Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему:

ВЕБРЕСУРС ДЛЯ ОРГАНІЗАЦІЇ ТА ПРОВЕДЕННЯ ОПИТУВАНЬ В НАВЧАЛЬНОМУ ПРОЦЕСІ

Виконав:
здобувач IV курсу
групи ІПЗ-41
спеціальності 121 «Інженерія
програмного забезпечення»
Гурик Ярослав Миколайович

Науковий керівник:
к. т. н., доцент Шевцова Н. В.

Рівне – 2025

ЗМІСТ

ВСТУП

Актуальність дослідження. Опитування дозволяють дізнатися думку цільової аудиторії, особливо, в різного роду компаніях та організаціях, а також в освітніх закладах та державних установах. Проблема зручності їх проведення завжди залишалася актуальною з моменту її появи як такої. Тому з розвитком всесвітньої мережі Internet з'явилися відповідні платформи, що пропонують послуги створення таких опитувань за допомогою спеціально розроблених для цього платформ. В закладах освіти процес навчання стає значно ефективнішим з проведенням опитувань, адже це спрощує процес навчання та отримання зворотного зв'язку від учнів та студентів. Таким чином вони можуть анонімно висловити невдоволення, критику, а також побажання щодо покращення умов та організації освітнього процесу.

Для комерційних установ це підвищує ефективність реклами, а також покращує розробку нового продукту, адже після проведення опитування можна краще зрозуміти очікування, проблеми клієнта, поточний відгук про послугу або товар.

Метою кваліфікаційної роботи є розробка вебресурсу для автоматичного створення та проведення опитувань, що дозволить користувачам швидко та ефективно створювати, налаштовувати та поширювати опитування без спеціальних знань у сфері програмування. Цей вебресурс має підвищити зручність проведення опитувань в навчальному процесі, автоматизувати процес створення форм і збір відповідей для аналізу.

Для досягнення мети було поставлено такі завдання:

1. розгляд наявних рішень автоматизації процесу опитування, виявлення їх недоліків та переваг;
2. проведення аналізу технологій, які використовуються на платформах автоматизації опитування;
3. розробка архітектури вебресурсу, визначення основних інструментів та технологій імплементації;
4. розробка вебресурсу автоматичного створення форми для опитувань.

Об’єкт дослідження – вебресурси для створення та поширення опитувань, що забезпечують автоматизацію та полегшують взаємодію з користувачами через інтуїтивно зрозумілий інтерфейс.

Предмет дослідження – процес автоматизованого створення опитувальних форм в інтернеті, зокрема інтерактивні методи створення, редагування та поширення таких форм для збору даних.

Апробація і впровадження результатів дослідження. Основні теоретичні та практичні результати здійсненого дослідження в межах цієї кваліфікаційної роботи обговорювалися та доповідалися на XVIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених «Наука, освіта, суспільство очима молодих» (м. Рівне, 14 травня 2025р.). Тези доповіді було опубліковано в збірнику матеріалів XVIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених «Наука, освіта, суспільство очима молодих» на сайті РДГУ [36]. Також результати дослідження доповідалися на звітній конференції викладачів, аспірантів та здобувачів освіти РДГУ за 2024 рік (м. Рівне, 15 травня 2025 р.) .

Структура роботи. Кваліфікаційна робота складається з чотирьох розділів, висновків та списку використаних джерел. Об’єм цієї роботи займає 49 сторінок. Перший має назву «Аналіз проблематики проведення опитувань, платформ автоматизації створення опитувань», займає 8 сторінок. Цей розділ складається 2 підрозділів, які розкривають суть теми та розділу. В першому підрозділі досліджується проблематика проведення опитувань та способи її вирішення за допомогою автоматизованих систем, а також недоліки в цих системах, перевірені методи для їх усунення. Також коротко описано основні технології використовувані на платформах для створення і проведення опитування. В другому, підрозділі розглядаються три платформи автоматизації опитувань: SurveyMonkey, Qualtrics, Google Forms. В цьому розділі міститься 5 рисунків.

Другий розділ має назву «Функціонал та архітектура вебресурсу автоматичного створення опитувань». Цей розділ займає 14 сторінок та містить 3 підрозділи. Він також містить 18 рисунків . В першому розглядається інтерфейсу

користувача та ключові можливості продукту. В другому підрозділі міститься огляд інтерфейсу аутентифікації користувача. В третьому описано обрану архітектуру та основні технології для розробки продукту.

Третій розділ має назву «Програмна реалізація функціоналу вебресурсу». Цей розділ займає 17 сторінок та складається з трьох підрозділів. В першому підрозділі розглядається імплементація адаптивного дизайну вебплатформи. В другому описано обробку подій та даних на Frontend стороні. Третій підрозділ містить основні аспекти Backend імплементації. Цей розділ має 3 рисунки та 16 лістингів. В кінці роботи подано висновки та список використаних джерел.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ПРОВЕДЕННЯ ОПИТУВАНЬ, ПЛАТФОРМ АВТОМАТИЗАЦІЇ СТВОРЕННЯ ОПИТУВАНЬ

Онлайн опитування – це віртуальна анкета для збору інформації з метою подальшої обробки та аналізу.

Автоматизація опитувань стосується використання технологій та програмного забезпечення для створення, поширення та здійснення аналізу опитувань. Це забезпечує можливість запобігти ручному введенню та обробці даних, що економить час та зменшує кількість помилок. Завдяки автоматизації опитування є можливість створювати персоналізовані опитування, орієнтовані на цільову аудиторію та отримувати дані в реальному часі [3].

1.1. Проблематика опитування та її вирішення

Складність збору даних, це одна з головних проблем, які вирішує автоматизація. Суть проблеми полягає в тому, що проведення онлайн опитування вручну вимагає багато часу та зусиль. Опитування особисто кожного респондента за допомогою організації розсилки та збору відповідей, є неефективним та довгим процесом. Автоматизовані рішення пропонують розробку спеціалізованих платформ, які автоматизують розсилку, сортування, зберігання, обробку. Це забезпечує необхідну зручність.

Важливою проблемою, яка потребує вирішення шляхом автоматизації, це низький рівень участі що може бути зумовлено невмотивованістю або ж незручністю проходження опитування, що неодмінно відображається на репрезентативності результатів опитування. Для запобігання цьому системи з автоматизації можуть пропонувати різного роду нагороди, бонуси за проходження опитування. Це може бути реалізовано за допомогою використання програмного забезпечення з інтеграцією CRM системи або ж системи лояльності. Така система

спроможна відстежувати поведінку респондентів та відповідно реагувати в реальному часі, наприклад, надсилати неактивному користувачу нагадування.

Дуже складно вручну аналізувати та обробляти дані проведеного опитування. Зі збільшенням об’єму даних, суттєво збільшується час обробки та ймовірність помилковості аналізу вхідних даних. Платформи з автоматизації опитувань можуть генерувати звіти та аналітику в реальному часі, що дає можливість зменшити час реакції на будь-які аномалії [13].

Платформи для автоматизованого проведення опитувань вирішують питання збереження та захисту конфіденційних даних. Реалізація полягає у використанні хмарних сервісів з можливість резервного копіювання.

Автоматизовані системи також дають змогу здійснювати персоналізовані опитування. Це дає можливість запобігти надсиланню однакових питань для різних людей з різним контекстом, що забезпечує кращу репрезентативність відповідей. Наприклад, при неактуальності якогось питання для конкретного учасника, система автоматично приховує або замінює його. В такому випадку, якість даних покращується, а кількість витраченого часу зменшується.

Онлайн опитування забезпечують комфортні умови для респондентів, адже учасник може це робити у зручний час, коли є змога сконцентруватися та подумати над питаннями. Учасник більше не залежний від строго визначеного місця і часу проведення опитування.

Вебресурси з автоматизації опитувань допомагають значно здешевити організацію їх проведення та подальшу обробку результатів. Це дуже актуально для освітніх закладів, адже вони часто опиняються в умовах недостатнього фінансування. Також економія завжди актуальна для будь-якого бізнесу, адже це означає, що за менші гроші результат, буде той самий, а можливо й кращий [6].

Крім цього, за допомогою автоматизації можна вирішити проблему масштабних опитувань, коли респондентів мільйони або десятки мільйонів. Такі масштаби змушують змінювати підхід і звертатись за допомогою до програмного забезпечення та хмарних технологій. Вони дозволяють легко охопити велику кількість респондентів по всьому світу не витрачаючи додаткових ресурсів. Хмарні сервіси дають велику

гнучкість для безперебійної роботи, за допомогою механізму балансування навантаження, навіть, при великій кількості одночасних учасників [13].

Багато вебресурсів такого типу надають повний або частковий доступ до функціоналу безкоштовно. У випадку повного безкоштовного функціоналу розробники можуть отримувати кошти шляхом показу реклами на сайті. В іншому випадку користувачу пропонують обмежений безкоштовний функціонал, який часто підходить лише для того, щоб зрозуміти чи варто купувати платну версію. У випадку з повністю платними платформами, вони зазвичай співпрацюють з бізнесом у якого є окремо виділений бюджет на це [1, 11, 12].

Автоматизація опитувань має не тільки переваги, але й недоліки з якими може стикнутися будь-яка система.

Конфіденційність даних і відповідність. Конфіденційність особистих даних користувачів стає все ретельніше захищена відповідними законами. Тому актуальність навігації в правовому полі цього питання для компаній та організацій стає все більшою. Їм часто приходиться працювати в декількох регіонах світу. Кожен з регіонів має свої правила захисту конфіденційності, наприклад GDRP, CCPA у Європі та Каліфорнії. Забезпечити відповідність вебресурсу з автоматизації опитувань таким законам є нетривіальним завданням, адже система має відповідати вимогам всіх регіонів де вона працює. Щоб уникнути потенційних проблем із законодавством важливо мати надійне шифрування інформації про користувачів, її анонімізацію та чітку політику щодо забезпечення конфіденційності даних.

Боротьба з втомою від опитувань. Часте запрошення пройти опитування може викликати невдоволення і втому. Це є серйозною проблемою, адже може відлякувати клієнтів, знижувати ефективність працівників компанії та погіршувати релевантність відповідей. Для того щоб усунути цю проблему варто розробити систему, яка автоматично плануватиме опитування для кожного користувача окремо, на основі його поведінки під час проходження попередніх опитувань. За допомогою такої стратегії можна досягнути максимальної зручності часу, що убереже від втоми та ігнорування.

Баланс автоматизації з особистим підходом. Хоча автоматизація є потужним інструментом для кращої ефективності, вона може виглядати безособовою, це може погано вплинути на стосунки з клієнтами. Рішення полягає в персоналізації на основі останніх взаємодій клієнта з компанією.

Проблематика індивідуального підходу. Для великих транснаціональних корпорацій працювати в різних культурних регіонах непросто. Виклик стає не меншим навіть при впровадженні автоматизації, адже тут необхідно щось більше аніж просто перекласти текст. Перш за все необхідне глибоке розуміння культурного і мовного контексту, що допоможе отримати якомога точніші і якісніші результати. Також такий підхід посприяє більшому комфорту для користувачів, які проходять опитування. Тож рішенням проблеми є адаптація системи до кожного регіону окремо [2].

1.2. Огляд існуючих платформ для проведення опитувань

Для повного розуміння проблематики та методів її рішення варто розглянути уже наявні реалізації. Нижче оглянуто три реалізації. Перша як одна з представників популярних серед звичайних користувачів платформ, а друга як така що представляє значно складнішу систему, якою користується скоріше великий бізнес ніж середньостатистичні користувачі. Третій сервіс це приклад повністю безкоштовної системи з відмінним базовим функціоналом.

SurveyMonkey. Платформа є однією з найпопулярніших онлайн-сервісів для опитувань. Вона підходить для користувачів для яких необхідний просунутий дизайн та розширені функції аналізу. Вебресурс розрахований на користувачів, які готові платити гроші за якісну аналітику, наприклад бізнес, дослідники або освітяни.

Переваги SurveyMonkey:

1. Великий вибір типів запитань: SurveyMonkey дозволяє створювати різноманітні типи запитань, в тому числі складні, наприклад, матричні. Це дає можливість отримати велику гнучкість і глибину опитувань.

2. Пропонує створення опитування як з чистого аркуша, з використанням шаблонів так і за допомогою штучного інтелекту (рисунок 1.1).

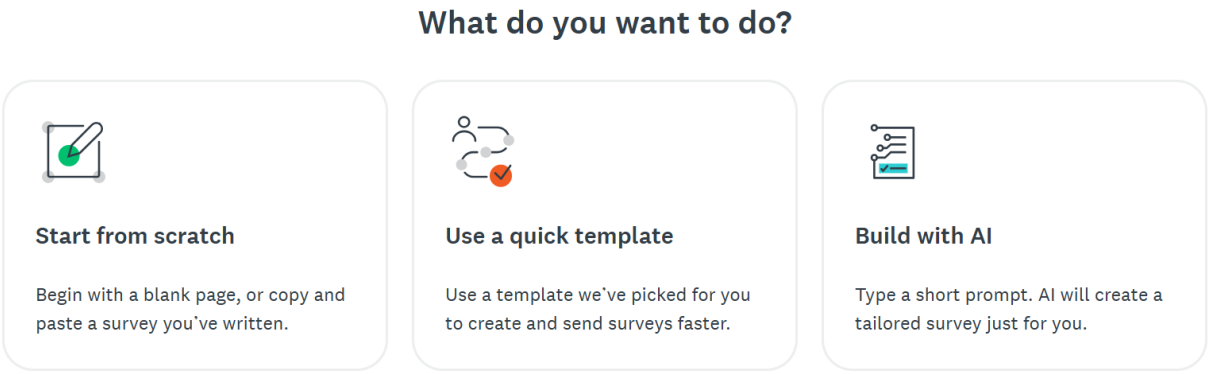

Рисунок 1.1. Запропоновані початкові варіанти створення опитувань

3. Розширена логіка опитування: має корисні функції: логіка пропуску, розгалуження, що дозволяє створювати опитування з складнішими та адаптивними шляхами на основі відповідей респондента.

4. Аналітичні інструменти: базові інструменти аналізу доступні навіть у безкоштовній версії, а платні версії надають доступ до розширеної аналітики, яка є часто незамінною для детальної інтерпретації даних (рисунок 1.2).

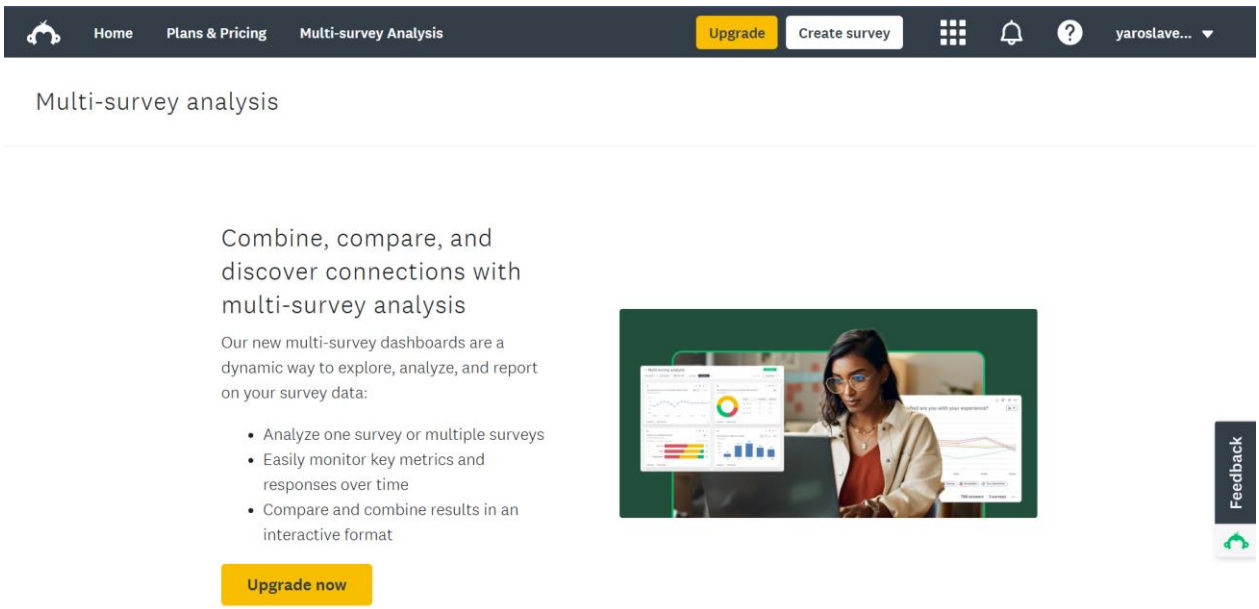

Рисунок 1.2. Інтерфейс розділу з аналізу опитувань для користувача без підписки

5. Налаштування та професійний дизайн: вебресурс надає різні варіанти налаштування дизайну, що підвищує рівень зацікавленості респондентів.

Недоліки SurveyMonkey:

1. Вартість: попри те, що є безкоштовна версія, платна містить багато розширеного функціоналу та аналітичних інструментів, що може бути перешкодою для користувачів невеликим бюджетом.

2. Обмеження: в безкоштовній версії наявні обмеження на кількість відповідей, що може перешкоджати проведенню великих опитувань [4, 5].

Qualtrics. Qualtrics — це доволі складна платформа для опитування, яка найбільше підходить для великих академічних, наукових, маркетингових або ж соціологічних досліджень.

Переваги Qualtrics:

1. Платформа має можливість налаштування складних дизайнів опитувань, таких як логічні шаблони, рандомізовані блоки та умовне розгалуження.

2. Qualtrics підтримує широкий вибір типів запитань, в тому числі спеціалізовані.

3. Доступні аналітичні можливості високого рівня з такими розширеними опціями як: статистичний аналіз, перехресне складання таблиць, інтелектуальне прогнозування. Такі характеристики надають ідеальні можливості для поглибленої інтерпретації даних (рисунок 1.3).

4. Вебресурс може бути інтегрований з іншими програмами та платформами.

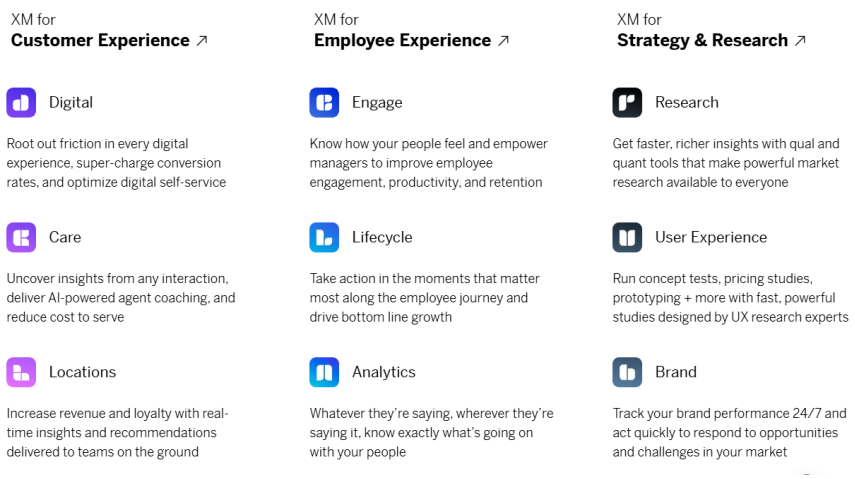

Рисунок 1.3. Перелік рішень для збору й аналізу даних на сайті Qualtrics

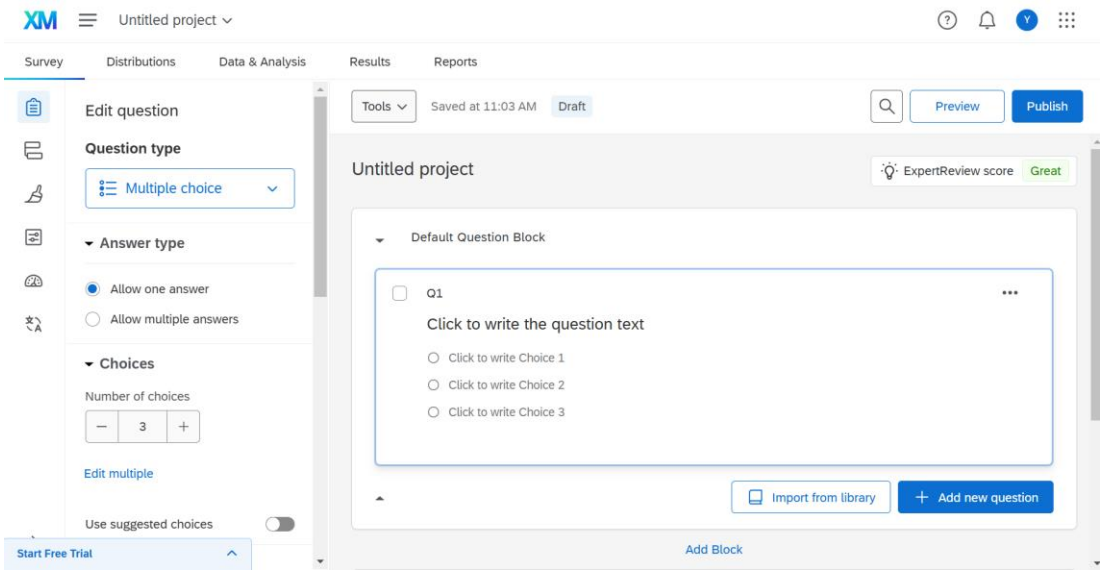

Рисунок 1.4. Робоча область для створення опитування користувача без підписки

Недоліки Qualtrics:

1. Qualtrics, зазвичай, дорожчий за більшість схожих інструментів. Це робить його набагато менш доступним для користувачів з обмеженим бюджетом.
2. Через величезний функціонал може бути складний в освоєнні для новачків.
3. Велика частина функціоналу може виявитись просто не потрібною для звичайних користувачів.
4. Безкоштовна версія дуже обмежена, що може відштовхувати частину користувачів [4].

Google Forms. Google форми це безкоштовний сервіс для автоматизації проведення опитування. Робота з Google акаунту, що дуже зручно. Можна створювати необмежене кількість опитувань, будь-якої довжини. Всього доступно одинадцять типів питань: текстові поля, списки та перемикачі що розкриваються, з однією відповіддю та з декількома і т. д. Опитування можна розбити на розділи, та зробити у вигляді тестування.

Інтерфейс. Ця платформа має адаптивний дизайн та можливість змінювати колірну тему (рисунок 1.5).

Статистика. Результати оновлюються в реальному часі, і для кожного питання генерується окремий графік або таблиця. Статистику можна отримувати через електронну пошту або експортувати у форматі CSV.

Інтеграція. Готове опитування можна відправити на пошту. Також є можливість відправити пряме посилання на нього або інтегрувати на сайт [4, 15].

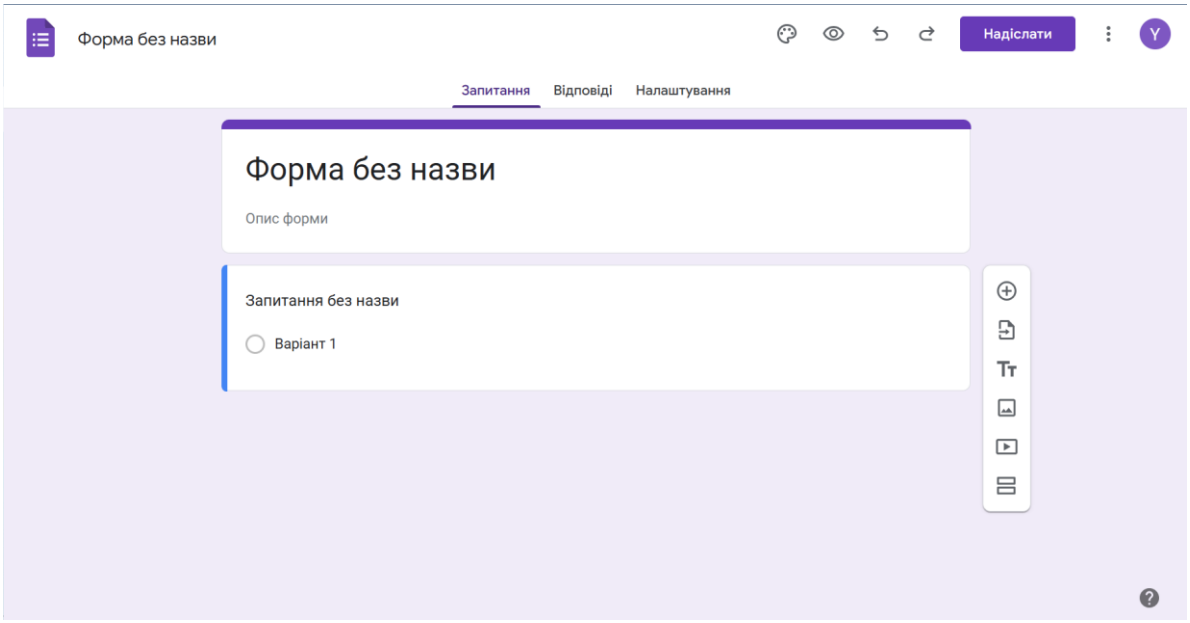

Рисунок 1.5. Інтерфейс платформи створення опитувань Google Forms

Google Forms є одним із найпопулярніших інструментів для проведення опитувань серед малого та середнього бізнесу. Однак, його обмежені можливості роблять його менш придатним для складних досліджень та професійного використання в галузях, де необхідний високий рівень аналітики [12, 14].

РОЗДІЛ 2
ФУНКЦІОНАЛ ТА АРХІТЕКТУРА ВЕБРЕСУРСУ АВТОМАТИЧНОГО СТВОРЕННЯ ОПИТУВАНЬ

2.1. Інтерфейс користувача та ключові можливості продукту

Платформа розроблена в межах кваліфікаційної роботи призначена для широкої аудиторії, адже вона вирішує задачу з якою може зіткнутися, в першу чергу, будь-який вчитель чи студент. Основний функціонал якраз полягає в тому щоб вирішити її, тобто спростити проведення опитувань, насамперед, шляхом автоматизації процесу створення опитування.

Як видно з рисунку 2.1 інтерфейс користувача достатньо простий для розуміння. Зверху розташоване меню, що дає змогу не загромаджувати інформацією основну сторінку. Мова інтерфейсу – англійська. На цьому етапі додано лише три пункти меню. Перший – «Main» відповідає за виведення основної сторінки (детальніше про реалізацію нижче у пункті 2.2). На цій сторінці спеціальний напис відразу запрошує користувача завантажити файл під яким розташована кнопка для завантаження. Важливо зазначити, що програма може обробляти файли тільки з розширенням .txt. З права від кнопки розташований напис який відображає поточний файл, який користувач завантажив або його відсутність.

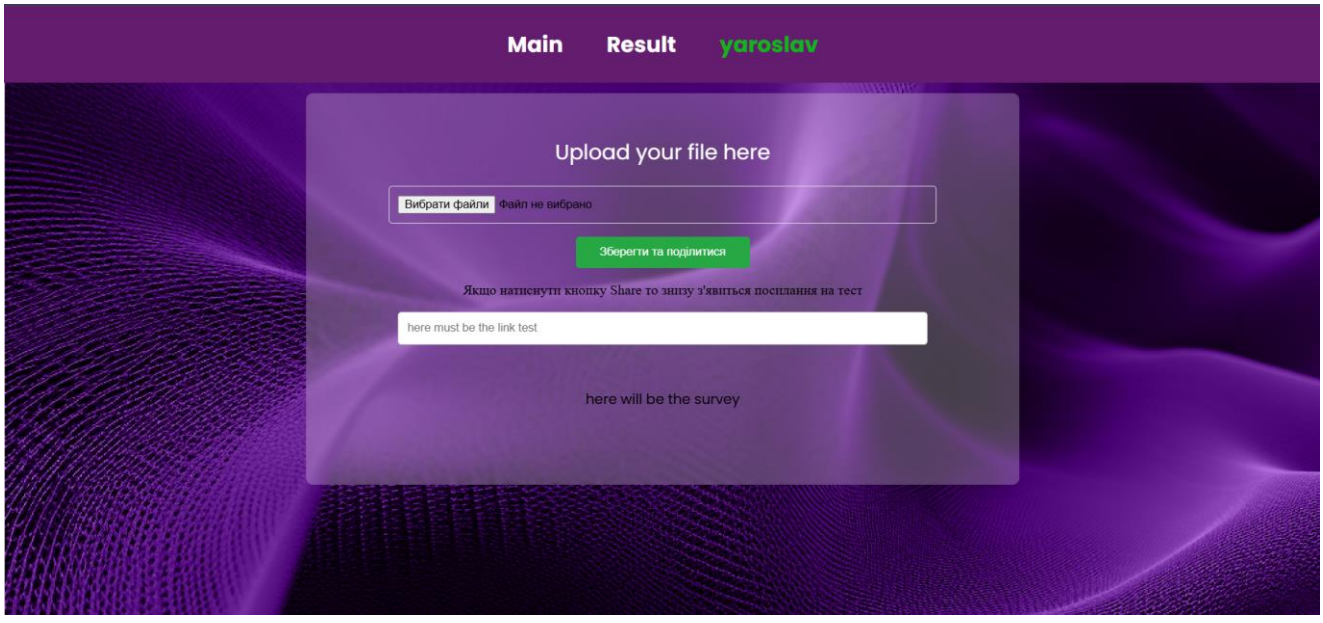

Рисунок 2.1. Початковий інтерфейс програми

Під цією частиною розташована кнопка яка відповідає за відправку змісту опитування в базу даних, а також для виклику функції генерації посилання, за допомогою якого можна поширити поточне опитування. Нижче, текстом вказано поле, де буде відображатися автоматично згенероване, інтерактивне опитування. Під ним розташована кнопка для відправки на подальшу обробку результатів пройденого опитування. Вона видима для користувача лише після збереження опитування та генерації посилання, а також у випадку, якщо користувач перейшов на опитування за посиланням (рисунок 2.2).

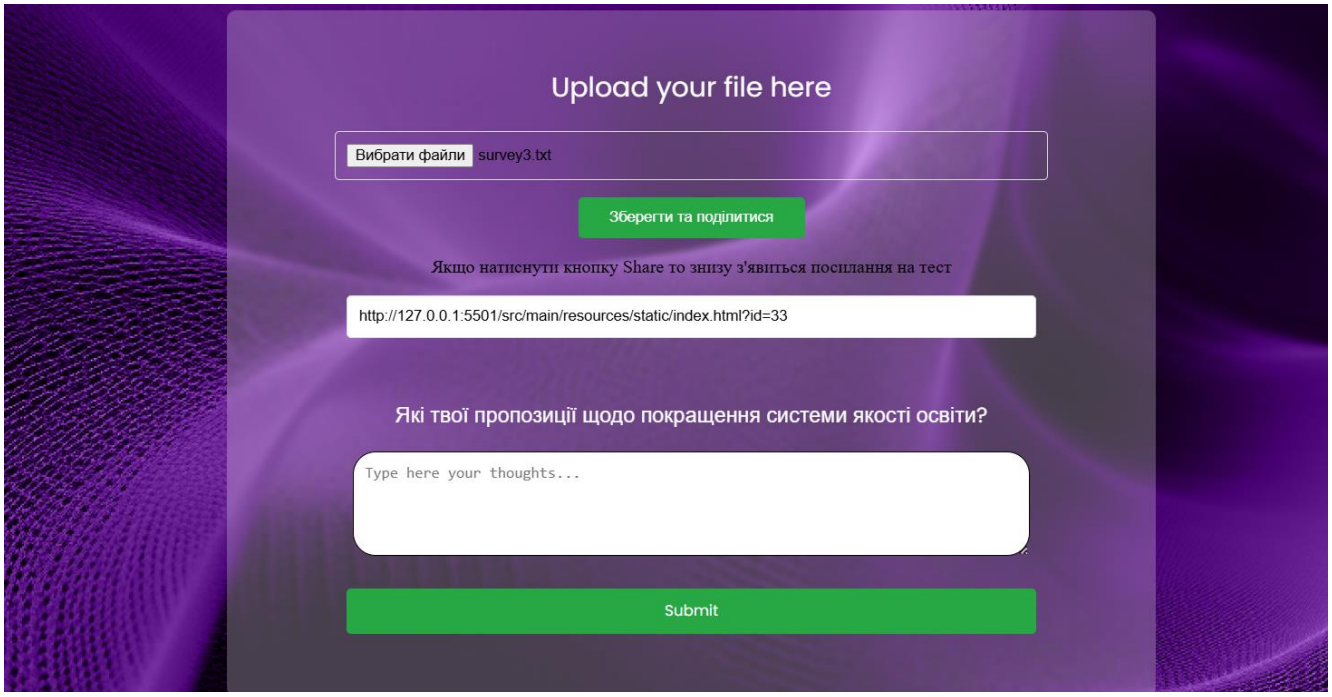

Рисунок 2.2. Стан інтерфейсу після натискання кнопки «Зберегти та поділитися»

При натисканні кнопки «Вибрати файли» відкривається вікно для вибору файлу (рисунок 2.3). Після того як користувач це зробив, програма автоматично створює інтерактивне опитування та відображає його у вказаному місці відповідно до вмісту файлу.

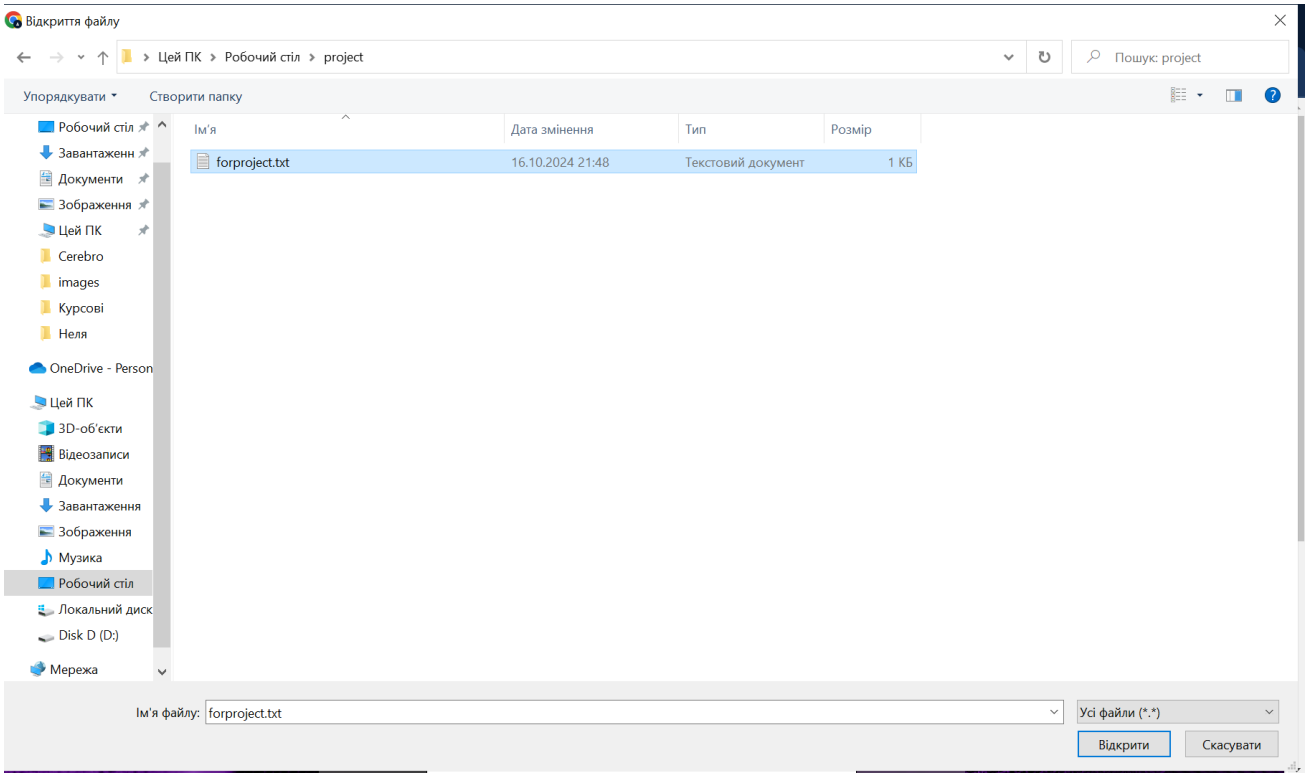

Рисунок 2.3. Вікно для вибору файлу з комп'ютера

Для коректного створення опитування необхідно дотримуватись правил написання вмісту опитування в текстовому файлі. Записувати питання треба з першого рядка і завершувати знаком питання. Після питання, писати відповіді слід без пропуску рядка.

Якщо необхідно мати можливість вибирати лише один варіант відповіді, то кожен варіант слід закінчувати символом крапки з комою (рисунок. 2.4). Результат створення по такому шаблону зображений на рисунку 2.5.

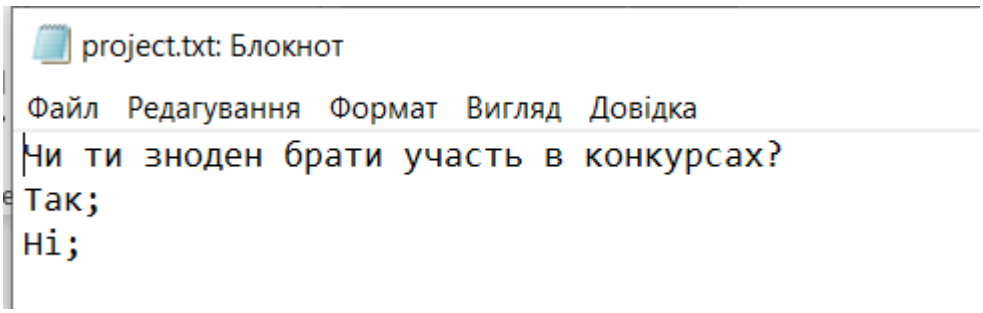

Рисунок 2.4. Приклад тексту для створення питання з одним варіантом відповіді

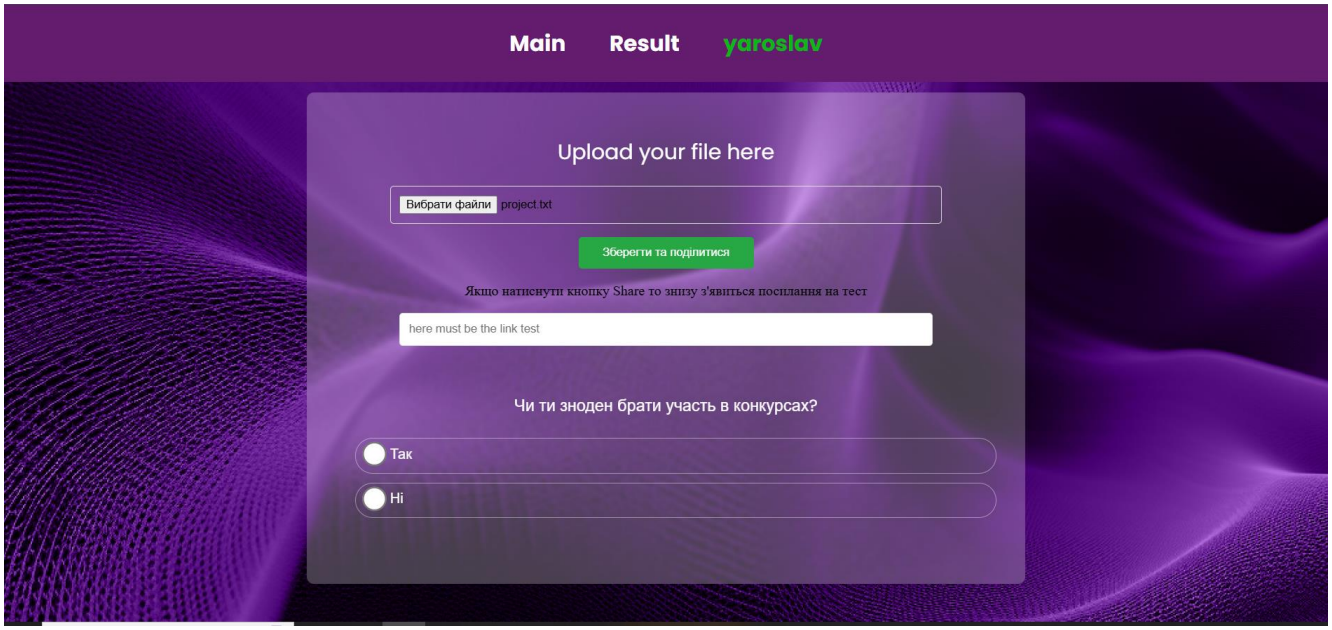

Рисунок 2.5. Створене опитування з одним варіантом відповіді

Для створення питання з можливістю вибрати декілька варіантів відповідей потрібно змінити початок питання з нумерації на знак «#» (рисунок 2.6). Всі інші правила зостаються такими самими.

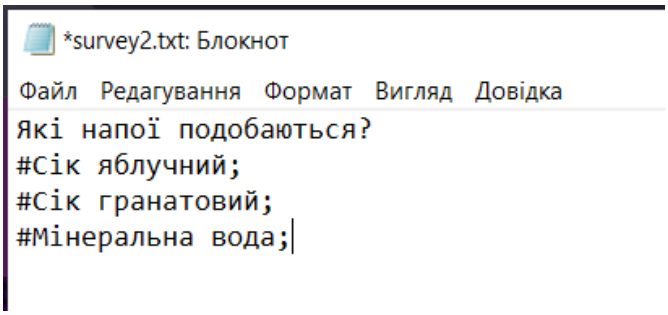

Рисунок 2.6. Приклад файлу для створення питання
з декількома варіантами відповідей

Як видно на рисунку 2.7, для вибору варіантів, програмою створений тип input - checkbox. Тепер користувач розуміє, що він може вибрати не один варіант.

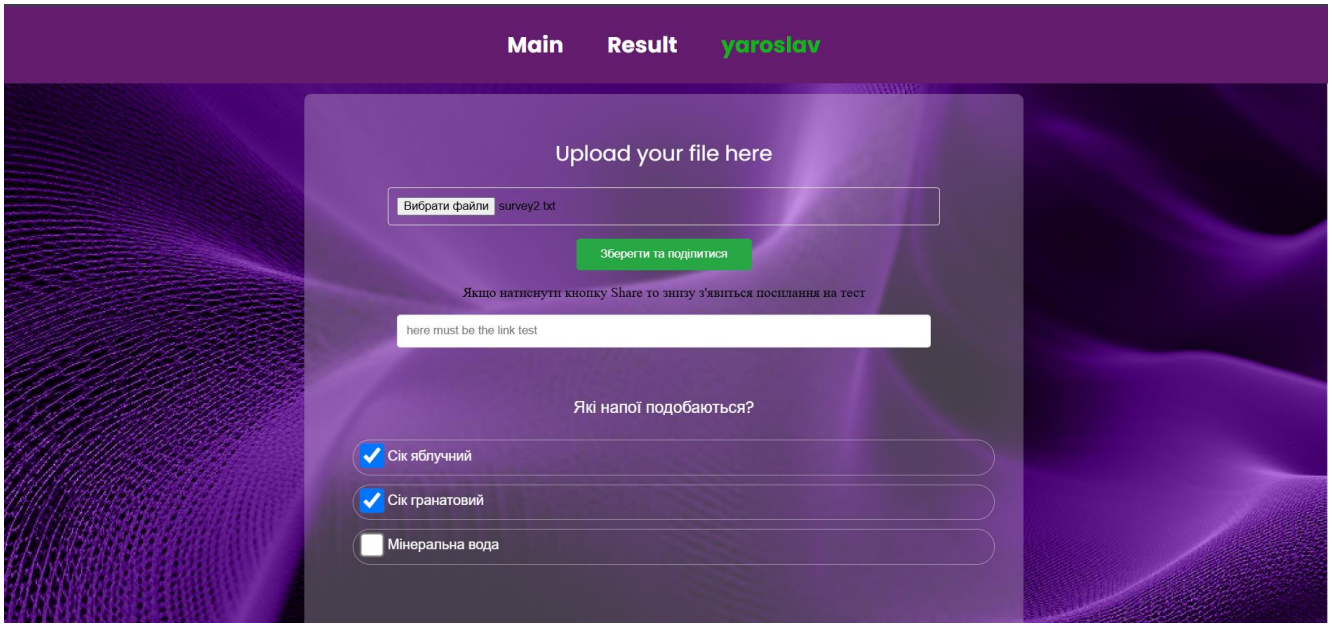

Рисунок 2.7. Створене опитування з декількома варіантами відповідей

На платформі передбачена можливість автоматичного створення опитування з відкритим питанням. Для цього в блокноті користувачу необхідно після питання прописати лінію, яка складатиметься більше ніж з одного знаку «_» (рисунок 2.8).

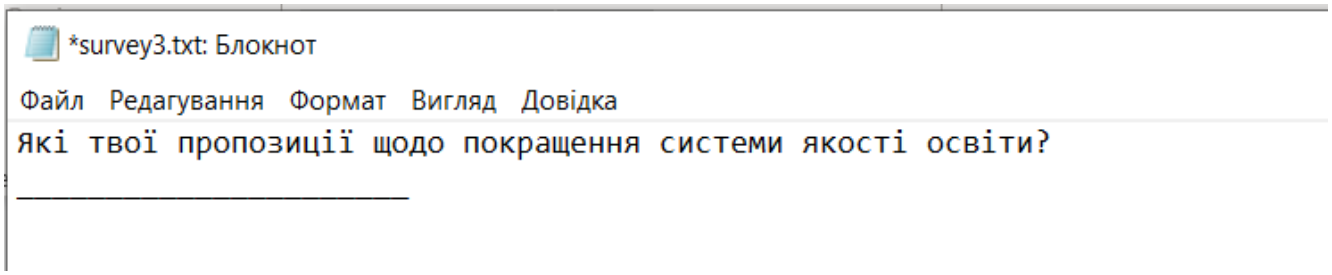

Рисунок 2.8. Приклад тексту для відкритого питання

Для того щоб користувачу легше було орієнтуватись куди вводити текст, в полі для введення вже є текст по замовчуванню (рисунок 2.9).

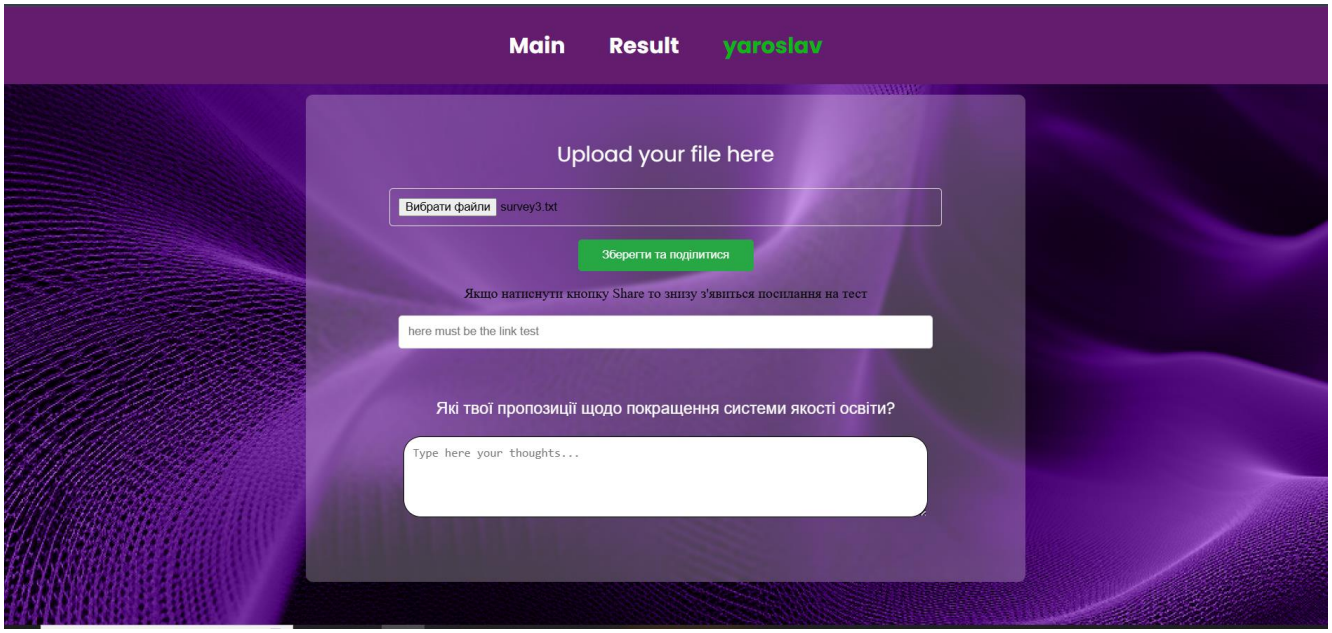

Рисунок 2.9. Створення відкритого питання

Крім окремих одинарних питань, користувач має змогу створювати складне опитування, яке може складатися з всіх доступних типів питань. В такому випадку, крім зазначених правил вище, потрібно між питаннями залишати пустий рядок (рисунок 2.10).

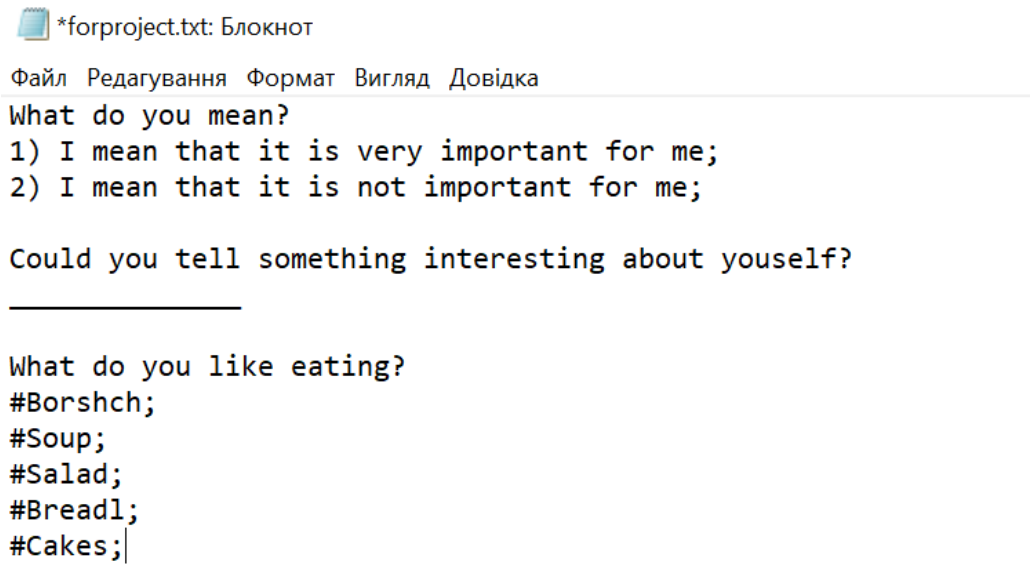

Рисунок 2.10. Приклад тексту для створення складного опитування

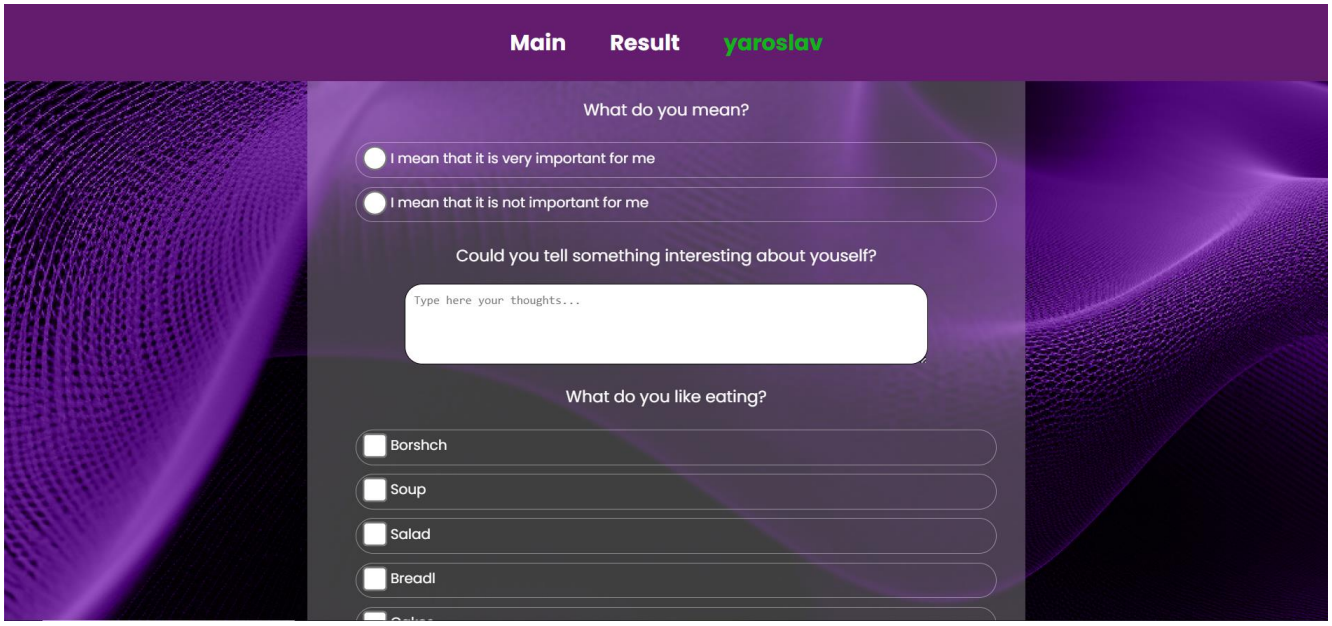

Рисунок 2.11. Автоматизоване створення опитування з декількома питаннями

Другий пункт меню – «Result» відповідає за виведення результатів опитування. По замовчуванню там розташований лише напис, що тут відображаються результати опитування, а нижче лише знак питання, що вказує на відсутність результатів проходження опитування (рисунок 2.12).

Проте після першого ж проходження користувачу відкривається це вікно, але вже з зазначеними ним відповідями (рисунок 2.13). Користувач, який пройшов опитування, при відкритті програми, спочатку бачить сторінку з власними результатами. Крім власних відповідей користувач також бачить статистику відповідей по кожному питанню цього опитування.

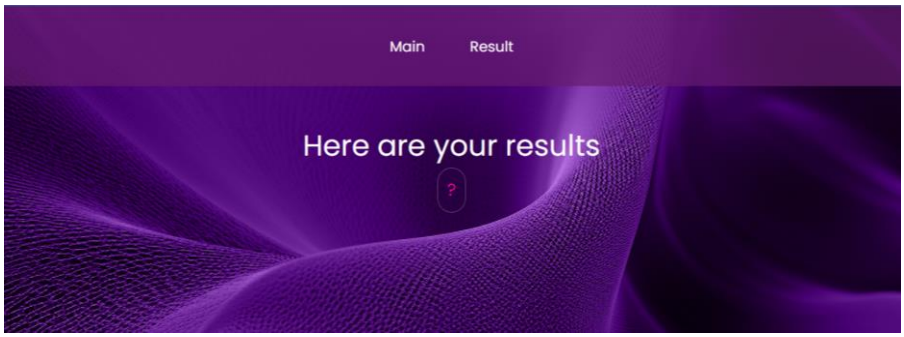

Рисунок 2.12. Інтерфейс сторінки результатів по замовчуванню

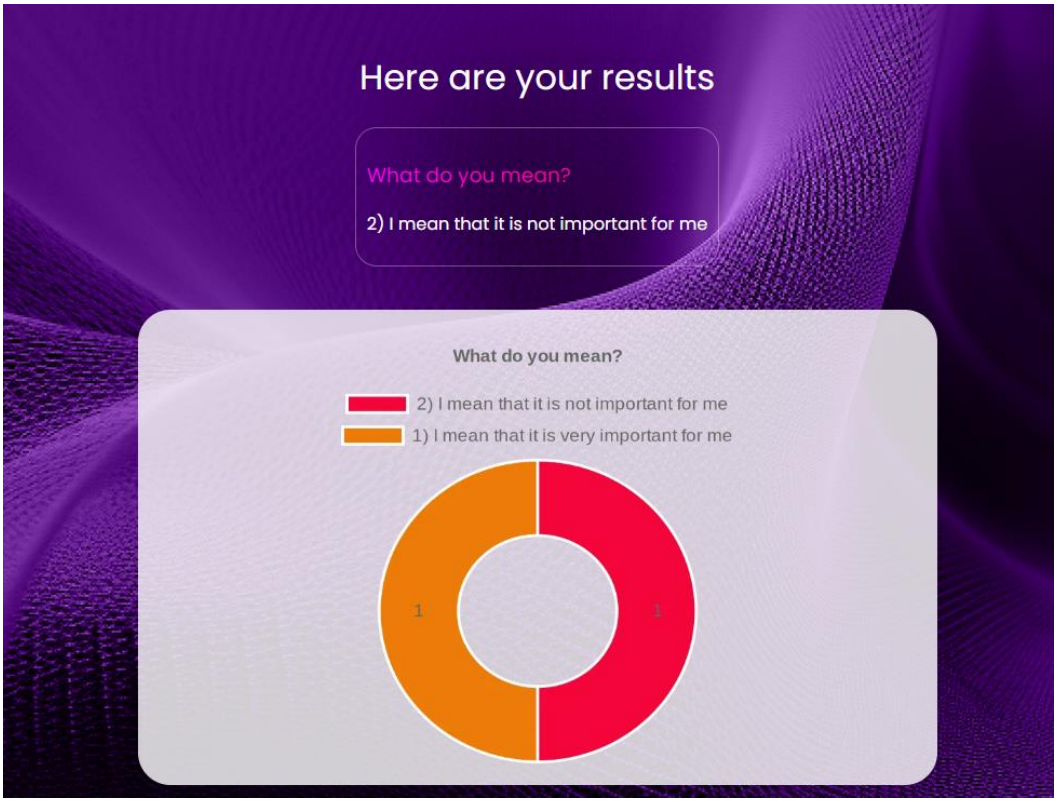

Рисунок 2.13. Виведення результатів опитування користувача

2.2. Система аутентифікації користувача

На початку, при вході, на цій платформі автоматично перевіряється наявність реєстрації користувача. Це відбувається за допомогою перевірки чи існує токен входу в сховищі браузера localStorage(). В такому разі користувач бачить дві кнопки як видно на рисунку 2.14. «SIGN IN» та «SIGN UP». Відповідно за ними користувач обравши свій варіант може або увійти в акаунт (якщо такий існує) або ж зареєструвати новий. При цьому всі інші пункти меню залишаються заблокованими. Це означає, що користувач не може користуватися сервісом без входу в систему.

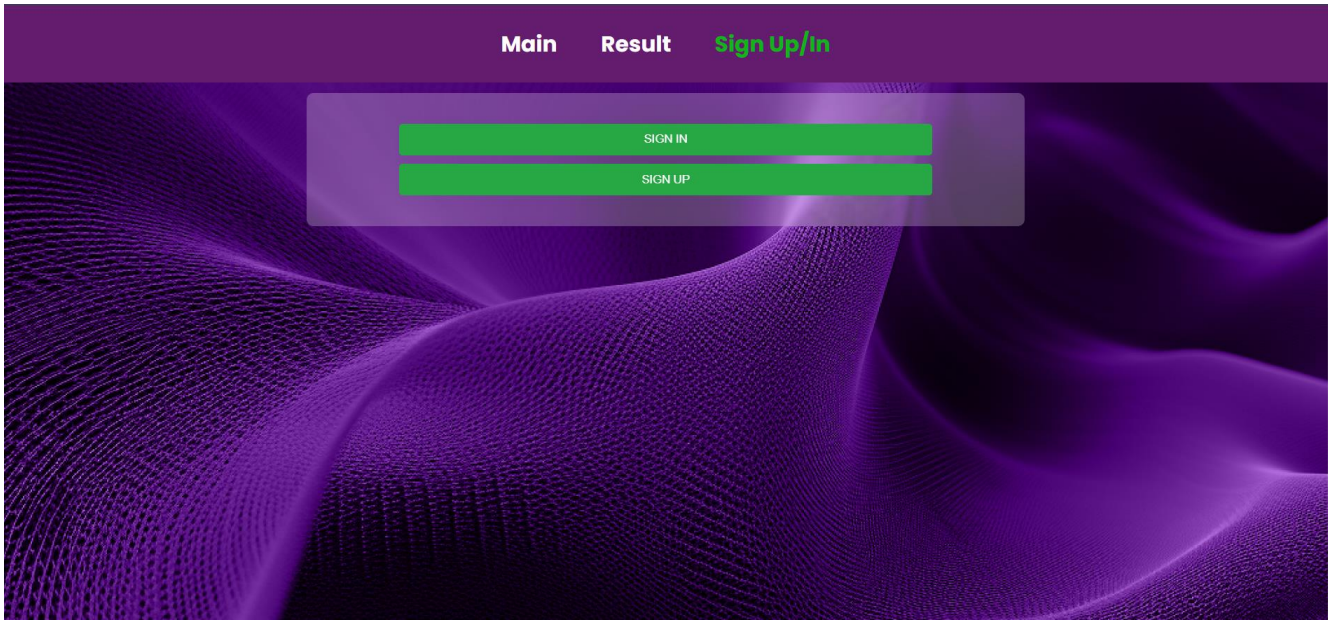

Рисунок 2.14. Розділ аутентифікації користувача

Для створення нового акаунту достатньо придумати ім’я користувача та пароль, при цьому ім’я користувача може не мати зовсім нічого спільно із справжнім ім’ям людини, яка реєструється (Рисунок 2.14). Також при збереженні даних реєстрації в базі даних, вони ніяк не пов’язуються з особистими даними особи за якими можна було б її ідентифікувати. Це дає можливість проходити опитування в анонімній формі без ризику розкриття особи.

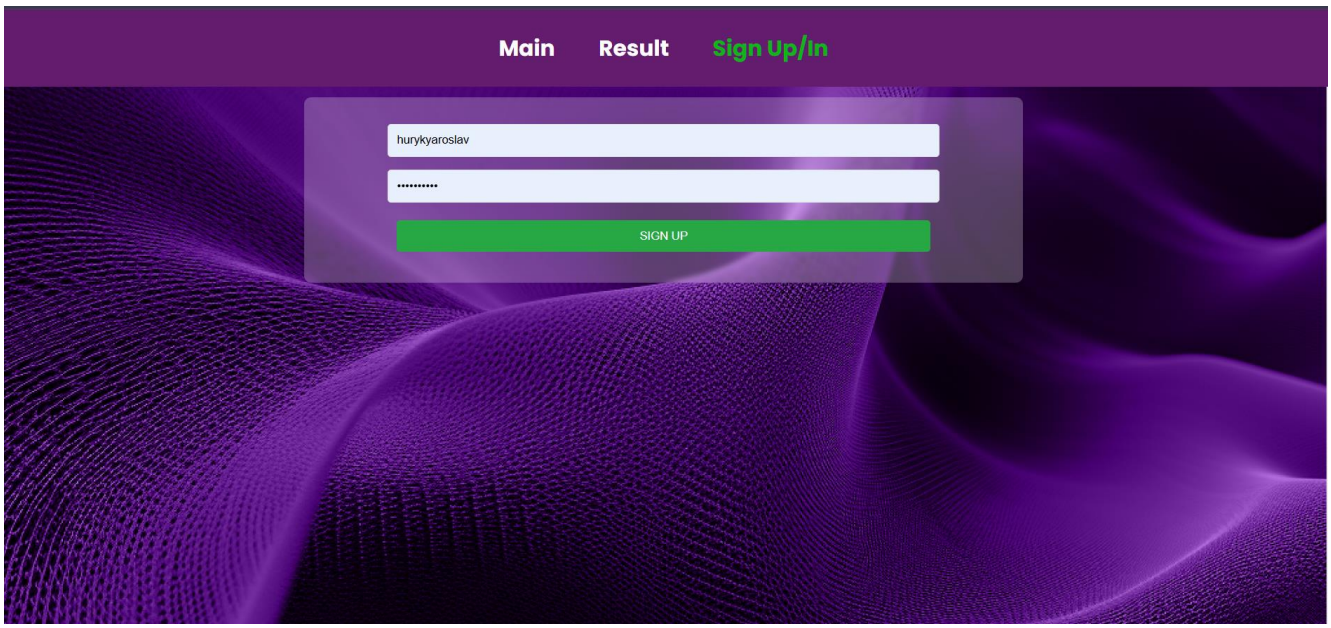

Рисунок 2.14 Форма реєстрації нового акаунту

Якщо є такий зареєстрований користувач при спробі створити такий самий акаунт відбувається обробка помилки у вигляді спливаючого вікна з відповідним повідомленням (Рисунок 2.15). В такому разі користувачу необхідно перейти за кнопкою «SIGN IN».

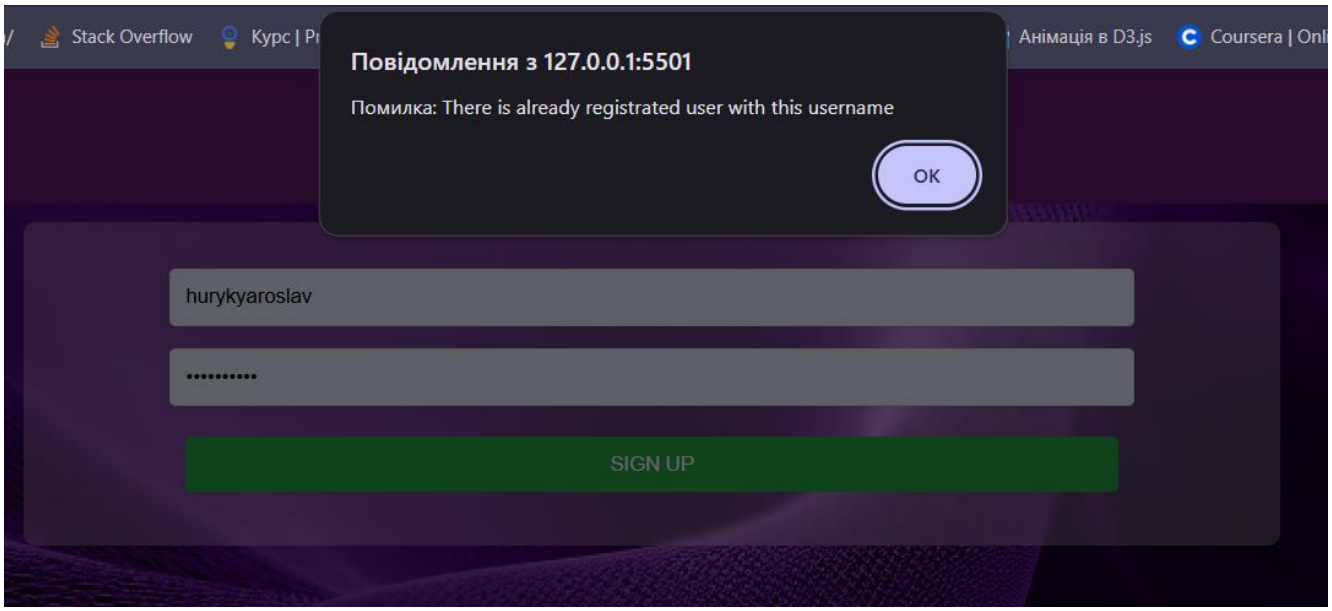

Рисунок 2.15. Обробка помилки реєстрації вже зареєстрованого акаунту

При успішній реєстрації або входу в акаунт користувач отримує повідомлення в такому ж самому вікні, але вже протилежного змісту на відміну від вищевказаного.

2.3. Архітектура та основні технології розробки

Ефективна платформи, яка взмозі вирішити актуальні проблеми використання вищезгаданих ресурсів для проведення опитувань, має бути в першу чергу із простим інтерфейсом, який забезпечуватиме достатню зручність при використанні та бути інтуїтивно зрозумілим. Для цього функціонал розбитий на три сторінки:

- Перша «Main» матиме основні можливості у вигляді автоматичного створення опитування шляхом завантаження файлу, а також збереження, генерація посилання для його поширення та можливість пройти опитування.

- Друга під назвою «Result» матиме функціонал відображення результатів пройденого опитування користувачем, а також відтворення аналітичних кругових діаграм для простоти сприйняття та зручності.
- Третя сторінка «Login/SignUp» має функціонал входу в акаунт, реєстрації нового акаунту, виходу з поточного акаунту.

Для кращої швидкодії реалізовано «умовні» сторінки. Тобто основні блоки сторінки створюються за допомогою функцій JavaScript. Це допомагає запобігти перезавантаженню всієї сторінки, що зменшує час відгуку.

В навчальних середовищах, таких як школи, університети та коледжі, діти не завжди мають доступ до особистого комп'ютера, тому критично важливо вирішити цю проблему. Найкраще рішення передбачає створення адаптивного дизайну. Це збільшить кількість студентів, які зможуть пройти опитування, а також знизить рівень можливого дискомфорту від цього процесу.

З метою полегшення створення опитування, реалізовано функціонал автоматизації цього процесу за допомогою зчитування інформації з документу.

Основні технологій та мови програмування використані для розробки frontend частини це: HTML, CSS, JavaScript.

Backend розроблений на об'єктно-орієнтованій мові програмування Java, вона має велику спільноту, а також підтримується й розробляється компанією Oracle [25].

При проектуванні серверної частини було обрано архітектурний стиль REST API (Representational State Transfer Application Programming Interface). Цей стиль найбільше підходить для реалізації поточної задачі, адже він доволі чіткий та простий для імплементації особливо якщо мова про такі невеликі проекти, як цей. При такому стилі взаємодія між frontend (клієнтська частина) та backend (серверна частина) відбувається за допомогою HTTP-запитів, основними методами яких є:

- GET – отримати дані,
- POST – створення нових даних,
- PUT – повне оновлення даних,

- PATCH – часткове оновлення даних,
- DELETE – видалення даних

Обмін даними відбувається у форматі JSON [26].

В REST API є такі основні принципи:

1. Незалежність від стану (stateless). Це означає, що кожен запит від клієнта повинен містити все необхідну інформацію для його успішного виконання. Кожен такий запит розглядається окремо від всіх інших.
2. Кешування даних (caching). В таку випадку система копіює ресурс на клієнтському або проміжному сервері для зменшення кількості запитів на сервер.
3. Єдиний інтерфейс (uniform interface). Це означає що існує певний уніфікований набір методів для доступу до ресурсів (GET, POST, PUT, PATCH, DELETE).
4. Система шарів (рисунок 2.16). Rest складається з шарів абстракції. Кожен компонент спілкується з сусідніми шарами над ним та під ним. Користувач не знає про кількість шарів в системі, взаємодіє тільки з одним із них. Наприклад клієнт не може записувати дані в базу даних напряму, він може це робити тільки через серверну частину [27, 28, 29].

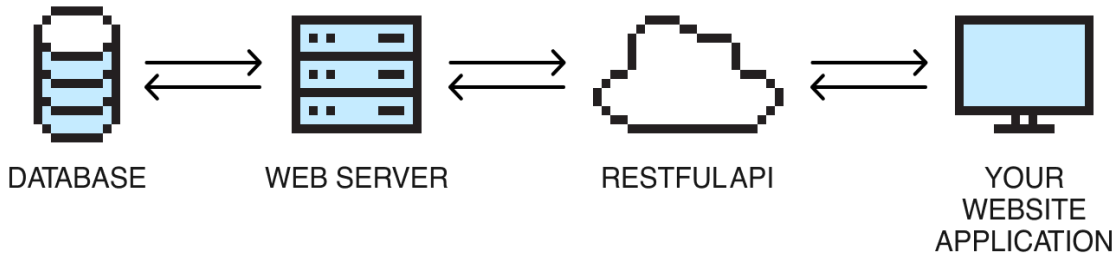

Рисунок 2.16. Система шарів RESTful [30]

Backend система спроектована таким чином, що має декілька шарів: Presentation Layer, Application Layer, Persistence Layer, Security Layer.

Для збереження відповідей на опитування, а також зареєстрованих користувачів та самих опитувань використовується реляційна база даних H2. Вона чудово підходить для тестування сервісу, тестових запусків та налагоджування проекту. Проте для кінцевого релізу варто перейти на кращі та надійніші інструменти, які взмозі забезпечити стабільність системи та технічну підтримку у вигляді якісної технічної документації чи ширшого ком'юніті (рисунок 2.17).

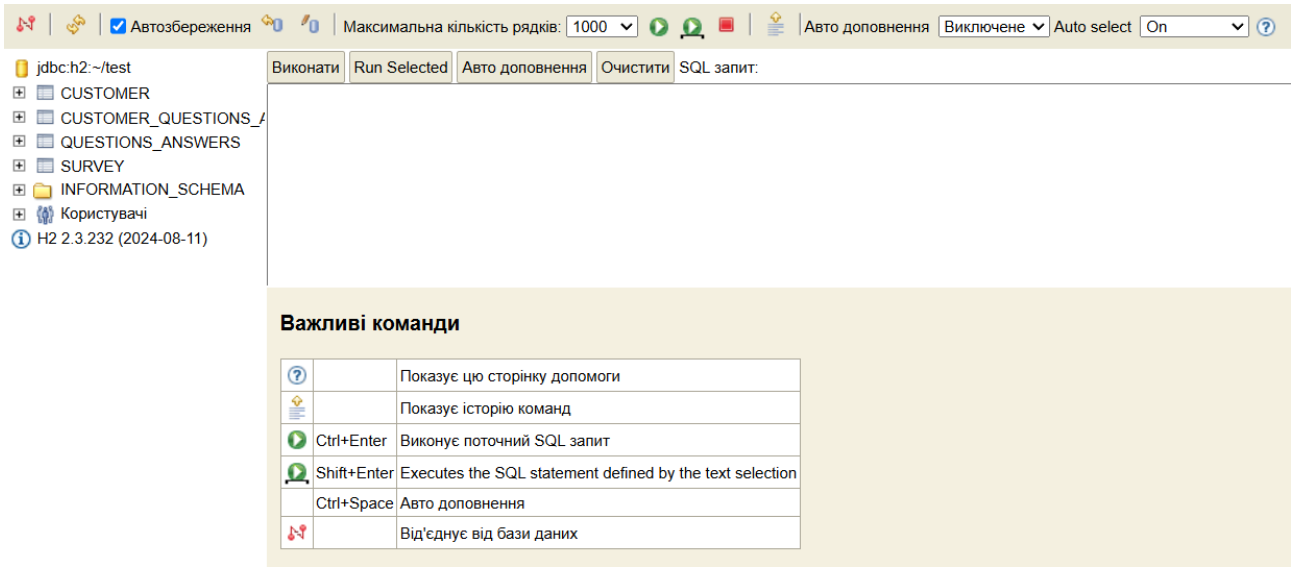

Рисунок 2.17. Інтерфейс бази даних H2.

В UML (Unified Modeling Language) діаграмі класів, поданої нижче, на рисунку 2.18 зображено основні класи, які відіграють найважливішу роль в реалізації функціоналу сервісу. Ця уніфікована мова моделювання використовується для створення моделі системи з використанням графічних позначень [34]. Це контролери, сервіси, сутності, репозиторії. Між ними зображено зв'язки, також в кожному класі зображено поля, типи даних цих полів. Крім цього, подано методи кожного класу, типи аргументів цих методів, а також тип даних які повертає метод. Це спрощує реалізацію програми, розуміння архітектури та взаємодій сутностей всередині системи. На зв'язках подекуди вказано тип зв'язку, наприклад, яку дію робить одна сутність над іншою. Дану діаграму зручно оновлювати та розширювати, при необхідності можна розділити на декілька менших.

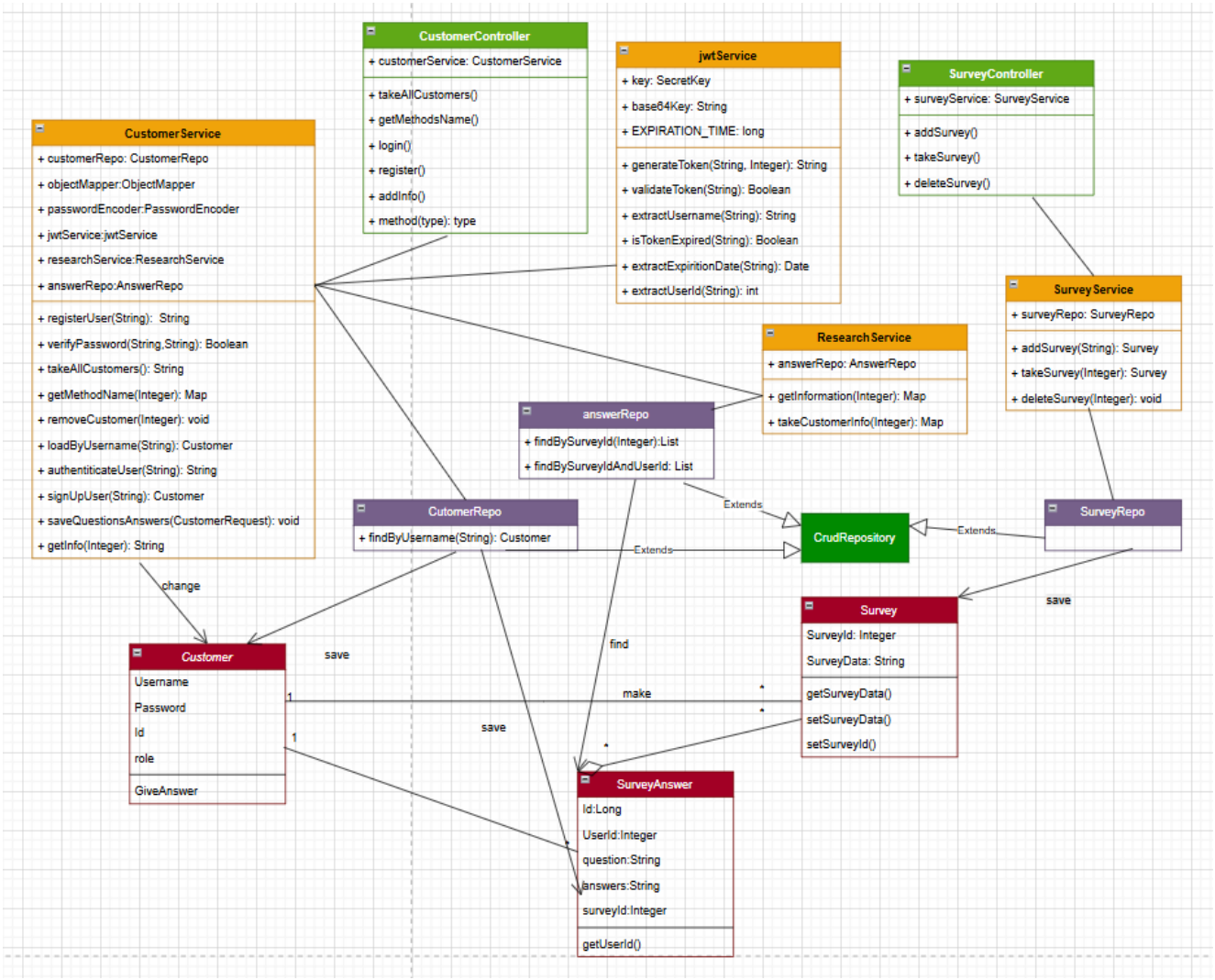

Рисунок 2.18. UML діаграма класів.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ ВЕБРЕСУРСУ

3.1. Імплементація адаптивного дизайну вебплатформи

Для реалізації відображення будь-якого тексту в цій програмі був використаний шрифт «Poppins». Для його підключення використано можливості хмарного сервісу Google Fonts де зібрано багато безкоштовних шрифтів у вільному доступі для розробників. З цією метою в середині <head> прописано відповідні теги:

Лістинг 3.1

```
<link rel="stylesheet" href="style.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Comfortaa:wght@300..700&fami
ly=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;
1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap"
rel="stylesheet">
```

Далі в файлі style.css створено необхідні класи для додавання цього шрифту. Меню сайту реалізовано з використанням тегу <header> для всієї шапки сайту та тегів маркованого списку <ul>, <li> для самих пунктів. Головний контент поміщений у <section> із заголовком <h1>. Для маніпуляцій з файлом використовується тег <input type=”file” />.

Лістинг 3.2

```
<form id="customerForm">
     <div class="test-container">
         <p class="phrase poppins-regular">here will be the survey</p>
     </div>
     <button  class="submit poppins-regular" type="submit">Submit</button>
</form>
```

Вище наведено частину коду з html формою в якій буде відображатиметься опитування згенероване за допомогою js при завантажені файлу. В ній також

міститься кнопка <button type=”submit”> для відправки відповідей по опитуванню. Нижче цієї форми знаходиться контейнер в якому динамічно з’являтимуться результати при натисканні кнопки «Submit» або при переході в розділ «Result». Перехід в меню з розділу на розділ відбувається на одній і тій же html сторінці. При натисканні кнопок один розділ приховується за допомогою додавання класу «.hidden» до елементу, а інший навпаки створюється. Головний розділ тільки приховується, а розділ з результатами видаляється і коли треба, генерується повторно.

Для гнучкого вирівнювання пунктів меню по центру, використана технологія flexbox. CSS Flexbox — це технологія для створення складних та гнучких макетів створення яких дуже складне за допомогою традиційних стилів css [7, 8, 20]. Ось як виглядає css код класу до якого застосовано цей підхід:

Лістинг 3.3

```
.menu {
    position: relative;
    max-width: 100%;
    height: 100%;
    background-color: #641c6e;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 0;
    margin: 0;
}
```

Властивість justify-content визначає розміщення по головній осі (main-axis). Тоді як align-items по поперечній (cross-axis). Елемент, до якого застосовано display: flex, називається flex container. Це завжди батьківський елемент в середині якого потрібно гнучко розташувати елементи. Дочірні елементи – це flex items [8].

Для того щоб користувачу було зрозуміло, що це пункти меню на які можна натискати, а не просто текст, до них додано псевдо-клас :hover. Він відстежує наведення курсора на пункт та змінює колір тексту (рис 3.1).

Рисунок 3.1. Результат роботи псевдо-класу :hover.

Гнучкі контейнери допомагають робити сайти більш адаптивними до різних розмірів екранів. Проте при великій різниці між розмірами цього часто недостатньо. В такому випадку використовуються медіа запити. Медіа висловлювання – це css правила, що дозволяють керувати стилями в залежності від характеристик пристрою на якому відкривається поточний вебсайт [9]. Для забезпечення необхідної адаптивності при відкритті програми на смартфоні, в цьому проекті також використано технологію медіа запитів (лістинг 3.4).

Спочатку написане ключове слово @media, а далі в дужках максимальна ширина екрану або вікна браузера для якої застосовуватимуться зміни, прописані в фігурних дужках нижче. В даному випадку, зміна ширини форми, де відображається опитування, на 90vw від ширини екрану. Та встановлення максимальної ширини цієї форми [18, 19].

Лістинг 3.4

```
@media (max-width:700px) {
    .form-container {
        width: 90%;
        max-width: 90vw;
        padding: 20px;
    }
}
```

Також реалізовано адаптивне меню сайту для мобільних пристроїв у вигляді бургер-меню. Для його реалізації використано псевдо-елементи, які використовуються для додавання вмісту на сторінку до елемента та після нього, а саме ::before, :: after. Скріншот адаптивного інтерфейсу та меню подано на рисунку 3.2.

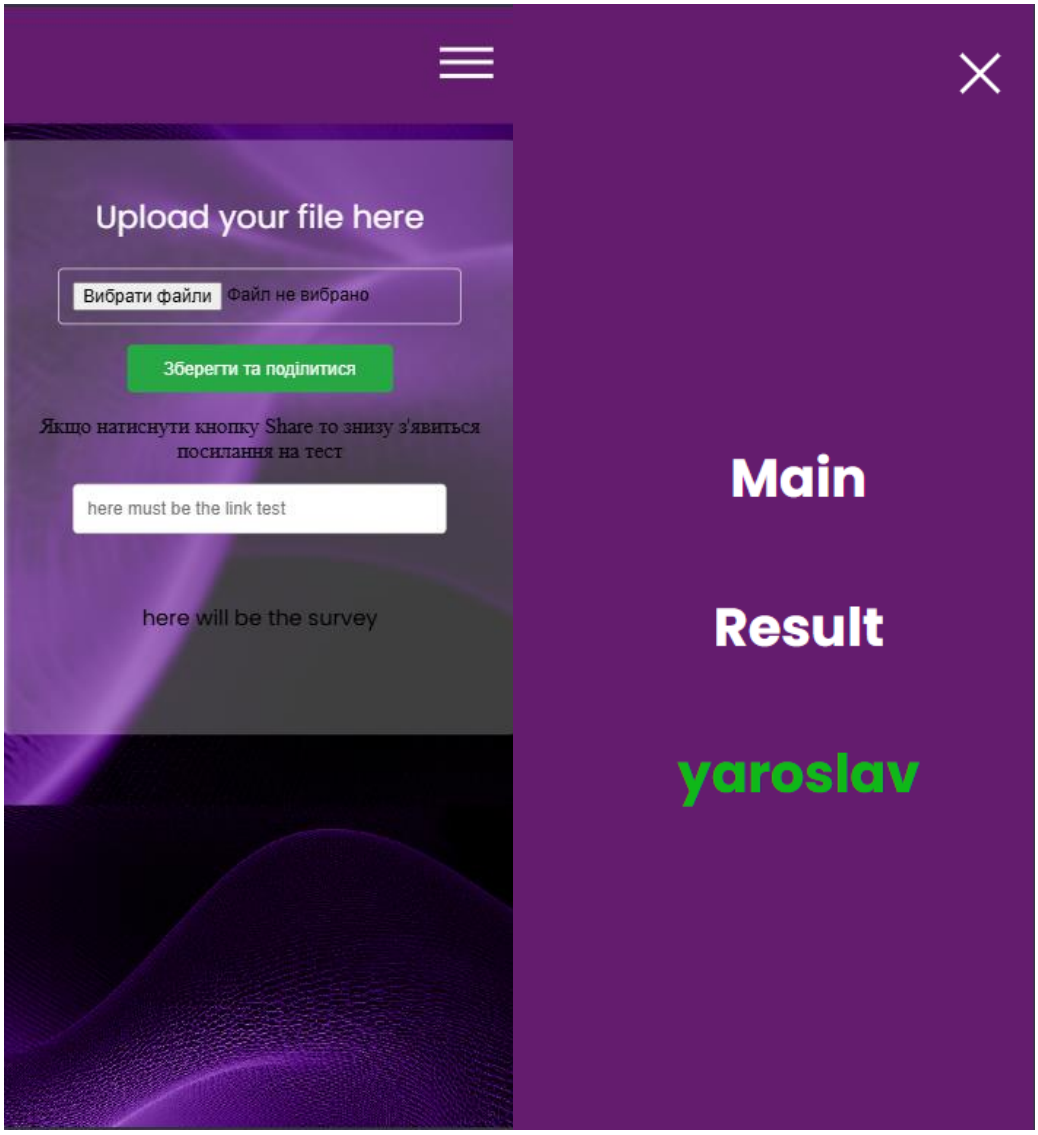

Рисунок 3.2. Симуляція відображення адаптивної верстки на iPhone12 у devTools

3.2. Обробка подій та даних на Frontend стороні

З метою обробки вибраного користувачем файлу до поля input з id=”input” прикріплений обробник подій, що реагуватиме на подію «change» викликаючи callback функцію handleInput. Вона свою чергу покликана зробити перевірку чи доступний файл та зчитати його викликавши наступну функцію proccessTheText() для подальшої обробки тексту. Для зчитування файлу використовується частина Web API (Application Programing Interface) FileReader. Він дозволяє програмі асинхронно зчитувати вміст файлу. Асинхронно, означає, що ця операція не блокує виконання наступного коду, адже вона переноситься в інший стек виконання [10, 18].

Лістинг 3.5

```
function handleInput(e) {
  const file = e.target.files[0];
  if (file) {
    let reader = new FileReader();
    reader.onload = function (event) {
      const text = event.target.result;
      proccessTheText(text);
    };
    reader.readAsText(file);
  } else {
    console.log(`I do not have any files to show you`);
  }
}
```

Задача функції proccesTheText() спочатку розпізнати всі окремі запитання у файлі та записати їх окремим набором рядків у масив.

Лістинг 3.6

```
let phrase = document.querySelector(".phrase");
phrase.classList.add("disactive");
let k = 0;
let texts = text.split(/\n\s*\n/);
```

Далі за допомогою циклу for of провести розділення запитання від відповідей та визначити якого типу відповіді потрібно створити викликавши функцію, що відповідає за цю частину роботи. На три типи запитань є три функції які створюють кожна свій тип: createInputField – створення поля для вводу тексту, createMultiAnswersTest – для опитування з декількома варіантами відповідей, createTest – для опитування з одним варіантом відповіді. Перед тим як їх викликати відповіді розділяються на масив рядків за допомогою методу split() та потім при виклику передаються окремо. Метод split() працює таким чином, що розділення рядка відбувається по символу який передається як аргумент цієї функції. Відокремлення питання відбувається по символу «?», а відповідей с «;». Тому ці символи мають важливе значення для коректної роботи програми. Щоб визначити

типи питань до рядків застосовується метод includes(). Він перевіряє чи містить даний рядок символ, що передається в дужки як аргумент. Якщо у рядку є цей символ? то метод вертає значення true, якщо ні – false. Цей символ є тим символом який позначає певний вид питання: «#» – питання з багатьма варіантами відповідей, «_» – відкрите питання і т.д.

Також важливим елементом алгоритму є інкрементування змінної k при обробці кожного наступного питання. Інкремент це унарна операція, яка збільшує значення змінної типу number на одиницю [21, 22]. Це допомагає правильно групувати варіанти відповідей в функції createMultiAnswersTest, та при зберіганні введених відповідей в об’єкті, створювати унікальні ключі за номером опитування.

Лістинг 3.7

```
for (const str of texts) {
    k++;
    if (str != "") {
      let strings = str.split("?");
      let question = strings[0];
      if (strings[1].includes("_____")) {
        createInputField(question, k);
      } else if (strings[1].includes("#")) {
        let answers = strings[1].split(";");
        createMultiAnswersTest(question, answers, k);
      } else {
        let answers = strings[1].split(";");
        createTest(question, answers, k);
      }
    }
  }
```

Нижче функція, що записує відповіді в об’єкт test.

Лістинг 3.8

```
function recordAnswer(question, k, answer) {
  test[k] = `${question}:${answer}`; }
```

Далі при натисканні кнопки «Submit» спрацьовує callback функція processData(), яка зчитує всі дані з форми, а також бере id значення поточного опитування яке зберігається в localStorage та розміщує цю інформацію в об’єкт під назвою customer [16,17]. У лістингу 3.9 наведено частину коду цієї функції.

Лістинг 3.9

```
let customer = {
    surveyId: surveyId,
    password: null,
    username: null,
    questionsAnswers: {},
  };
  let i = 0;
  for (const key in test) {
    const element = test[key];
    let strings = element.split(":");
    let question = strings[0].replace(/[\r\n]/g, "").trim();
    let answer = strings[1].replace(/[\r\n]/g, "").trim();
    customer.questionsAnswers[question] = answer;
    i++;
  }
```

Далі ця функція викликає наступну sentDataToServer() яка передає створений об’єкт customer, який містить всю необхідну інформацію для запиту на сервер з метою подальшого збереження інформації в базу даних H2. Ця функція повертає результат асинхронного запиту реалізованого за допомогою Fetch API, а саме Promise. В функцію fetch передаються два параметри: URL, та об’єкт що містить метод (в даному випадку POST), заголовки, серед яких Authorization. За допомогою нього передається разом із запитом jwtToken за допомогою якого відбувається верефікація користувач що увійшов в систему. Сам токен витягується із localStorage, у разі відсутності в цьому сховищі токена користувач вважається не авторизованим. Також об’єкт після заголовків передає саму інформацію у вигляді JSON рядка. Нижче наведено функцію запиту без обробки Promise.

Лістинг 3.10

```
return fetch("http://localhost:8081/api/add", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${getToken()}`,
    },
    body: JSON.stringify(dataObject),
  })
```

Однією з найважливіших функцій, з якої відбувається початок взаємодії користувача з платформою є функція checkUser(). Вона перевіряє чи є токен в локальному сховищі браузера, таким чином визначає чи зареєстрований користувач, чи ні. Якщо такого токена немає, викликає функцію, що генерує форму для реєстрації користувача або входу в систему - authorization(). Також перевіряє чи існує значення doneTest в localStorage. Якщо обидва записи існують, викликає функцію показу результатів, або ж за наявності параметру id в поточному url виводить першу сторінку. Якщо користувач увійшов в систему, але не пройшов опитування (doneTest = null), то виводиться початкова сторінка. В загальному ця функція викликається напочатку програми, після реєстрації та при натиску на кнопку меню «Result» (лістинг 3.11).

Лістинг 3.11

```
function checkUser() {
  const isLoged = localStorage.getItem("token");
  const doneTest = localStorage.getItem("doneTest");
  if(isLoged != null&&doneTest != "yes"){
    changeLoginBtn();
    checkUrl();
    showTheMain();
  }else if (isLoged!=null&&doneTest == "yes") {
    let isUrl = checkUrl();
    changeLoginBtn();
    if(isUrl!=true){
```

```
            showTheResult();
          }else{
            showTheMain();
          }
        } else {
          authorization();
          checkUrl();
        }
        let buttonMain = document.querySelector(".button-main");
        let buttonResult = document.querySelector(".button-result");
        buttonMain.addEventListener("click", showTheMain);
        buttonResult.addEventListener("click", showTheResult);
      }
```

У localStorage зберігаються такі дані: поточне ім’я користувача, id останнього пройденого опитування, інформація чи пройшов якесь опитування користувач чи ні та токен для авторизації.

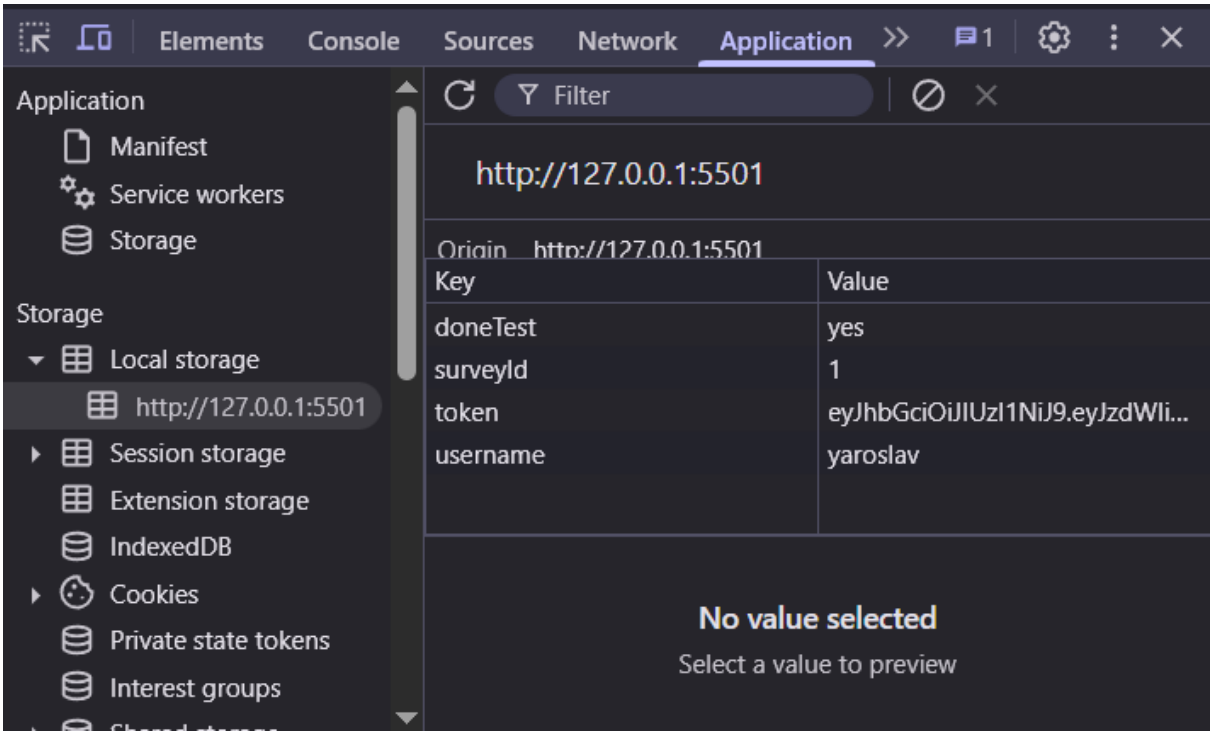

Рисунок 3.3. Записані дані в localStorage браузера Google Chrome

3.3. Реалізація Backend частини вебресурсу

Backend частина має архітектуру Layered Architecture. Суть цієї архітектури базується на ідеї розподілення об’єктів на окремі шари, що спілкуються тільки з сусідніми шарами та мають конкретну і вузьку спеціалізацію. В даній програмі це:

1. Security Layer (Security). Це перший рівень на якому знаходяться фільтри та механізм авторизації користувача. Якщо користувач авторизований або просто має певний доступ до деяких функцій вебсайту цей шар відправляє запит до Presentation Layer.
2. Presentation Layer (Controller). В цьому шарі знаходяться контролери, які відповідальні за комунікацію з клієнтською частиною вебсайту. Вони приймають тіло запиту та передають далі на Application Layer.
3. Application Layer (Service). Цей шар відповідальний за безпосереднє виконання обчислень, обробки інформації. По-іншому тут реалізована бізнес логіка вебплатформи.
4. Persistence Layer (Repository + Entity). Цей рівень напряму відповідальний за взаємодію із базою даних та реалізацію пошуку чи запису інформації. Також це відбувається за допомогою спеціальних об’єктів , які теж відносяться до цього шару (Entity) [31].

На рівні Security основними компонентами є jwtService, PasswordConfig, SecurityConfig. Їх головна задача аутентифікація користувача, перевірка статусу користувача, надання або обмеження дозволів у визначених випадках. В SecurityConfig основні налаштування CORS (Cross-origin resource sharing) наведені в лістингу 3.12. Це механізм, що визначає спосіб взаємодії клієнтських вебпрограм, які завантажені в одному домені, але ресурси знаходяться в іншому [32].

Ці налаштування означають:

- Дозвіл доступу до API лише з двох доменів: "http://127.0.0.1:5501", “http://localhost:5501”.
- Дозволяє такі запити з frontend: PUT, DELETE, POST, GET OPTIONS.
- Дозволяє будь-які заголовки у запитах.

- Дозволяє надсилати із запита дані для авторизації.

Лістинг 3.12

```
.cors(cors -> cors.configurationSource(request -> {
     var corsConfiguration = new
org.springframework.web.cors.CorsConfiguration();
     corsConfiguration.setAllowedOrigins(List.of("http://127.0.0.1:55
01", "http://localhost:5501"));
corsConfiguration.setAllowedMethods(List.of("GET", "POST", "PUT",
"DELETE", "OPTIONS"));
corsConfiguration.setAllowedHeaders(List.of("*"));
     corsConfiguration.setAllowCredentials(true);
     return corsConfiguration;
}))
```

Нижче розташований код реєстрації jwtDecoder в @Bean який вміє розшифровувати токен підписаний моїм секретним ключем, який прописаний в jwtService. Цей клас з точки зору архітектури належить до шару Service, тому він буде розглянутий пізніше. При цьому використовується класичний варіант з бібліотекою Nimbus (лістинг 3.13). Далі він використовується в SecurityFilterChain для того щоб кожен раз при запиті користувача перевіряти токен. У разі невдачі у створенні його прописана обробка помилки за допомогою try, catch.

Лістинг 3.13

```
@Bean
    public JwtDecoder jwtDecoder() {
        try {
            SecretKey secretKey = jwtService.getSecretKey();
            return NimbusJwtDecoder.withSecretKey(secretKey).build();
        } catch (Exception e) {
            logger.error("Error while creating JwtDecoder: ", e);
            throw e;
        }}
```

На рівні презентації реалізовано два контролери, один відповідає за взаємодію з об’єктом Customer, а інший з об'єктом Survey (опитування). Їхні назви відповідні: CustomerController, SurveyController.

В CustomerController є такі маршрути:

- **api/all (GET)** – відповідає за повернення всіх користувачів які є зареєстровані в базі даних. Відповідь – масив об’єктів користувачів.
- **api/takesome (GET)** – очікує в параметрах запиту id опитування, щоб потім повернути відповіді поточного користувача на опитування за цим id. Відповідь – об’єкт з полями питання та відповідь на питання.
- **api/add (POST)** – очікує отримати в параметрах запите об’єкт DTO класу CustomerRequest. Виконує додавання записів відповідей користувача на конкретне опитування. Відповідь – текст та код чи успішно виконався запит.
- **api/info (GET)** – очікує в параметрах id опитування для того щоб знайти всі відповіді на опитування за цим id та сформувати інформацію в об’єкт який може надсилатись в сервіс Chart.js для формування діаграм. Відповідь – об’єкт з всією необхідною інформацію про відповіді на опитування. Поля: питання, вкладений об’єкт з відповідями на це питання та з їх кількістю.
- **api/auth/login (POST)** – це маршрут за яким користувач може відправити запит на отримання токену тим самим здійснює вхід користувача в систему. Відповідь – текст результату запиту.
- **api/auth/register (POST) –** цей маршрут відповідає за реєстрацію нового користувача та запис в базу, даних в зашифрованому вигляді. Відповідь – текст результату запиту.

В SurveyController реалізовано наступні маршрути для управління опитуваннями:

- **api/survey/save (POST)** – очікує в параметри DTO об’єкт типу SurveyRequest. В ньому має бути інформація для створення опитування

для того щоб зберегти в базу даних. Відповідь – текстове повідомлення результату збереження в базу даних.

- **api/survey/get (GET)** – очікує в параметри id опитування для того щоб за ним знайти його в базі даних. Відповідь – об’єкт типу Survey з полями surveyId та surveyData.
- **api/survey/delete (DELETE)** – очікує в параметри id опитування, здійснює видалення з бази даних опитування. Відповідь – текстовий результат здійснення видалення в базі даних.

Рівень Application містить: SurveyService, CustomerService, ResearchService, jwtService. Компонент SurveyService реалізує бізнес-логіку, пов’язану з управлінням опитуванями, а саме збереженням, отриманням та видаленням опитувань в базі даних.

Основні функції цього сервісу:

- Збереження опитування: функція addSurvey();
- Читання опитування: функція takeSurvey();
- Видалення опитування: функція deleteSurvey();

Наступний CustomerService реалізує бізнес-логіку, пов’язану з управлінням користувачем здійснення вибірок, а також авторизації корситувача.

До основних функцій належать:

- Здійснення кодування registerUser();
- Здійснення верифікації паролю verifyPassword();
- Здійснення вибірки усіх користувачів takeAllCutomers();
- Здійснення вибірки корситувача по id getMethodName();
- Пошук користувача по імені loadByUsername();
- Здійснення входу користувача в систему authentiticateUser();
- Здійснення реєстрації користувача signUpUser();
- Здійснення збереження записів відповідей користувача saveQuestionsAnswers();

- Отримання інформації про відповіді всіх користувачів на поточне опитування getInfo();

Особливість цього сервісу є те що для перевірки аутентифікаці корисувача використовується SecurityContextHolder, що є класичним рішенням в Service. В умові перевірки перевіряється чи взагалі немає жодної аутентифікації, чи встановлений прапорець аутентифікації, чи користувач анонімний (лістинг 3.14).

Лістинг 3.14

```
Authentication authentication = SecurityContextHolder.getContext()
.getAuthentication();
if (authentication == null || !authentication.isAuthenticated() ||
authentication.getPrincipal().equals("anonymousUser")) {
            throw new AccessDeniedException("User is not
authenticated");
        }
```

ResearchService реалізує бізнес-логіку пошуку і аналізу даних та перетворення в необхідну структуру для подальшої відправки з метою рендерингу діаграм. Ось його основні функції:

- Вибірка всіх даних по опитуванню, складання необхідної структури даних getInformation();
- Знаходження інформації щодо відповідей на конкретне опитування щодо конкретного користувача takeCustomerInfo();

jwtService дуже важливий і основний в реалізації jwt футентифікації. Тут розташована основна бізнес логіка цієї технології. До основних функцій відносяться:

- Генерація токену generateToken();
- Здійснення перевірки валідності токену validateToken();
- Витягування імені користувача із токену extractUsername();
- Витягування дати закінчення терміну дії токену extractExpirationDate();
- Витягування id користувача із токену extractUserId();

Також термін дії токену встановлено 1 рік. Секретний ключ генерується за допомогою кодування HS256. HS256 (HMAC з SHA-256) – це такий алгоритм хешування, який вимагає реалізацію не двох ключів(приватний та публічний), а один спільний. Такий ключ називається симетричним, він використовується як для підпису так і для його перевірки. В такому випадку потрібно бути дуже пильним щоб верифікатори були захищеними. Загалом симетричний ключ пропонує більшу простоту в реалізації, проте якщо є необхідність в більшій безпеці краще використовувати асиметричний [33]. З коду програми наведеного в лістингу 3.15, можна побачити реалізацію генерацію секретного ключа та його перетворення в формат base64.

Лістинг 3.15.

```
SecretKey key = Jwts.SIG.HS256.key().build();
   String base64Key = java.util.Base64.getEncoder()
.encodeToString(key.getEncoded());
```

В функції generateToken() реалізовано генерацію токену таким чином, що він складається із userId користувача, а також із username. Це безпечно, адже ніякі паролі чи конфіденційні дані там не знаходяться.

Persistence Layer складається з таких репозиторіїв та сутностей як: Customer, Survey, SurveyAnswer, CustomerRepo, SurveyRepo, AnswerRepo.

Найголовнішою сутністю є Customer адже вона відповідальна за реєстрацію та «існування» користувача в системі. Вона має анотацію @Entity, а також @Table. Ім’я таблиці в яку зберігатиметься цей об’єкт за допомогою Hibernate відповідає назві сутності – «customer». Цей клас має наступні поля:

- Id типу Long має анотацію @Id, а також GeneratedValue (strategy = GenerationType.IDENTITY). Це означає, що id автоматично генеруватиметься та ніколи не повторюватиметься (буде унікальним для кожного користувача).
- Username типу String, це поле набуває значення із форми для реєстрації.

- Password також типу String, це поле також набуває значення відповідно до того який пароль ввів користувач. Цей пароль зберігається не у відкритому вигляді, а хешованому за допомогою passwordEncoder.encode(), який повертає хешований пароль у форматі BCrypt. Цей метод дає більше безпеки через випадкові дані. Таким чином навіть два однакові паролі будуть мати різне хешування, що суттєво збільшує безпеку [35].
- Role типу String, поки завжди null, адже ще не реалізовано функціонал для користувачів з різними ролями. В майбутньому його варто розробити.

Наступною важливою сутністю є Survey. Вона набагато простіша за попередню та призначена для збереження опитування в базі даних. Дані цієї сутності зберігаються в таблицю «survey». Survey має наступні поля:

- Id з анотаціями @Id, @GeneratedValue (strategy = GeneretionType.IDETITY), що визначає це поле як id та задає автоматичну генерацію унікальних Id. Тип даних поля: Integer.
- SurveyData типу String, це поле призначене для збереження всієї інформації про опитування, наприклад: питання.

Наступною дуже важливою сутністю, яка зв’язує дані про користувача та про опитування є SurveyAnswer. Ця сутність містить наступні поля:

- Приватне поле Id з тими самими анотаціями, що і в попередніх сутностях. Призначене для ідентифікації кожного окремого запису в таблиці під назвою «questions-answers».
- Поле userId типу Integer. Це поле з ідентифікатором відповідного користувача який відповідь якого зберігається в цьому записі.
- Поле question типу String зберігає конкретне питання опитування.
- Поле answers типу String зберігає відповіді одного користувача на питання.

- Поле типу Integer під назвою surveyId зберігає ідентифікатор конкретного опитування до я кого належить це питання.

Репозиторії всі успадковуються від інтерфейсу crudRepository, що належить до фреймворку Spring Data у Spring Boot.

В репозиторії AnswerRepo методи з лістингу 3.16. є частиною Spring Data JPA, який автоматично генерує запити до бази даних на основі назви методів. В даному випадку перший метод відповідає за вибірку по id опитування, а другий за двома параметрами, id опитування та користувача.

Лістинг 3.16

```
public interface AnswerRepo extends CrudRepository<SurveyAnswer,
Integer>{
    List<SurveyAnswer> findBySurveyId(Integer surveyId);
    List<SurveyAnswer> findBySurveyIdAndUserId(Integer surveyId,
Integer userId);
}
```

В репозиторії CustomerRepo додано лише один такий метод findByUsername() здійснює пошук користувача за іменем. В SurveyRepo немає доданих спеціальних методів пошуку, використовуються стандартні CRUD операції, успадковані від інтерфейсу crudRepository.

ВИСНОВКИ

В результаті здійсненого дослідження, вивчено проблему проведення опитувань та виявлено, що найкращим методом її вирішення це проведення автоматизації цього процесу. У цій курсовій роботі розглянуто існуючі вебресурси для автоматизованого проведення опитувань та проведено їх аналіз. На його основі здійснено вибір необхідних технологій та функціоналу для розробки власного рішення.

В результаті розроблено вебзастосунок автоматизації створення форм для опитувань. Для досягнення цієї мети було розглянуто наявні рішення в автоматизації проведення опитування, а також проведення аналізу їх недоліків та переваг, технологій застосованих в сучасних платфомах даного типу. А також за допомогою сучасних технологій розроблено адаптивний дизайн. Завдяки цьому забезпечено можливість користувачів проходити опитування з будь-якого пристрою, що значно розширює цільову аудиторію та створює основу для популярності додатка в майбутньому.

Також було розроблено архітектуру та backend частину вебплатформи з використанням фреймворків таких як Java Spring Boot, Hibernate. Реалізовано збереження опитування в базу даних, аутентифікацію корситувача, та можливість поширення опитування за допомогою згенерованого посилання. Також розроблено вивід базової аналітики про опитування за допомогою API стороннього сервісу Chart.js.

Дана вебплатформа може використовуватись у сфері освіти для проведення опитувань в закладах освіти також для організації онлайн навчання. Дуже корисним може бути у сфері бізнесу для опитування клієнтів з метою покращення якості сервісу та випуску нових продуктів.

Додаток має великі перспективи розвитку, удосконалення та оновлень. Наступними кроками можуть бути створення різних ролей користувачів, покращення аналітики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 18 сервісів для проведення опитувань. URL: https://www.plerdy.com/ua/blog/18-servisov-dlja-provedenija-oprosov/#Online (дата звернення 5.05.25)
2. Survey Automation: Turning Data into Delightful Customer Experiences. URL: https://www.zonkafeedback.com/blog/survey-automation (дата звернення 05.05.25)
3. The ultimate guide to automating survey research. URL: https://www.kantar.com/campaigns/pf/the-ultimate-guide-to-automating-survey-research (дата звернення 05.05.25)
4. The 5 Most Popular Online Survey Tools. URL: https://imotions.com/blog/insights/the-5-most-popular-online-survey-tools/(дата звернення 05.05.25)
5. Головна сторінка платформи SurveyMonkey. URL: https://www.surveymonkey.com/home/ (дата звернення 06.05.25)
6. Як працює CSS Flexbox. URL: https://petrov.net.ua/how-css-flexbox works/ (дата звернення 06.05.25)
7. Документація для веброзробників MDN, Flexbox. URL: https://developer.mozilla.org/en-US/docs/Learn _web_ development /Core /CSS_ layout/Flexbox (дата звернення 06.05.25)
8. Що таке медіа запити і для чого вони потрібні. URL: https://freehost.com.ua/ukr/faq/articles/chto-takoe-media-zaprosi-css-i-dlja-chego-oni-nuzhni/ (дата звернення 06.05.25)
9. Документація для веброзробників MDN, FileReader. URL: https://developer.mozilla.org/en-US/docs/Web/API/FileReader (дата звернення 07.05.25)
10. Створити онлайн-опитування | 2024 Покроковий посібник. URL: https://ahaslides.com/uk/blog/create-survey-online/(дата звернення 07.05.25)

11. 5 конструкторів онлайн опитувань, які зроблять ваш маркетинг краще. URL: https://www.imena.ua/blog/5-constructors-of-online-surveys/ (дата звернення 07.05.25)
12. Офіційний сайт платформи для проведення опитувань Qualtrics. URL: https://www.qualtrics.com/strategy/research/survey-software/ (дата звернення 08.05.25)
13. Top Tools and Technologies for Creating Effective Customer Surveys. URL: https://www.chatbot.com/blog/top-technologies-for-customer-surveys/ (дата звернення 08.05.25)
14. 13 сервісів для створення онлайн-опитувань на сайті. URL: https://hostiq.ua/blog/ukr/survey-tools/(дата звернення 08.05.25)
15. Marijn Haverbeke. Eloquent JavaScript, 4th Edition. Сан-Франциско: No Starch Press, 2024, 456 с.
16. Ерік Фрімен, Елізабет Робсон переклад Ганна Якубовська Програмування на JavaScript. Харків: Фабула, 2022. 672с.
17. Ben Frain Responsive Web Design with HTML5 and CSS , 4th Edition. Birmingham: Packt Publishing 2022. 498с.
18. Етайн Браун Вивчаємо JavaScript: керівництво зі створення сучасних вебсайтів (3-є видання). Харків: ProfiBooks, 2021 368с.
19. Бородкіна І. Л. Бородкін Г. О. Web-технології та web-дизайн: застосування мови HTML для створення електронних ресурсів: навч. посібник. Київ : Ліра-К, 2020. 210 с.
20. Шикула, О. М. Вступ до сучасного Web-дизайну: HTML5+CSS3: навчальний посібник. Київ : ІПДО, 2019. 240 с.
21. David Flanagan. JavaScript: The Definitive Guide. 7th Edition. O'Reilly, 2020. 706 p.
22. Сучасний підручник та довідник з JavaScript. URL: https://javascript.tutorial.in.ua/ /(дата звернення 15.05.2025)

23. ::before/::after — A Complete Guide. URL: https://medium.com/@RitikaAgrawal08/before-after-a-complete-guide 5ae39240d520# (дата звернення: 15.05.2025)
24. Advantages and Disadvantages of Online Surveys: Everything You Need to Know. URL: https://www.proprofssurvey.com/blog/advantages-disadvantages-of-online-surveys/ (дата звернення: 15.05.2025)
25. What is Java technology and why do I need it? URL: https://www.java.com/en/download/help/whatis_java.html (дата звернення:16.05.2025)
26. REST API як спосіб спілкування компонент вебдодатків. URL: https://foxminded.ua/shcho-take-rest-api/ (дата звернення: 16.05.2025)
27. REST API: що це, як працює, переваги і приклади. URL: https://goit.global/ua/articles/rest-api-shcho-tse-iak-pratsiuie-perevahy-i-pryklady/ (дата звернення: 16.05.2025)
28. Що таке REST API? Основні принципи REST та GET, POST, PUT, PATCH, DELETE. URL: https://tseivo.com/b/memecode/t/jgjojmb1ar (дата звернення: 17.05.2025)
29. Повний огляд REST: нюанси, поради, приклади. URL: https://dou.ua/forums/topic/50364/ (дата звернення: 17.05.2025)
30. Вступ до REST API — RESTful вебсервіси. URL: https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi (дата звернення: 17.05.2025)
31. Chapter 1. Layered Architecture. URL: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html (дата звернення: 17.05.2025)
32. What is CORS? URL: https://aws.amazon.com/what-is/cross-origin-resource-sharing/ (дата звернення:17.05.2025)
33. Comparison of RS256 and HS256 Algorithms for Token Signing in Cryptography. URL:https://bvsreyanth.medium.com/comparison-of-rs256-and-hs256-algorithms-for-token-signing-in-cryptography-bd21e9e7a54d

(дата звернення:17.05.2025)

34. Unified Modeling Language. URL: https://uk.wikipedia.org/ wiki/ Unified_ Modeling_Language (дата звернення: 17.05.2025)
35. Password hashing using BCrypt in spring security. (PART-8). URL: https://tharushkaheshan.medium.com/password-hashing-using-bcrypt-in-spring-security-part-8-d867a83b8695 (дата звернення: 17.05.2025)
36. Гурик Я.М., Шевцова Н.В. Розробка вебресурсу для автоматизованого створення та проведення опитувань. Наука, освіта, суспільство очима молодих : матеріали XVIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих учених м. Рівне, 14 травня 2025 р. Рівне, 2025. С.273–274.