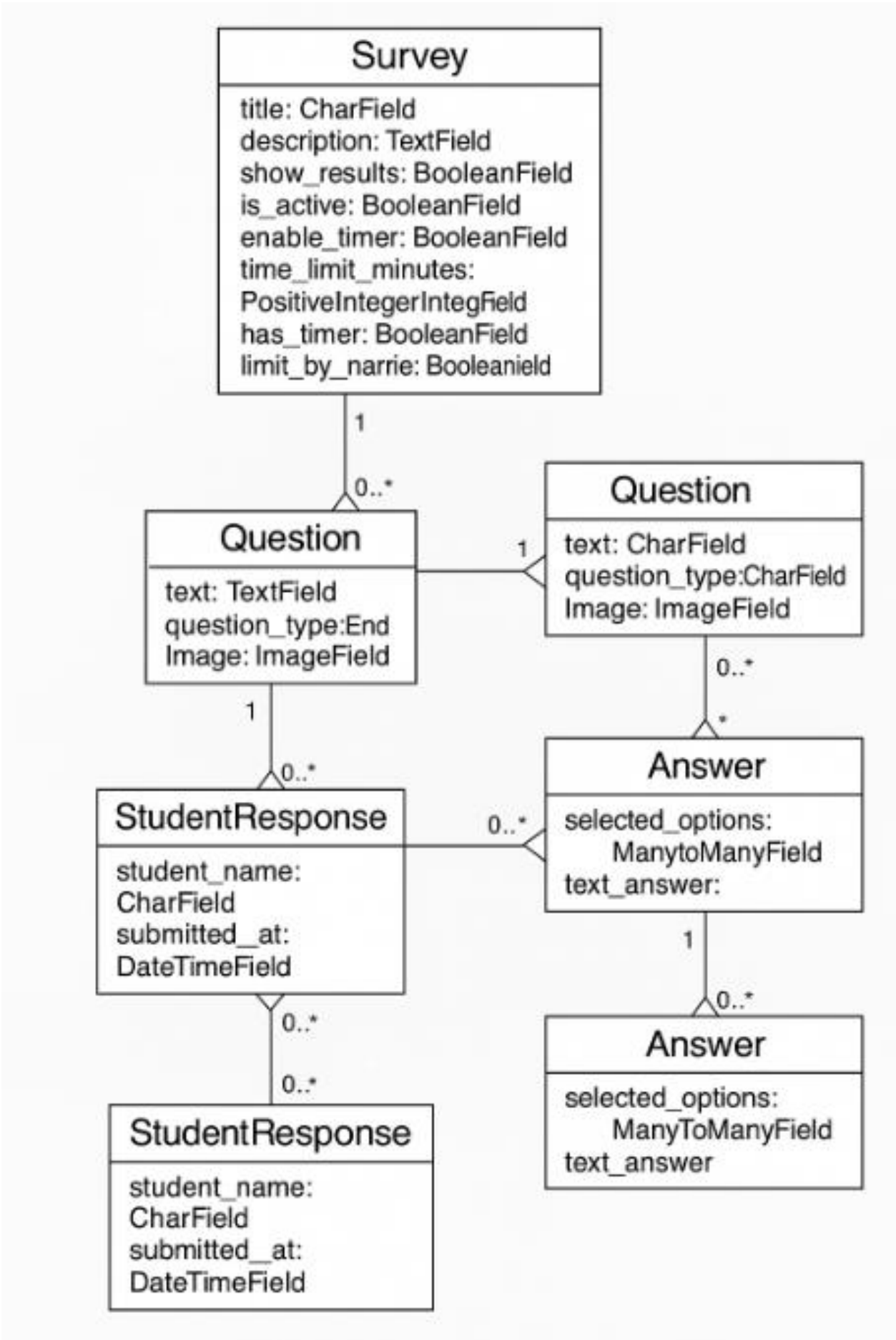


Рис. 2.2 Діаграма класів

Типові сценарії взаємодії

1. Опитувач може:

- Створити нове опитування;
- Додати або редагувати питання;
- Встановити таймер і обмеження за іменем;
- Публікувати/приховувати опитування;
- Переглядати відповіді користувачів;
- Завантажити результати у форматах CSV або Excel.

2. Користувач може:

- Переглядати доступні опитування;
- Проходити опитування;
- Переглядати результат проходження (якщо дозволено);
- Отримувати повідомлення про недопустимість повторного проходження (якщо обмеження активне).

Ці сценарії дозволяють задовольнити як потреби управління опитуваннями з боку опитувача, так і зручність проходження опитувань для користувачів. Подібна рольова модель забезпечує зрозумілу логіку доступу та функціональну безпеку платформи.

2.5. Планування логіки таймера, обмеження за іменем, збереження результатів

Для забезпечення контролю та достовірності опитувань у системі UniSurvey реалізовано спеціальні механізми, які дозволяють обмежити повторне проходження опитування, задати обмеження за часом та забезпечити надійне збереження даних у базі.

Таймер

Механізм таймера дає змогу обмежити час на проходження опитування. Ця функція особливо важлива для проведення контрольних або оцінювальних опитувань.

- У моделі Survey передбачено логічні поля has_timer та timer_minutes.
- При старті опитування клієнтська частина (JavaScript) запускає зворотний відлік.
- Після завершення часу форма автоматично блокується або надсилається на сервер із поточними відповідями.
- Таймер працює локально на стороні користувача, але його налаштування зберігаються на сервері.

Обмеження за іменем

Щоб уникнути багаторазового проходження опитування однією особою, реалізовано обмеження за іменем:

- У полі student_name зберігається ім’я користувача.
- Якщо в опитуванні увімкнено поле limit_by_name, система перевіряє, чи вже існує запис з таким іменем для цього опитування.
- У разі виявлення повторного проходження користувач отримує повідомлення про відмову в доступі до опитування.
- Це дозволяє уникнути дублювання відповідей та підвищує достовірність результатів.

Збереження результатів

Усі дані, отримані в процесі проходження опитування, зберігаються в базі даних через відповідні моделі:

- Відповіді користувача зберігаються в StudentResponse, а окремі відповіді — у моделі Answer.
- Для питань з одним або кількома варіантами відповіді використовується зв’язок ManyToManyField до AnswerOption.
- Усі відповіді зберігаються з позначкою дати та часу (submitted_at), що дає змогу формувати статистику та звіти.

Реалізація цих механізмів дозволяє забезпечити контроль проходження, достовірність даних та гнучкість в організації опитувального процесу.

2.6 Формування вимог до функціональності системи

Формування функціональних вимог є ключовим етапом у розробці програмного забезпечення, що забезпечує чітке розуміння очікувань користувачів та визначає основні функції системи. У контексті розробки системи UniSurvey, яка призначена для створення та проведення онлайн-опитувань, важливо врахувати як загальні принципи формування вимог, так і специфічні потреби цільової аудиторії.

Підходи до формування функціональних вимог

Функціональні вимоги описують, що система повинна робити, тобто які функції вона повинна виконувати для задоволення потреб користувачів. Вони включають опис вхідних даних, обробку та очікувані результати. Згідно з рекомендаціями [8], ефективне формування функціональних вимог передбачає:

- Чіткість та однозначність: кожна вимога повинна бути сформульована так, щоб уникнути неоднозначностей.
- Тестованість: вимоги мають бути такими, щоб їх можна було перевірити під час тестування.
- Відповідність потребам користувачів: вимоги повинні ґрунтуватися на реальних сценаріях використання.

Подібні рекомендації узгоджуються з практиками, описаними у професійній літературі з розробки ПЗ [9].

Основні функціональні вимоги до системи UniSurvey

Враховуючи специфіку системи UniSurvey, основні функціональні вимоги включають:

- Реєстрація та автентифікація користувачів.
- Створення та редагування опитувань.
- Проходження опитувань із підтримкою таймера та обмеження за іменем.

- Збереження, аналіз та експорт результатів.
- Рольова модель доступу для опитувачів і користувачів.

Висновки

Формування чітких функціональних вимог на етапі проєктування дозволяє мінімізувати ризики неправильного розуміння функцій, оптимізує розробку та сприяє якості підсумкової системи [8], [9].

2.7. Ризики проєктування та способи їх мінімізації

У процесі проєктування та реалізації веб-застосунку для проведення опитувань виникає низка ризиків, які можуть вплинути на якість, функціональність або терміни виконання проєкту. Їх вчасне виявлення та аналіз дозволяє сформувати відповідні стратегії зменшення негативних наслідків.

Основні ризики:

1. Невизначені або змінні вимоги.
   У процесі реалізації може виявитися, що функціональні потреби користувачів змінюються або були недостатньо чітко сформульовані. *Спосіб мінімізації:* детальне документування вимог (див. п. 2.6), погодження з науковим керівником, реалізація MVP (мінімально життєздатного продукту) з можливістю масштабування.
2. Технічна складність та помилки в реалізації.
   Неправильна реалізація логіки (наприклад, перевірки обмежень, таймера або обробки форм) може спричинити втрату даних або некоректні результати.
   Спосіб мінімізації: покриття критичних компонентів системи тестами (див. розділ 4), ревізія коду, перевірка сценаріїв.
3. Проблеми сумісності з різними пристроями.
   Існує ризик, що інтерфейс буде працювати некоректно на певних браузерах або мобільних пристроях.

Спосіб мінімізації: адаптивна верстка, тестування в популярних браузерах (Google Chrome, Firefox, Edge), дотримання принципів доступності (WCAG) [15].

4. Безпека даних та повторне проходження опитувань. Користувачі можуть намагатися обійти обмеження або модифікувати дані. *Спосіб мінімізації:* обмеження за іменем, перевірка унікальності, захищена робота з базою даних, валідація на серверній стороні.
5. Труднощі з розгортанням або міграцією бази даних. У разі переходу на інший сервер або змін у моделях можлива втрата даних. *Спосіб мінімізації:* використання системи міграцій Django (makemigrations, migrate), регулярне резервне копіювання.

Висновок.

Урахування потенційних ризиків ще на етапі проєктування дозволяє не лише уникнути критичних помилок у майбутньому, але й підвищити якість програмного забезпечення загалом. Правильне планування, використання перевірених інструментів і регулярне тестування — основа надійної системи.

3 РЕАЛІЗАЦІЯ ТА ІНТЕГРАЦІЯ СИСТЕМИ

3.1. Вибір середовища розробки

Вибір інструментів та середовища розробки є критичним етапом у процесі створення надійного, масштабованого та підтримуваного програмного забезпечення. Рішення приймалося з урахуванням цілей проєкту, вимог до функціональності та досвіду розробника.

Аналіз альтернатив

На початковому етапі було розглянуто кілька технологій для реалізації платформи:

- PHP (Laravel) — популярний фреймворк, однак вимагає більшої кількості налаштувань для реалізації безпечного API та ефективної роботи з базою даних. Крім того, екосистема Python є більш дружньою для освітніх та дослідницьких проєктів.
- Firebase (Google) — хмарне рішення з вбудованим хостингом та базою даних (Firestore). Попри швидкий старт, його використання унеможливлює повний контроль над даними, а також ускладнює реалізацію кастомної логіки опитувань, таймерів та перевірок.
- Node.js + Express — надає більшу гнучкість, однак потребує більше часу на реалізацію базової логіки, яку Django вже має "з коробки".

З огляду на ці фактори, для реалізації системи UniSurvey було обрано стек Python + Django + PostgreSQL, який відповідає сучасним вимогам до безпеки, гнучкості та ефективності.

Серверна частина (Back-end)

Основу серверної логіки становить фреймворк Django, який написаний мовою Python. Django дозволяє ефективно реалізовувати:

- CRUD-операції;
- роботу з базою даних;

- автентифікацію та захист від CSRF;
- шаблони HTML і механізм форм;
- вбудовану адміністративну панель.

Python обрано як мову програмування завдяки її лаконічному синтаксису, широкій екосистемі бібліотек та активній спільноті підтримки. Django має вбудовану ORM (Object-Relational Mapping), яка дозволяє взаємодіяти з базою даних за допомогою Python-класів, уникаючи прямого використання SQL [10].

База даних

Для зберігання інформації було обрано PostgreSQL — надійну реляційну СУБД з відкритим кодом. Вона підтримує:

- складні зв’язки між таблицями;
- перевірки даних на рівні схеми;
- транзакції та цілісність;
- розширення, резервне копіювання та масштабування.

PostgreSQL забезпечує високу продуктивність і добре інтегрується з Django ORM.

Клієнтська частина (Front-end)

Інтерфейс реалізовано за допомогою HTML, CSS та JavaScript. Основний акцент зроблено на доступність і простоту:

- Використано адаптивну верстку для підтримки мобільних пристроїв.
- Усі стилі реалізовано вручну без додаткових CSS-фреймворків, що дозволило повністю контролювати зовнішній вигляд.
- Використано стандартний шрифт Roboto, з темною темою дизайну та анімацією елементів форм.

Інструменти розробника

- Середовище розробки: Visual Studio Code із розширеннями для Django, Python, автоформатування (Black, Prettier), Linting.
- Віртуальне середовище: venv для ізоляції залежностей.
- Менеджер пакетів: pip — для встановлення бібліотек (openpyxl, django-import-export, pillow тощо).

- Тестування: вбудований інструмент TestCase від Django з використанням pytest для глибшої перевірки (у перспективі).

Висновок

Обраний набір інструментів і технологій забезпечив швидкий старт, ефективну командну розробку, безпеку, масштабованість і просту інтеграцію між усіма компонентами системи. Саме завдяки Django-підходу "батарейки в комплекті" вдалося реалізувати більшість вимог із мінімальними зусиллями.

3.2. Реалізація моделей і бази даних

У системі UniSurvey зберігання, обробка та взаємозв’язки даних реалізовані за допомогою об’єктно-реляційного відображення (ORM), яке надає фреймворк Django. Це дозволяє описати структуру бази даних через Python-класи та автоматично генерувати необхідні SQL-таблиці без ручного втручання [10].

Огляд основних моделей

У системі реалізовано такі базові моделі:

- Survey – модель опитування. Містить поля:
  - title (назва опитування),
  - description (опис),
  - is_active (стан публікації),
  - show_results (дозвіл на перегляд результатів),
  - has_timer, timer_minutes – параметри таймера,
  - limit_by_name – заборона повторного проходження.
- Question – питання опитування. Має зв’язок із Survey (ForeignKey) та містить:
  - текст питання,
  - тип питання (text, radio, checkbox),
  - можливе зображення до питання (ImageField).

- AnswerOption – варіанти відповідей до одного питання. Пов’язані з Question через ForeignKey. Кожен варіант має:
  текст,
  мітку is_correct для подальшої перевірки.
- StudentResponse – подання відповіді від користувача на певне опитування. Містить:
  ім’я користувача (student_name),
  посилання на опитування,
  дату і час проходження (submitted_at).
- Answer – окрема відповідь на конкретне питання. Пов’язана з:
  StudentResponse (через ForeignKey),
  Question (питання, на яке дана відповідь),
  AnswerOption (у випадку вибору варіантів, через ManyToManyField),
  або текстове поле text_answer (для відкритих питань).

Побудова міграцій

Для створення таблиць у базі даних використовуються команди Django:

python manage.py makemigrations

python manage.py migrate

Ці команди формують SQL-структуру автоматично на основі описаних моделей. Завдяки ORM-архітектурі, зміни у моделях можна легко оновлювати без втрати даних [10].

Зв’язки між моделями

Один Survey → багато Question

Один Question → багато AnswerOption

Один StudentResponse → багато Answer

Один Answer → одна Question і (опційно) кілька AnswerOption

Ці зв’язки відображено також у UML-діаграмі класів (див. розділ 2.3).

3.3. Обробка форм, збереження результатів, таймер і перевірка за іменем

У системі UniSurvey реалізовано повний цикл взаємодії користувача з опитуванням — від заповнення форми до автоматичного збереження результатів. Окрему увагу приділено перевірці повторного проходження за іменем, а також реалізації таймера для обмеження часу виконання.

Обробка форм і отримання відповідей

Форма опитування формується динамічно відповідно до вмісту об’єкта Survey, до якого прив’язані Question та AnswerOption. Після натискання кнопки "Надіслати", дані надсилаються через POST-запит до представлення survey_detail.

На сервері:

- Ім’я користувача зчитується з поля student_name;
- Для кожного питання створюється об’єкт Answer;
- У разі відкритого питання (text) відповідь записується в поле text_answer;
- У разі питань із вибором (radio або checkbox) в selected_options зберігаються обрані варіанти (AnswerOption) через ManyToManyField.

Усі відповіді пов’язуються з об’єктом StudentResponse, який містить загальну інформацію про проходження.

Перевірка повторного проходження (обмеження за іменем)

У моделі Survey передбачено логічне поле limit_by_name, що активує механізм перевірки:

```
if survey.limit_by_name:
    already_exists = StudentResponse.objects.filter(
        survey=survey,
        student_name__iexact=student_name
    ).exists()
```

Якщо система знаходить у базі даних відповідь із таким самим ім’ям і для того ж опитування — користувачу відмовляється в повторному проходженні, і виводиться відповідне повідомлення [10].

Таймер на проходження

Таймер реалізовано на стороні клієнта з використанням JavaScript:

- Перед початком проходження завантажується значення таймера (timer_minutes);
- Запускається зворотний відлік;
- По завершенню часу відбувається автоматичне надсилання форми або її блокування (залежно від реалізації);
- Усі відповіді все одно зберігаються на сервері, якщо користувач встиг надіслати форму.

Для підвищення надійності серверна частина також зберігає дату та час надсилання (submitted_at) — це дозволяє перевірити, чи була форма відправлена в межах встановленого часу [9].

Висновок

Таке поєднання функціоналу — обробка форм, перевірка унікальності користувача, таймер — дозволяє зробити процес опитування максимально контрольованим, що особливо важливо для закладів вищої освіти, де подібні інструменти використовуються для контролю знань, збору фідбеку та досліджень.

3.4. Створення та управління опитуваннями в інтерфейсі адміністратора

Інтерфейс адміністратора в системі UniSurvey реалізований на основі стандартного адміністративного модуля Django, що забезпечує ефективне управління опитуваннями, питаннями та відповідями. Цей інтерфейс дозволяє

адміністраторам створювати, редагувати та видаляти опитування, а також аналізувати результати.

Реєстрація моделей в адміністративному інтерфейсі

Для забезпечення управління опитуваннями в адміністративному інтерфейсі, необхідно зареєструвати відповідні моделі. Наприклад:

```
from django.contrib import admin
from .models import Survey, Question, AnswerOption
admin.site.register(Survey)
admin.site.register(Question)
admin.site.register(AnswerOption)
```

Це дозволяє адміністраторам створювати та редагувати об'єкти цих моделей через веб-інтерфейс.

Кастомізація відображення моделей

Для покращення зручності використання адміністративного інтерфейсу, можна налаштувати відображення моделей за допомогою класів ModelAdmin. Наприклад:

```
class SurveyAdmin(admin.ModelAdmin):
    list_display = ('title', 'description', 'created_at')
    search_fields = ('title',)
admin.site.register(Survey, SurveyAdmin)
```

Це дозволяє відображати додаткові поля в списку опитувань та забезпечує можливість пошуку за назвою опитування.

Інлайн-редагування пов'язаних моделей

Для зручності редагування пов'язаних моделей, таких як питання та варіанти відповідей, можна використовувати інлайн-редагування:

```
class AnswerOptionInline(admin.TabularInline):
    model = AnswerOption
    extra = 1
class QuestionAdmin(admin.ModelAdmin):
    inlines = [AnswerOptionInline]
```

admin.site.register(Question, QuestionAdmin)

Це дозволяє адміністраторам додавати варіанти відповідей безпосередньо при створенні або редагуванні питання.

Аналіз результатів опитувань

Для аналізу результатів опитувань, адміністративний інтерфейс може бути розширений за допомогою додаткових функцій або сторонніх бібліотек, таких як django-import-export для експорту даних або django-admin-interface для покращення вигляду інтерфейсу [11].

При цьому сам механізм адміністрування побудований на базі модуля django.contrib.admin, що дозволяє швидко налаштовувати панель керування моделями за допомогою класів ModelAdmin, інлайнів та фільтрів [12].

3.5. Експорт відповідей у CSV та Excel

Система UniSurvey передбачає можливість експорту результатів опитувань для подальшого аналізу. Така функціональність особливо актуальна в умовах академічного середовища, де потрібно швидко формувати звіти, обробляти статистику або архівувати результати опитувань.

Формати експорту

Система підтримує два найпоширеніших формати:

- CSV (Comma-Separated Values) — зручний для імпорту в Excel, Google Sheets, SPSS та інші аналітичні інструменти;
- Excel (XLSX) — формат, який забезпечує більш наочне представлення даних, підтримує таблиці та стилі, і підходить для офіційних звітів [13].

Функціональність експорту

Адміністратор платформи може експортувати дані з будь-якого опитування. Дані, які експортуються:

- ім’я респондента;
- дата проходження;
- запитання та надані відповіді;

- кількість правильних відповідей;
- загальний бал та максимальний можливий.

У випадку експорту у CSV використовується стандартна бібліотека csv. Цей підхід широко застосовується в Python-додатках для формування табличних даних [14]:

```
response = HttpResponse(content_type="text/csv")
response["Content-Disposition"] = "attachment; filename=results.csv"
writer = csv.writer(response)
writer.writerow(["Ім’я", "Дата", "Бал", "Макс. бал"])
```

Для формування Excel-файлів використовується бібліотека openpyxl:

```
wb = openpyxl.Workbook()
ws = wb.active
ws.title = "Результати"
ws.append(["Ім’я", "Питання", "Відповідь", "Дата"])
# Запис по кожному респонденту
wb.save(response)
```

Ці фрагменти коду вбудовані у відповідні представлення Django та доступні лише для авторизованих користувачів із правами адміністратора.

Переваги експорту

- Швидке формування звітів у зручному форматі;
- Збереження архіву проходжень опитувань;
- Можливість подальшої обробки в аналітичних середовищах;
- Адаптація під потреби внутрішнього аудиту або зовнішньої перевірки.

Обмеження доступу

Експорт доступний виключно через захищені маршрути та лише для користувачів з правами адміністратора (через декоратор @staff_member_required), що відповідає вимогам захисту персональних даних.

3.6. Інтерфейс користувача: основні сторінки, шаблони

Інтерфейс користувача є важливою складовою платформи UniSurvey, оскільки саме через нього відбувається взаємодія респондента із системою. Його структура повинна бути інтуїтивно зрозумілою, простою та доступною для всіх категорій користувачів, зокрема студентів, викладачів та адміністративного персоналу.

Технології реалізації

Для створення інтерфейсу використовувалися класичні вебтехнології:

- HTML — для структури сторінки;
- CSS — для стилізації елементів;
- JavaScript — для динамічної поведінки (зокрема реалізації таймера, анімацій, повідомлень);
- Django шаблони (Django Templates) — для генерації сторінок на основі даних із серверної частини.

Це дозволяє розділити логіку відображення та логіку обробки даних, підтримувати масштабованість і повторне використання компонентів.

Основні сторінки інтерфейсу користувача

1. Головна сторінка (index.html):
   Містить привітання, загальну інформацію про платформу, кнопки переходу до проходження опитувань або до адміністрування.
2. Сторінка вибору опитування (survey_start.html):
   Містить список активних опитувань, фільтр пошуку, відображення кількості питань. Дозволяє перейти до детального перегляду.
3. Сторінка проходження опитування (survey_detail.html):
   Відображає питання з відповідними типами: текст, radio, checkbox. Реалізовано перевірку за іменем, запуск таймера, повідомлення про помилки або успіх.

4. Сторінка результатів (survey_result.html): Після завершення показує кількість правильних відповідей (за наявності), повну статистику або повідомлення, що результати недоступні.
5. Інтерфейс адміністратора (manage_surveys.html, edit_survey.html): Реалізовано функції створення, редагування, перегляду опитувань, додавання питань, відмітки правильних відповідей, кнопки активації/деактивації.
6. Сторінка статистики (survey_statistics.html): Відображає результати респондентів у вигляді таблиці, з балами, правильними відповідями та можливістю фільтрації.
7. Сторінка завантаження результатів (upload_results.html): Дозволяє імпортувати відповіді з файлів Excel або CSV, які були зібрані за межами системи.

Особливості дизайну

- Темна тема інтерфейсу з адаптивною верстkою для мобільних пристроїв;
- Стилі оформлення узгоджені через змінні CSS (--blue, --green, тощо);
- Використання кастомних класів btn, question-box, message для стилізованого UX;
- Повідомлення про успішне збереження/помилки реалізовані через блочні елементи ({% if messages %}...{% endif %});
- У шаблонах застосовується спадкування від базового шаблону base.html для єдності стилю.

Приклад шаблонного блоку:

```
{% for question in questions %}
  <div class="question-box">
    <label>{{ question.text }}</label>
    {% if question.question_type == "text" %}
      <input type="text" name="question_{{ question.id }}">
    {% elif question.question_type == "radio" %}
```

```
        {% for option in question.options.all %}
          <label><input type="radio" name="question_{{ question.id }}" value="{{
option.id }}"> {{ option.text }}</label>
        {% endfor %}
      {% endif %}
  </div>
{% endfor %}
```

Адаптивність і доступність

Весь інтерфейс протестовано на різних пристроях: ПК, планшетах, смартфонах. Основні елементи мають великі області кліку, відповідний контраст, повідомлення зворотного зв’язку, що відповідає рекомендаціям з веб-доступності (WCAG) [15].

3.7. Адміністративна частина платформи

Адміністративна частина системи UniSurvey реалізована для забезпечення повного контролю над опитуваннями, їх структурою, респондентами та результатами. Вона доступна лише авторизованим користувачам з правами адміністратора (staff).

Можливості адміністратора

Адміністратор системи має змогу:

- створювати нові опитування, вказуючи назву, опис, таймер, обмеження тощо;
- додавати до кожного опитування питання різних типів (відкрите, один або кілька варіантів);
- прикріплювати зображення до питань;
- позначати правильні відповіді;
- публікувати або приховувати опитування;

- переглядати відповіді респондентів у вигляді таблиць із фільтрами;
- експортувати результати у CSV або Excel;
- завантажувати зовнішні результати (наприклад, із Google Forms);
- бачити статистику кількості відповідей, середніх балів, динаміки проходження.

Інтерфейс керування

Інтерфейс реалізовано на окремій сторінці manage/, де доступна таблиця опитувань зі швидким переглядом кількості питань, статусу (активне/приховане), та кнопками:

- Редагувати
- Статистика
- Опублікувати / Приховати
- Видалити

Також реалізовано:

- Пошук опитування за назвою;
- Сортування за датою створення;
- Перехід до режиму створення нового опитування;
- Перегляд статистики конкретного респондента;
- Інтерфейс для завантаження результатів через upload_results.html.

Захист та доступ

Всі маршрути адміністративної частини захищені за допомогою декораторів:

```
@staff_member_required
def manage_surveys(request):
```

Це унеможливлює несанкціонований доступ до функцій створення/редагування/експорту, що відповідає вимогам захисту персональних даних і логіки розмежування прав [16].

Додаткові інструменти

- Реалізовано кастомне адміністрування моделей через ModelAdmin;
- Додано інлайн-редагування (TabularInline) для варіантів відповідей;
- Використано систему повідомлень Django (messages.success, messages.error) для зворотного зв’язку;
- Застосовано модифікацію шаблонів manage_surveys.html, edit_survey.html з адаптивним дизайном.

Приклад структури сторінки керування

- Верхня панель із пошуком і кнопкою створення нового опитування;
- Список усіх опитувань зі статистикою;
- Кнопки керування кожним опитуванням;
- Перехід до візуалізації відповідей та статистики.

4. ТЕСТУВАННЯ ТА ОЦІНКА СИСТЕМИ

4.1. Опис методу тестування

Тестування — невід'ємна частина життєвого циклу програмного забезпечення, яка забезпечує перевірку коректності реалізації логіки та стабільність функціонування системи. У розробці системи UniSurvey було застосовано юніт-тестування — підхід, що передбачає перевірку окремих функціональних блоків коду в ізоляції від решти системи.

Використаний інструментарій

Для реалізації тестів було використано вбудовану систему тестування Django, яка базується на unittest — стандартному фреймворку Python. Django надає зручний інтерфейс до бази даних, створює тимчасові таблиці під час тестування та дозволяє проводити тести без впливу на реальні дані [17].

Структура тестів

Усі тести реалізовано у файлі tests.py в межах додатку surveys. Для їх запуску використовується команда:

python manage.py test surveys

В основному було протестовано наступне:

- Обмеження повторного проходження:
  Перевіряється, що при активованому прапорці limit_by_name користувач із тим самим іменем не може пройти опитування повторно.
- Таймер:
  Тестується правильність збереження таймера (has_timer, timer_minutes) у моделі Survey.
- Створення відповідей:
  Перевіряється, що після проходження опитування відповіді коректно зберігаються у пов'язані моделі Answer, StudentResponse.

- Експорт:
  Тестується наявність збережених відповідей, що дозволяє їх експортувати у форматах CSV та Excel.
- Створення нового питання:
  Перевіряється, що нове питання зберігається в базі і кількість питань у опитуванні збільшується.
- Сторінки інтерфейсу:
  Тестується доступність ключових представлень (наприклад, /survey/start/ повертає статус 200).

Приклад тесту:

```
def test_limit_by_name_blocks_repeat_response(self):
    StudentResponse.objects.create(
        survey=self.survey,
        student_name="Іван",
        submitted_at=timezone.now()
    )
    exists = StudentResponse.objects.filter(
        survey=self.survey,
        student_name__iexact="Іван"
    ).exists()
    self.assertTrue(exists)
```

Цей тест перевіряє, що користувач із ім’ям “Іван” уже має відповідь, і повторне проходження буде заблоковано.

Переваги обраного підходу

- повна ізольованість від реальної бази даних;
- швидкість виконання тестів;
- можливість автоматизації перевірки у CI/CD;
- легкість масштабування тестової бази при додаванні нового функціоналу.

4.2. Функціональне тестування основних можливостей

У межах розробки системи UniSurvey було реалізовано функціональне тестування основних сценаріїв користувача за допомогою вбудованого інструментарію Django — класу TestCase. Це дозволяє перевіряти повні робочі ланцюги, які охоплюють взаємодію між моделями, формами, представленнями та шаблонами.

Мета функціонального тестування

Перевірити, чи реалізований функціонал (таймер, проходження опитування, обмеження за іменем, збереження результатів, експорт тощо) працює відповідно до бізнес-логіки та вимог користувача. Тестування проводилося ізоляційно, з використанням окремої тестової бази даних.

Таблиця 4.1

Перелік протестованих сценаріїв

№	Сценарій	Реалізація у Django тестах
1	Створення нового опитування	Тест через створення об'єкта Survey
2	Додавання питання	Тест створення об'єкта Question
3	Проходження опитування	Перевірка створення StudentResponse
4	Обмеження за іменем (limit_by_name)	Тест з student_name__iexact
5	Налаштування та збереження таймера	Перевірка полів has_timer, timer_minutes
6	Збереження відповідей (Answer + AnswerOption)	Перевірка створення пов’язаних об’єктів
7	Експорт відповідей (перевірка наявності у базі)	Тест на Survey.responses.count()
8	Доступність ключових сторінок (наприклад, /survey/start)	self.client.get(...) → статус 200

Приклад тесту функціональності:

```
def test_timer_is_applied(self):
    self.assertTrue(self.survey.has_timer)
    self.assertEqual(self.survey.timer_minutes, 10)
```

Цей тест перевіряє, чи коректно зберігаються таймерні налаштування при створенні опитування.

Переваги використання Django TestCase

- створення тимчасової бази для кожного запуску тестів;
- швидке тестування моделей, URL-маршрутів, форм і логіки обробки;
- автоматизація — можна інтегрувати у CI/CD (наприклад, через GitHub Actions);
- контроль змін — при редагуванні логіки видно, які тести "падають".

Запуск тестів

Для запуску всієї тестової логіки в межах додатку surveys використовується команда:

```
python manage.py test surveys
```

4.3. Приклади тестів

У системі UniSurvey реалізовано набір модульних і функціональних тестів, які охоплюють ключові компоненти платформи: таймер, обмеження повторного проходження, збереження відповідей, зв’язки між моделями та доступність основних представлень. Усі тести реалізовано з використанням класу django.test.TestCase, що дозволяє виконувати їх із використанням тестової бази даних.

Нижче наведено приклади тестів, які були створені під час розробки та підтвердили коректну роботу системи.

Тест 1. Перевірка обмеження повторного проходження

```
def test_limit_by_name_blocks_repeat_response(self):
```

```
    StudentResponse.objects.create(
        survey=self.survey,
        student_name="Іван",
        submitted_at=timezone.now()
    )
    exists = StudentResponse.objects.filter(
        survey=self.survey,
        student_name__iexact="Іван"
    ).exists()
    self.assertTrue(exists)
```

Що перевіряє:

Чи зберігається відповідь з конкретним іменем у базі та чи активується механізм блокування повторного проходження для того ж користувача.

Тест 2. Перевірка збереження параметрів таймера

```
def test_timer_is_applied(self):
    self.assertTrue(self.survey.has_timer)
    self.assertEqual(self.survey.timer_minutes, 10)
```

Що перевіряє:

Коректність збереження параметрів таймера, які вказані при створенні об’єкта опитування.

Тест 3. Збереження відповіді на питання

```
def test_answer_is_saved(self):
    response = StudentResponse.objects.create(
        survey=self.survey,
        student_name="Оля",
        submitted_at=timezone.now()
    )
    answer = response.answers.create(question=self.question)
```

```
    self.assertEqual(answer.question.text, "Який твій улюблений колір?")
```

Що перевіряє:

Чи створюється об’єкт Answer та чи встановлено зв’язок між відповіддю й питанням.

Тест 4. Доступність сторінки з опитуваннями

```
def test_survey_start_view_status(self):
    url = reverse("survey_start")
    response = self.client.get(url)
    self.assertEqual(response.status_code, 200)
```

Що перевіряє:

Чи успішно завантажується сторінка /survey/start/, яка відображає всі доступні опитування.

Тест 5. Перевірка кількості відповідей (експорт)

```
def test_export_data_csv_excel_structure(self):
    response = StudentResponse.objects.create(
        survey=self.survey,
        student_name="Оля",
        submitted_at=timezone.now()
    )
    response.answers.create(question=self.question)
    self.assertEqual(self.survey.responses.count(), 1)
```

Що перевіряє:

Чи зберігається відповідь в базі даних, що є основою для подальшого експорту результатів у формати CSV або Excel.

Тест 6. ManyToMany-зв’язок між Answer та AnswerOption

```
def test_selected_options_many_to_many(self):
    response = StudentResponse.objects.create(
        survey=self.survey,
        student_name="Андрій",
        submitted_at=timezone.now()
```

```
)
answer = response.answers.create(question=self.question)
answer.selected_options.add(self.option)
self.assertIn(self.option, answer.selected_options.all())
```

Що перевіряє:

Чи працює зв’язок ManyToMany — збереження обраних варіантів відповіді при питанні з кількома варіантами.

Ці тести охоплюють найбільш критичні аспекти роботи платформи та допомагають гарантувати її стабільність після внесення змін у код. Їх регулярне виконання дозволяє виявляти помилки на ранніх етапах розробки й підвищує якість готового продукту.

4.4. Аналіз результатів роботи системи

У результаті реалізації та тестування системи UniSurvey було підтверджено відповідність функціональних можливостей заявленим вимогам. Система забезпечує повний цикл проведення опитувань: від створення анкети до збереження результатів і подальшої аналітики.

Підтверджені можливості:

- Створення опитування та додавання питань з різними типами відповідей (текст, radio, checkbox);
- Встановлення таймера на проходження опитування;
- Блокування повторного проходження за іменем респондента;
- Збереження відповідей у базі даних із коректними зв'язками між сутностями;
- Експорт результатів у форматах CSV та Excel;
- Фільтрація і перегляд детальної статистики по кожному респонденту;
- Захищений доступ до адміністрування системи;
- Адаптивний і зручний для користувача інтерфейс.

Оцінка стабільності та надійності

Під час функціонального та модульного тестування не було виявлено критичних помилок, що свідчить про високу стабільність роботи платформи. Результати тестів свідчать, що:

- логіка перевірки повторного проходження працює коректно;
- таймер зупиняє можливість подальших дій після завершення часу;
- відповіді зберігаються як у текстовому, так і у варіантному форматах.

Оцінка продуктивності (обмеженою кількістю даних)

На тестовому сервері система демонструє стабільну роботу при 50+ проходженнях опитування, включно з одночасним збереженням, виведенням статистики та експортом даних. Результати вивантажуються без втрат або затримок.

Юзабіліті-аналіз

За результатами пробного використання:

- інтерфейс інтуїтивно зрозумілий;
- усі поля логічно згруповані;
- повідомлення про помилки/успіхи відображаються коректно;
- доступ із мобільних пристроїв забезпечений (адаптивна верстка).

Висновок

Система UniSurvey відповідає вимогам до програмного забезпечення для проведення онлайн-опитувань у навчальних закладах. Вона може бути легко адаптована до нових умов, масштабована, а також інтегрована в освітнє середовище. Тестування підтвердило її функціональність, зручність та стабільність роботи.

4.5. Порівняння з аналогами

Для оцінки ефективності реалізованої системи UniSurvey було проведено порівняння з популярними платформами для створення опитувань: Google Forms, SurveyMonkey та LimeSurvey. Критеріями порівняння обрано основні функціональні можливості, які мають безпосереднє значення для використання в закладах освіти.

Таблиця 4.2

Критерій / Система	**Google Forms**	**SurveyMonkey**	**LimeSurvey**	**UniSurvey**
Відкритий код	Ні	Ні	Так	Так
Можливість локального розгортання	Ні	Ні	Так	Так
Обмеження повторного проходження (за іменем)	Ні	Ні	Ні	Так
Таймер на проходження	Ні	Ні	Н	Так
Підтримка варіантів + відкритих питань	Ні	Ні	Так	Так
Експорт у CSV та Excel	Так	Так	Так	Так
Завантаження зовнішніх результатів	Ні	Ні	Ні	Так
Статистика з фільтрацією	Ні	Так	Так	Так
Темна тема, адаптивність	Ні	Ні	Ні	Так

Висновки порівняння

Система UniSurvey поєднує ключові переваги згаданих платформ і при цьому усуває низку обмежень:

- Відсутність потреби в комерційній ліцензії;
- Повна модифікованість коду;
- Підвищений контроль за проходженням опитувань;

- Підтримка додаткових функцій, таких як імпорт зовнішніх результатів, статистика, таймер і валідація користувача за іменем.

Таким чином, UniSurvey охоплює усі ключові функціональні потреби, які не забезпечують відомі сервіси без комерційної підписки чи складного налаштування. Особливою перевагою є відкритий код, можливість локального розгортання, імпорт зовнішніх результатів та вбудоване обмеження повторного проходження — саме ці функції є критично важливими для навчальних закладів.

ВИСНОВОК

У результаті виконання кваліфікаційної роботи було розроблено функціональну онлайн-платформу UniSurvey, призначену для проведення соціологічних опитувань у середовищі закладів вищої освіти. Під час реалізації поставленої мети було виконано повний цикл програмної розробки: від аналізу предметної галузі й вимог до системи — до тестування та апробації готового застосунку.

На першому етапі дослідження було проведено аналіз існуючих рішень для створення онлайн-опитувань (Google Forms, SurveyMonkey, LimeSurvey). Встановлено, що вони не повністю відповідають вимогам навчального середовища, зокрема: обмеженням повторного проходження, підтримці таймерів, адаптації під внутрішню інфраструктуру навчального закладу, імпорту та експорту результатів. Це обґрунтувало необхідність розробки власного веб-застосунку.

Було спроєктовано структуру системи, охоплено моделювання структури даних (ER-діаграма, UML-діаграма класів), описано логіку взаємодії користувачів (діаграма варіантів використання), а також визначено функціональні та нефункціональні вимоги до системи. Для реалізації обрано стек технологій: Python (мова програмування), Django (серверна логіка), PostgreSQL (база даних), HTML/CSS/JavaScript (інтерфейс).

Реалізовано зручний адміністративний інтерфейс для створення опитувань, додавання питань і варіантів відповідей. Особливу увагу приділено функціям контролю проходження: таймер на виконання, перевірка повторного проходження за іменем, збереження часу, облік балів. Результати зберігаються у реляційній базі даних, а платформа підтримує експорт у формати CSV та Excel.

Проведено функціональне тестування ключових модулів за допомогою Django TestCase. Зокрема, протестовано таймер, збереження відповідей,

обмеження за іменем та доступність основних сторінок. Тести підтвердили коректну роботу критично важливих компонентів.

Таким чином, у ході дипломної роботи:

- спроєктовано та реалізовано повноцінний програмний продукт для соціологічних опитувань;
- досягнуто гнучкості у налаштуванні типів опитувань;
- забезпечено стабільність і масштабованість архітектури;
- виконано тестування та перевірку відповідності функціоналу вимогам.

Результати дослідження можуть бути використані для подальшого впровадження платформи в освітньому середовищі, з можливістю її розширення (наприклад, інтеграції з LMS, аналітичними модулями тощо). Отримані знання й навички можуть бути основою для подальших наукових досліджень у сфері освітніх інформаційних технологій та електронної соціології.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Тимошенко М.О., Гаманюк І.М. Використання діаграми предметної галузі для створення моделі предметної галузі системи з опитування студентів: Матеріали Застосування програмного забезпечення в ІКТ, 15.04.25, ДУІКТ, м. Київ. С.135-137
1. Тимошенко М.О., Гаманюк І.М. Використання діаграми моделі прецедентів для моделювання функціональності системи з опитування студентів: Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.25, ДУІКТ, м. Київ. (подано до друку)

ПЕРЕЛІК ПОСИЛАНЬ

1. Міністерство освіти і науки України. Збираємо дані про якість освіти: запускаємо всеукраїнське опитування. 27 серпня 2024. URL: https://mon.gov.ua/news/zbyraiemo-dani-pro-iakist-osvity-zapuskaiemo-vseukrainske-opytuvannia (дата звернення: 25.05.2025).
2. Міністерство освіти і науки України. МОН запускає масштабне опитування школярів. 19 лютого 2025. URL: https://mon.gov.ua/news/mon-zapuskaie-masshtabne-opytuvannia-shkoliariv (дата звернення: 25.05.2025).
3. Міністерство освіти і науки України. Результати міжнародного дослідження якості освіти PISA-2022. 2023. URL: https://mon.gov.ua/news/rezultati-mizhnarodnogo-doslidzhennya-yakosti-osviti-pisa-2022 (дата звернення: 25.05.2025).
4. Міністерство освіти і науки України. Система оцінювання якості освіти в Україні: аналітичний звіт. Київ: МОН, 2022. 56 с.
5. Овчарук О. В., Іванюк І. В., Сороко Н. В., Кравчина О. Є., Гриценчук О. О. Результати онлайн-опитування «Потреби учителів у підвищенні фахового рівня з питань використання цифрових засобів та ІКТ в умовах карантину». Інститут інформаційних технологій і засобів навчання НАПН України, 2020. URL: https://lib.iitta.gov.ua/719908/ (дата звернення: 25.05.2025).
6. Приходько Т. О. Дослідження проблем молоді та освіти в умовах пандемії COVID-19. Соціологія та сучасні соціальні процеси. 2021. № 3. С. 45–56. URL: https://periodicals.karazin.ua/ssms/article/view/17692 (дата звернення: 25.05.2025).
7. Unified Modeling Language (UML) – Class Diagram. GeeksforGeeks. 2023. URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams (дата звернення: 25.05.2025).
8. DateTimeField – Django Models. GeeksforGeeks. 2023. URL: https://www.geeksforgeeks.org/datetimefield-django-models (дата звернення: 25.05.2025).

9. Functional and Nonfunctional Requirements Specification. AltexSoft. URL: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/ (дата звернення: 25.05.2025).
10. Django Software Foundation. Django ORM. Django Documentation. URL: https://docs.djangoproject.com/en/stable/topics/db/models/ (дата звернення: 25.05.2025).
11. Customizing the Django Admin. TestDriven.io. URL: https://testdriven.io/blog/customize-django-admin/ (дата звернення: 25.05.2025).
12. Django Software Foundation. The Django admin site. Django Documentation. URL: https://docs.djangoproject.com/en/5.2/ref/contrib/admin/ (дата звернення: 25.05.2025).
13. Openpyxl documentation. Reading and writing Excel (XLSX) files in Python. URL: https://openpyxl.readthedocs.io/en/stable/ (дата звернення: 25.05.2025).
14. Working With CSV Files in Python. Real Python. URL: https://realpython.com/python-csv/ (дата звернення: 25.05.2025).
15. Web Accessibility Initiative (WAI). Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 25.05.2025).
16. Django Software Foundation. Django documentation. The Django admin site. URL: https://docs.djangoproject.com/en/5.2/ref/contrib/admin/ (дата звернення: 25.05.2025).
17. Django Software Foundation. Testing in Django. Django Documentation. URL: https://docs.djangoproject.com/en/5.2/topics/testing/ (дата звернення: 25.05.2025).
18. Visual Paradigm**.** *UML Class Diagram Tutorial*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (дата звернення: 27.05.2025).
19. GeeksforGeeks**.** *DateTimeField – Django Models*. URL: https://www.geeksforgeeks.org/datetimefield-django-models/ (дата звернення: 27.05.2025).

20. **AltexSoft.** *Functional and Nonfunctional Requirements Specification*. URL: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/ (дата звернення: 27.05.2025).

21.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка онлайн-платформи для організації соціологічних опитувань студентів

Виконав студент 4 курсу
групи ПД-42
Тимошенко Микола Олексійович
Керівник роботи
старший викладач кафедри ІПЗ Гаманюк Ігор Михайлович

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення та покращення процесу збору та аналізу даних студентських опитувань

Об'єкт дослідження: процес організації та проведення опитувань серед студентів із використанням веб-сервісів для автоматизації збору даних

Предмет дослідження: розробка програмного забезпечення для автоматизації збору, обробки та аналізу даних у рамках системи опитувань для організаторів освітнього процесу

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Ознайомитися з сучасними методами проведення соціологічних опитувань серед студентів та їх автоматизацією.
2. Провести аналіз існуючих програмних рішень для організації соціологічних опитувань (Google Forms, LimeSurvey, SurveyMonkey) та визначити їхні недоліки.
3. Провести огляд та аналіз засобів реалізації веб-платформ для автоматизації збору відповідей, формування звітів та аналізу результатів опитувань.
4. Спроектувати архітектуру системи для автоматизації опитувань із можливістю додавання таймера, обмеження за іменем та формування звітів.
5. Реалізувати функціонал веб-платформи для організації соціологічних опитувань, включаючи створення опитувань, збір відповідей та виведення результатів.
6. Виконати тестування реалізованих функцій.

АНАЛІЗ АНАЛОГІВ

Критерії	Google Forms	SurveyMonkey	LimeSurvey	UniSurvey
Завантаження результатів стороннього опитування	-	-	-	+
Імпорт результатів з Excel/CSV	-	-	-	+
Формування звітів	-	+	-	+
Статистика	-	-	+	+
Автоматичне відображення результатів	+	-	-	+
Обмеження за іменем	-	-	-	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Створення опитувань.
2. Додавання зображень до питань.
3. Встановлення обмеження за іменем.
4. Проходження опитувань із таймером.
5. Збереження відповідей у базі даних.
6. Формування результатів з підрахунком балів.
7. Експорт даних у CSV та Excel.

Нефункціональні вимоги:

1. Основний функціонал доступний в 2 кліки
2. Безпека доступу до даних (Django authentication system)
3. Надійність зберігання відповідей (PostgreSQL)
4. Сумісність із браузерами (Microsoft Edge,Google Chrome,Firefox)

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Python

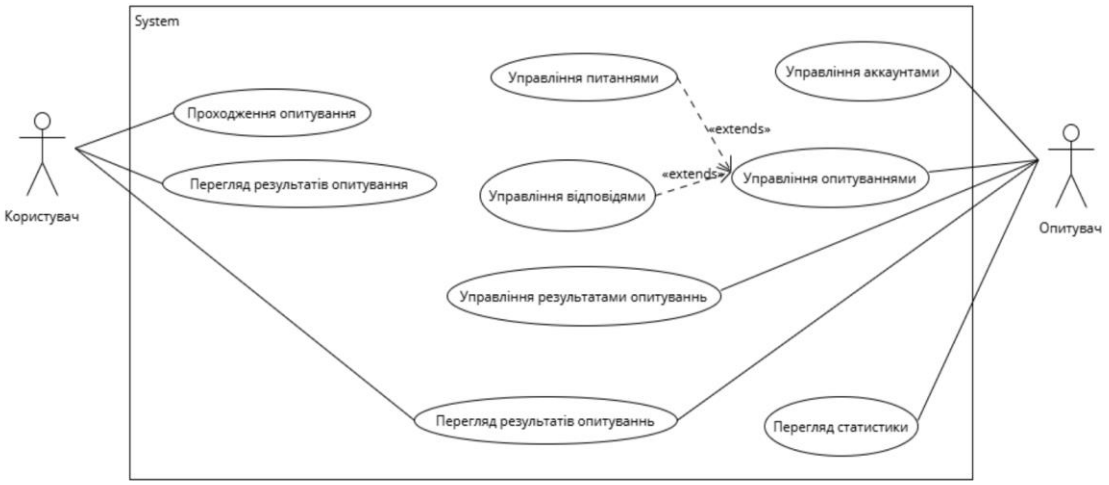
Django

Figma

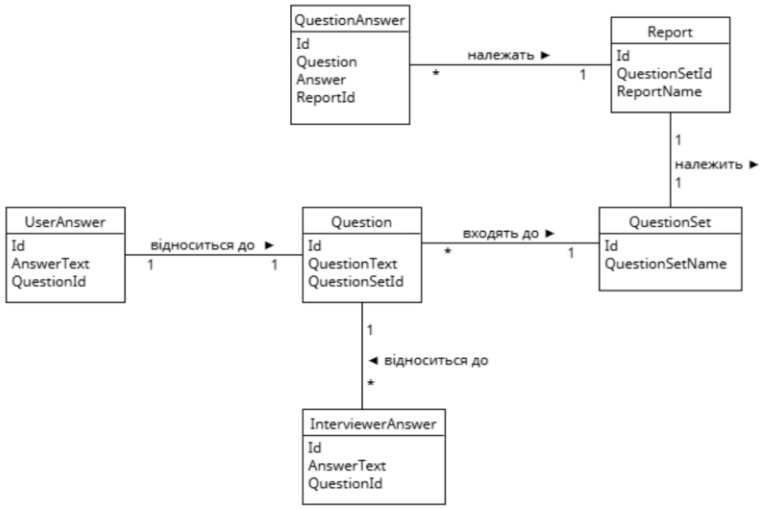
Visual Studio Code

PostgreSQL

Діаграма варіантів використання

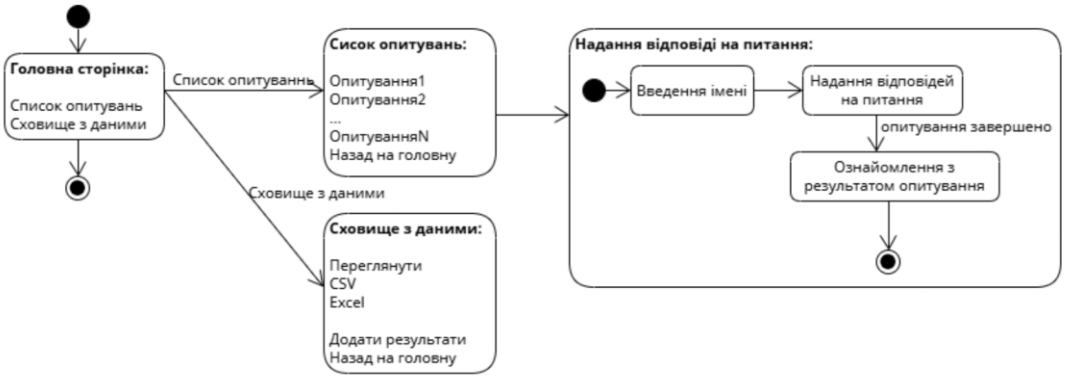


Діаграма предметної галузі

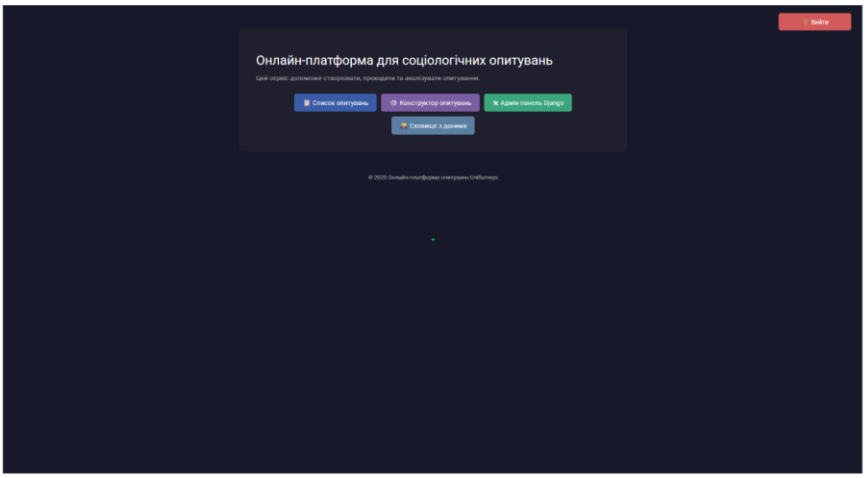


Діаграма діяльності

Діаграма Станів

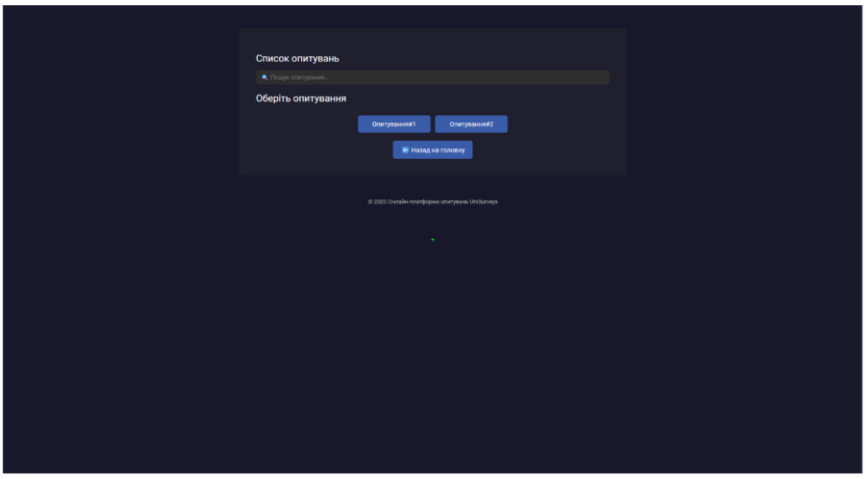


ЕКРАННІ ФОРМИ



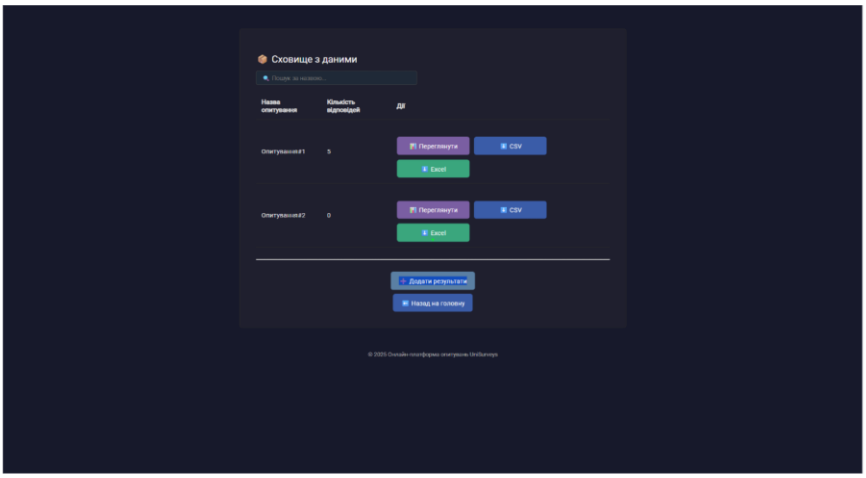
Головний екран додатку

ЕКРАННІ ФОРМИ



Список опитувань

ЕКРАННІ ФОРМИ



Сховище з даними

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Тимошенко М.О., Гаманюк І.М. Використання діаграми предметної галузі для створення моделі предметної галузі системи з опитування студентів: Матеріали Застосування програмного забезпечення в ІКТ, 15.04.25, ДУІКТ, м. Київ. С.135-137

2. Тимошенко М.О., Гаманюк І.М. Використання діаграми моделі прецедентів для моделювання функціональності системи з опитування студентів: Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.25, ДУІКТ, м. Київ. (подано до друку)

ВИСНОВКИ

1. На основі аналізу джерел та сучасних підходів до організації опитувань було визначено актуальні методи збору соціологічної інформації серед студентів та обґрунтовано доцільність їх автоматизації.

2. Проведено порівняння функціоналу існуючих платформ (Google Forms, LimeSurvey, SurveyMonkey), що дозволило виявити їх обмеження та сформулювати вимоги до розробки власного веб-рішення.

3. Оцінено ефективність різних інструментів для реалізації систем збору відповідей, статистичного аналізу та формування звітності, що стало основою для вибору технологій розробки.

4. Спроєктовано структуру системи із врахуванням ключових функціональних компонентів: моделі опитування, таймер, обмеження за іменем, механізм збору відповідей, зв'язки між сутностями.

5. Реалізовано веб-платформу UniSurvey з можливістю створення, проходження та збереження результатів опитувань, а також управління ними через адміністративний інтерфейс.

6. Проведено тестування реалізованого функціоналу, що підтвердило працездатність системи, її відповідність сформульованим вимогам та готовність до практичного застосування у навчальному процесі.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Код з модулю models.py

```
from django.db import models

class Survey(models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    show_results = models.BooleanField(default=False)
    is_active = models.BooleanField(default=False)
    enable_timer = models.BooleanField(default=False)
    time_limit_minutes = models.PositiveIntegerField(null=True, blank=True)
    has_timer = models.BooleanField(default=False)
    timer_minutes = models.PositiveIntegerField(default=5)
    limit_by_name = models.BooleanField(default=False)

    def __str__(self):
        return self.title

class Question(models.Model):
    QUESTION_TYPES = [
        ("text", "Відкрите питання"),
        ("radio", "Один варіант"),
        ("checkbox", "Кілька варіантів"),
    ]

    survey = models.ForeignKey(
        Survey, on_delete=models.CASCADE, related_name="questions"
    )
    text = models.TextField()
    question_type = models.CharField(max_length=20, choices=QUESTION_TYPES)
    image = models.ImageField(
        upload_to="question_images/", blank=True, null=True
    )

    def __str__(self):
        return self.text

class AnswerOption(models.Model):
    question = models.ForeignKey(
        Question, on_delete=models.CASCADE, related_name="options"
    )
    text = models.CharField(max_length=255)
    is_correct = models.BooleanField(default=False)

    def __str__(self):
        return self.text

class StudentResponse(models.Model):
    survey = models.ForeignKey(Survey, on_delete=models.CASCADE, related_name="responses")
    student_name = models.CharField(max_length=255)
    submitted_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.student_name} - {self.survey.title}"

class Answer(models.Model):
    response = models.ForeignKey(
        StudentResponse, on_delete=models.CASCADE, related_name="answers"
    )
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    selected_options = models.ManyToManyField(AnswerOption, blank=True)
    text_answer = models.TextField(blank=True)

    def __str__(self):
        return f"Відповідь на: {self.question.text}"
```

Код з модулю views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import user_passes_test
from django.contrib.admin.views.decorators import staff_member_required
from django import forms
from django.db.models import Q
from django.http import HttpResponse
from django.contrib import messages
from django.utils import timezone
from collections import defaultdict

from django.shortcuts import render, get_object_or_404
from .models import Survey

import csv
import openpyxl

from .models import Survey, Question, AnswerOption, StudentResponse, Answer
from .forms import QuestionForm

from django.shortcuts import render, get_object_or_404

from django.db.models import Count

def index(request):
    return render(request, "surveys/index.html")

def survey_start(request):
    query = request.GET.get("q", "")
    surveys = Survey.objects.filter(is_active=True)

    if query:
        surveys = surveys.filter(title__icontains=query)

    return render(
        request, "surveys/survey_start.html", {"surveys": surveys, "query": query}
    )

from django.contrib import messages

def survey_detail(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    questions = survey.questions.all()

    if request.method == "POST":
        student_name = request.POST.get("student_name").strip()


        if survey.limit_by_name:
            already_exists = StudentResponse.objects.filter(
                survey=survey, student_name__iexact=student_name
            ).exists()

            if already_exists:
                messages.error(request, "⚠ Ви вже проходили це опитування.")
                return redirect("survey_detail", survey_id=survey.id)


        response = StudentResponse.objects.create(
            survey=survey, student_name=student_name
        )

        for question in questions:
            answer = Answer.objects.create(response=response, question=question)

            if question.question_type == "text":
                text = request.POST.get(f"question_{question.id}")
                answer.text_answer = text
                answer.save()
            else:
```

```
                selected_options = request.POST.getlist(f"question_{question.id}")
                answer.selected_options.set(selected_options)


        request.session["response_id"] = response.id

        return redirect("survey_result", survey_id=survey.id)

    return render(
        request,
        "surveys/survey_detail.html",
        {"survey": survey, "questions": questions},
    )

def survey_result(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    print("DEBUG: show_results =", survey.show_results)

    if not survey.show_results:
        return render(
            request, "surveys/survey_result.html", {"survey": survey, "show": False}
        )

    response_id = request.session.get("response_id")
    if not response_id:
        return redirect("survey_start")

    response = get_object_or_404(StudentResponse, id=response_id, survey=survey)
    answers = response.answers.all()

    score = 0
    total = 0
    for answer in answers:
        correct_options = answer.question.options.filter(is_correct=True)
        selected = answer.selected_options.all()
        if correct_options.exists():
            total += 1
            if set(correct_options) == set(selected):
                score += 1

    return render(
        request,
        "surveys/survey_result.html",
        {
            "survey": survey,
            "show": True,
            "score": score,
            "total": total,
            "answers": answers,
        },
    )

from django.db.models import Count

def data_storage(request):
    search_query = request.GET.get("search", "")

    if request.GET.get("export") == "csv":
        survey_id = request.GET.get("survey_id")
        if survey_id:
            survey = get_object_or_404(Survey, id=survey_id)
            responses = survey.responses.all()

            response = HttpResponse(content_type="text/csv")
            response["Content-Disposition"] = (
                f'attachment; filename="survey_{survey_id}_results.csv"'
            )
            writer = csv.writer(response)
            writer.writerow(["Ім’я", "Дата", "Бал", "Макс. бал"])
```

```
            for r in responses:
                answers = r.answers.all()
                score = 0
                total = 0
                for a in answers:
                    correct = a.question.options.filter(is_correct=True)
                    selected = a.selected_options.all()
                    if correct.exists():
                        total += 1
                        if set(correct) == set(selected):
                            score += 1
                writer.writerow(
                    [
                        r.student_name,
                        r.submitted_at.strftime("%Y-%m-%d %H:%M"),
                        score,
                        total,
                    ]
                )
            return response

    surveys = Survey.objects.annotate(response_count=Count("responses"))

    if search_query:
        surveys = surveys.filter(title__icontains=search_query)

    return render(
        request,
        "surveys/data_storage.html",
        {"surveys": surveys, "search_query": search_query},
    )

@user_passes_test(lambda u: u.is_staff)
def response_detail(request, response_id):
    response = get_object_or_404(StudentResponse, id=response_id)
    answers = response.answers.all()
    return render(
        request,
        "surveys/response_detail.html",
        {
            "response": response,
            "answers": answers,
        },
    )

@user_passes_test(lambda u: u.is_staff)
def export_excel(request):
    wb = openpyxl.Workbook()
    ws = wb.active
    ws.title = "Результати"
    ws.append(["Ім'я", "Опитування", "Питання", "Відповідь", "Дата"])

    all_responses = StudentResponse.objects.all().prefetch_related(
        "answers__selected_options", "answers__question", "survey"
    )

    for response in all_responses:
        for answer in response.answers.all():
            if answer.question.question_type == "text":
                value = answer.text_answer or ""
            else:
                selected = [opt.text for opt in answer.selected_options.all()]
                value = ", ".join(selected)

            ws.append(
                [
                    response.student_name,
                    response.survey.title,
                    answer.question.text,
                    value,
```

```
                response.submitted_at.strftime("%Y-%m-%d %H:%M"),
            ]
        )

    response = HttpResponse(
        content_type="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
    )
    response["Content-Disposition"] = "attachment; filename=results.xlsx"
    wb.save(response)
    return response

class SurveyForm(forms.ModelForm):
    class Meta:
        model = Survey
        fields = ["title", "description", "show_results"]

@staff_member_required
def manage_surveys(request):
    query = request.GET.get("q")
    surveys = Survey.objects.all()

    if query:
        surveys = surveys.filter(title__icontains=query)

    surveys = surveys.order_by("-id")
    return render(
        request, "surveys/manage_surveys.html", {"surveys": surveys, "query": query}
    )

def create_survey(request):
    if request.method == "POST":
        title = request.POST.get("title")
        description = request.POST.get("description", "")
        show_results = "show_results" in request.POST
        enable_timer = "enable_timer" in request.POST
        time_limit_minutes = request.POST.get("time_limit_minutes") or 1
        limit_by_name = "limit_by_name" in request.POST

        survey = Survey.objects.create(
            title=title,
            description=description,
            show_results=show_results,
            has_timer=enable_timer,
            timer_minutes=(int(time_limit_minutes) if enable_timer else 1),
            limit_by_name=limit_by_name,
        )

        return redirect("edit_survey", survey_id=survey.id)

    return render(request, "surveys/create_survey.html")

@staff_member_required
def edit_survey(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)

    questions = survey.questions.all()
    selected_type = None
    show_option_fields = False
    question_to_edit = None
    options_to_edit = []

    edit_id = request.GET.get("edit")
    if edit_id:
        try:
            question_to_edit = Question.objects.get(id=edit_id, survey=survey)
            selected_type = question_to_edit.question_type
            show_option_fields = selected_type in ["radio", "checkbox"]
            options_to_edit = question_to_edit.options.all()
        except Question.DoesNotExist:
            pass
```

```
    if request.method == "POST":
        q_id = request.POST.get("question_id")
        text = request.POST.get("text")
        q_type = request.POST.get("question_type")
        selected_type = q_type
        option_texts = request.POST.getlist("option_text[]")
        correct_flags = request.POST.getlist("option_correct[]")
        show_option_fields = q_type in ["radio", "checkbox"]
        image_file = request.FILES.get("image")

        if q_id:
            question = Question.objects.get(id=q_id, survey=survey)
            question.text = text
            question.question_type = q_type
            if image_file:
                question.image = image_file
            question.save()
            question.options.all().delete()
            messages.success(request, "✅ Питання оновлено")
        else:
            question = Question.objects.create(
                survey=survey,
                text=text,
                question_type=q_type,
                image=image_file,
            )
            messages.success(request, "✅ Питання створено")

        if q_type in ["radio", "checkbox"]:
            for i, opt_text in enumerate(option_texts):
                if opt_text.strip():
                    is_correct = str(i) in correct_flags
                    AnswerOption.objects.create(
                        question=question, text=opt_text.strip(), is_correct=is_correct
                    )

        return redirect("edit_survey", survey_id=survey.id)

    return render(
        request,
        "surveys/edit_survey.html",
        {
            "survey": survey,
            "questions": questions,
            "selected_type": selected_type,
            "show_option_fields": show_option_fields,
            "question_to_edit": question_to_edit,
            "options_to_edit": options_to_edit,
        },
    )

class QuestionForm(forms.ModelForm):
    class Meta:
        model = Question
        fields = ["text", "question_type"]

@staff_member_required
def add_options(request, question_id):
    question = get_object_or_404(Question, id=question_id)
    options = question.options.all()

    if request.method == "POST":
        text = request.POST.get("text")
        is_correct = request.POST.get("is_correct") == "on"
        if text:
            AnswerOption.objects.create(
                question=question, text=text, is_correct=is_correct
            )
            return redirect("add_options", question_id=question.id)
```

```
    return render(
        request, "surveys/add_options.html", {"question": question, "options": options}
    )

@staff_member_required
def review_survey(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    questions = survey.questions.all().prefetch_related("options")
    return render(
        request,
        "surveys/review_survey.html",
        {
            "survey": survey,
            "questions": questions,
        },
    )

@staff_member_required
def delete_survey(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    if request.method == "POST":
        survey.delete()
        messages.success(request, "🗑 Опитування видалено")
    return redirect("manage_surveys")

@staff_member_required
def toggle_survey(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    survey.is_active = not survey.is_active
    survey.save()

    if survey.is_active:
        messages.success(request, "✅ Опитування опубліковано")
    else:
        messages.error(request, "🚫 Опитування приховано")

    return redirect("manage_surveys")

@staff_member_required
def statistics_view(request):
    from .models import Survey, Question, StudentResponse

    total_surveys = Survey.objects.count()
    total_questions = Question.objects.count()
    total_responses = StudentResponse.objects.count()

    return render(
        request,
        "surveys/statistics.html",
        {
            "total_surveys": total_surveys,
            "total_questions": total_questions,
            "total_responses": total_responses,
        },
    )

from collections import defaultdict

@staff_member_required
def survey_statistics(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    search_query = request.GET.get("q", "")

    responses = StudentResponse.objects.filter(survey=survey)

    if search_query:
        responses = responses.filter(student_name__icontains=search_query)
```

```
    responses = responses.prefetch_related(
        "answers__question", "answers__selected_options"
    )

    total_responses = responses.count()
    total_questions = survey.questions.count()

    response_data = []

    for r in responses:
        correct = 0
        answers_detail = []

        for a in r.answers.all():
            correct_options = list(a.question.options.filter(is_correct=True))
            selected = list(a.selected_options.all())
            is_correct = (
                set(correct_options) == set(selected) if correct_options else True
            )
            if is_correct and correct_options:
                correct += 1

            answers_detail.append(
                {
                    "question": a.question.text,
                    "selected": [opt.text for opt in selected],
                    "correct": [opt.text for opt in correct_options],
                    "is_correct": is_correct,
                    "question_type": a.question.question_type,
                    "text_answer": a.text_answer,
                }
            )

        response_data.append(
            {
                "id": r.id,
                "student_name": r.student_name,
                "submitted_at": r.submitted_at,
                "score": correct,
                "total": total_questions,
                "answers": answers_detail,
            }
        )

    return render(
        request,
        "surveys/survey_statistics.html",
        {
            "survey": survey,
            "total_responses": total_responses,
            "total_questions": total_questions,
            "response_data": response_data,
            "search_query": search_query,
        },
    )

@staff_member_required
def survey_statistics_detail(request, response_id):
    response = get_object_or_404(StudentResponse, id=response_id)
    answers = response.answers.all().prefetch_related("question", "selected_options")

    return render(
        request,
        "surveys/survey_statistics_detail.html",
        {
            "response": response,
            "answers": answers,
        },
    )
```

```
@staff_member_required
def survey_storage_detail(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    responses = StudentResponse.objects.filter(survey=survey).prefetch_related(
        "answers__question", "answers__selected_options"
    )

    data = []
    for r in responses:
        score = 0
        total = 0
        for a in r.answers.all():
            correct_options = a.question.options.filter(is_correct=True)
            selected = a.selected_options.all()
            if correct_options.exists():
                total += 1
                if set(correct_options) == set(selected):
                    score += 1
        data.append(
            {
                "student_name": r.student_name,
                "submitted_at": r.submitted_at,
                "score": score,
                "total": total,
                "response_id": r.id,
            }
        )

    export = request.GET.get("export")
    if export == "csv":
        response = HttpResponse(content_type="text/csv")
        response["Content-Disposition"] = (
            f"attachment; filename={survey.title}_results.csv"
        )
        writer = csv.writer(response)
        writer.writerow(["Ім'я", "Бал", "Максимум", "Дата"])
        for row in data:
            writer.writerow(
                [row["student_name"], row["score"], row["total"], row["submitted_at"]]
            )
        return response

    if export == "excel":
        wb = openpyxl.Workbook()
        ws = wb.active
        ws.title = "Результати"
        ws.append(["Ім'я", "Бал", "Максимум", "Дата"])
        for row in data:
            ws.append(
                [
                    row["student_name"],
                    row["score"],
                    row["total"],
                    row["submitted_at"].strftime("%Y-%m-%d %H:%M"),
                ]
            )
        response = HttpResponse(
            content_type="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
        )
        response["Content-Disposition"] = (
            f"attachment; filename={survey.title}_results.xlsx"
        )
        wb.save(response)
        return response

    return render(
        request,
        "surveys/survey_storage_detail.html",
        {"survey": survey, "results": data},
    )
```

```
@staff_member_required
def survey_responses_view(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    responses = StudentResponse.objects.filter(survey=survey).prefetch_related(
        "answers__question", "answers__selected_options"
    )

    data = []
    for response in responses:
        score = 0
        total = 0
        for answer in response.answers.all():
            correct_opts = answer.question.options.filter(is_correct=True)
            selected = answer.selected_options.all()
            if correct_opts.exists():
                total += 1
                if set(correct_opts) == set(selected):
                    score += 1
        data.append(
            {
                "name": response.student_name,
                "score": score,
                "total": total,
                "submitted": response.submitted_at,
            }
        )

    return render(
        request, "surveys/survey_responses.html", {"survey": survey, "data": data}
    )

@staff_member_required
def export_survey_responses_csv(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    responses = StudentResponse.objects.filter(survey=survey).prefetch_related(
        "answers__question", "answers__selected_options"
    )

    response = HttpResponse(content_type="text/csv")
    response["Content-Disposition"] = (
        f'attachment; filename="{survey.title}_results.csv"'
    )

    writer = csv.writer(response)
    writer.writerow(["Ім’я", "Балів", "Максимум", "Дата"])

    for r in responses:
        score = 0
        total = 0
        for a in r.answers.all():
            correct = a.question.options.filter(is_correct=True)
            selected = a.selected_options.all()
            if correct.exists():
                total += 1
                if set(correct) == set(selected):
                    score += 1
        writer.writerow(
            [r.student_name, score, total, r.submitted_at.strftime("%Y-%m-%d %H:%M")]
        )

    return response

@staff_member_required
def survey_data_detail(request, survey_id):
    survey = get_object_or_404(Survey, id=survey_id)
    responses = StudentResponse.objects.filter(survey=survey).order_by("-submitted_at")

    data = []
    for response in responses:
```

```
        answers = response.answers.all()
        correct = 0
        total = 0
        for answer in answers:
            correct_opts = answer.question.options.filter(is_correct=True)
            if correct_opts.exists():
                total += 1
                if set(correct_opts) == set(answer.selected_options.all()):
                    correct += 1
        data.append(
            {
                "response": response,
                "name": response.student_name,
                "score": correct,
                "total": total,
                "date": response.submitted_at,
            }
        )

    return render(
        request,
        "surveys/survey_data_detail.html",
        {
            "survey": survey,
            "data": data,
        },
    )

@staff_member_required
def upload_results_view(request):
    from openpyxl import load_workbook
    import csv

    if request.method == "POST":
        survey_title = request.POST.get("survey_title")
        file = request.FILES.get("file")

        if not survey_title or not file:
            messages.error(request, "❗ Введіть назву опитування та виберіть файл.")
            return redirect("upload_results")

        survey, created = Survey.objects.get_or_create(title=survey_title)

        filename = file.name.lower()
        if filename.endswith(".csv"):
            decoded = file.read().decode("utf-8").splitlines()
            reader = csv.reader(decoded)
            next(reader)
            for row in reader:
                name, date_str, score, total = row[:4]
                StudentResponse.objects.create(
                    survey=survey,
                    student_name=name,
                    submitted_at=timezone.now(),
                )
        elif filename.endswith((".xlsx", ".xls")):
            wb = load_workbook(file)
            ws = wb.active
            rows = list(ws.iter_rows(min_row=2, values_only=True))
            for row in rows:
                name, survey_name, question_text, value, date_str = row[:5]
                StudentResponse.objects.create(
                    survey=survey,
                    student_name=name,
                    submitted_at=timezone.now(),
                )
        else:
            messages.error(request, "❗ Непідтримуваний формат файлу.")
            return redirect("upload_results")
```

```
            messages.success(request, "✅ Результати успішно імпортовано.")
            return redirect("data_storage")

        return render(request, "surveys/upload_results.html")

    def survey_results(request, survey_id):
        survey = get_object_or_404(Survey, id=survey_id)
        responses = StudentResponse.objects.filter(survey=survey)
        return render(
            request,
            "surveys/survey_results.html",
            {"survey": survey, "responses": responses},
        )

    @staff_member_required
    def save_and_publish(request, survey_id):
        messages.success(request, "✅ Опитування опубліковано!")
        return redirect("manage_surveys")

    from django.shortcuts import render, get_object_or_404
    from .models import Survey

    from django.shortcuts import render, redirect, get_object_or_404
    from django.contrib import messages
    from django.contrib.admin.views.decorators import staff_member_required
    from .models import Survey, Question, AnswerOption

    @staff_member_required
    def edit_question(request, survey_id, question_id):
        survey = get_object_or_404(Survey, id=survey_id)
        question = get_object_or_404(Question, id=question_id, survey=survey)
        options = question.options.all()

        if request.method == "POST":
            question.text = request.POST.get("text")
            question.question_type = request.POST.get("question_type")
            question.save()

            question.options.all().delete()
            option_texts = request.POST.getlist("option_text[]")
            correct_flags = request.POST.getlist("option_correct[]")

            for i, text in enumerate(option_texts):
                if text.strip():
                    is_correct = str(i) in correct_flags
                    AnswerOption.objects.create(
                        question=question, text=text.strip(), is_correct=is_correct
                    )

            messages.success(request, "✅ Питання оновлено")
            return redirect("edit_survey", survey_id=survey.id)

        return render(
            request,
            "surveys/edit_question.html",
            {"survey": survey, "question": question, "options": options},
        )
```