Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для проведення соціальних опитувань.
Серверна високонавантажена частина
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-6

Масенко Я. В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Валенда Н. А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Масенко Ярославу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система для проведення соціальних опитувань. Серверна високонавантажена частина

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи Розробити серверну частину застосунку для проведення соціальних опитувань, до функціоналу якої належить генерація розмітки опитувань, збирання з них статистики, забезпечення додаткової логіки, тощо.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	09.04.2024	*виконано*
2	Створення специфікації ПЗ	14.04.2024	*виконано*
3	Проектування ПЗ	23.04.2024	*виконано*
4	Розробка ПЗ	02.05.2024	*виконано*
5	Тестування ПЗ	30.05.2024	*виконано*
6	Оформлення пояснювальної записки	03.06.2024	*виконано*
7	Підготовка презентації та доповіді	07.06.2024	*виконано*
8	Попередній захист	10.06.2024	*виконано*
9	Нормоконтроль, рецензування	13.06.2024	*виконано*
10	Здача роботи у електронний архів	14.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	18.06.2024	*виконано*

Дата видачі завдання 08 квітня 2024р.

Студент _____ Масенко Я. В.

(підпис)

Керівник роботи _____ доц. кафедри ПІ Валенда Н. А.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра містить: 81 сторінку, 5 розділів, 46 рисунків, 10 джерел посилання.

СОЦІОЛОГІЧНЕ ДОСЛІДЖЕННЯ, ОПИТУВАННЯ, АНКЕТУВАННЯ, СТАТИСТИКА, ТАРГЕТУВАННЯ, СТОВПЧИКОВА БАЗА ДАНИХ.

Метою роботи є розробка серверної високонавантаженої частини програмної системи для проведення соціальних опитувань, що дозволить швидко обробляти запити на отримання опитування, події по проходженню опитування та запис цих подій в БД для подальшої обробки статистики.

Технології розробки серверу базуються на використанні середовища розробки Goland, мови програмування Golang та доступу до баз даних MS SQL Server та Vertica. З'єднання між різними сервісами підтримується за допомогою протоколу gRPC та сховища даних.

У результаті роботи проведено аналіз предметної галузі, визначено функціональні вимоги до системи, спроектовано та реалізовано схему стовпчикової бази даних та програмну архітектуру високонавантаженої частини системи «PureSurvey» (що складається з обробника подій, генератора розмітки та менеджера статистики).

SOCIOLOGICAL RESEARCH, SURVEY, QUESTIONNAIRE, STATISTICS, TARGETING, COLUMNAR DATABASE.

The aim of the work is to develop a server-based high-load part of a software system for conducting social surveys, which will allow to quickly process survey requests, survey completion events and record these events in the database for further processing of statistics.

Server development technologies are based on the Goland development environment, Golang programming language and access to MS SQL Server and Vertica

databases. The connection between different services is supported by the gRPC protocol and the data warehouse.

As a result of the work, the subject area was analysed, the functional requirements for the system were determined, the columnar database scheme and the software architecture of the high-load part of the PureSurvey system (consisting of an event handler, a markup generator and a statistics manager) were designed and implemented.

Я, Масенко Ярослав Володимирович, студент гр. ПЗПІ-20-6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для проведення соціальних опитувань. Серверна високонавантажена частина», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Соціологічні дослідження не зменшують своєї важливості протягом останніх десятиліть, оскільки сприяють вивченню та аналізу різноманітних аспектів суспільного життя. Перенесення цих процесів у Інтернет-простір стає необхідністю у зв’язку з активним розвитком інформаційних технологій та зміною способів взаємодії.

Україна не є винятком у цьому процесі, забезпечуючи доступ до досить доступного та швидкого Інтернету, а також впроваджуючи цифрові ініціативи, такі як Дія. Ситуація з пандемією COVID-19 також прискорила перехід держави до онлайн-сервісів, що може стати активним рушієм у напрямку впровадження цифрових інструментів для проведення соціологічних досліджень.

Поява електронних сервісів, які дозволяють проводити опитування, має великий потенціал у забезпеченні ефективності та швидкодії цих процесів. Такі сервіси дозволяють збирати та аналізувати дані швидше, ефективніше та точніше, забезпечуючи можливість проведення глибоких аналізів за різними параметрами, такими як вік, гендер, країна тощо.

Метою даного проєкту є створення серверної частини програмної системи для проведення соціальних опитувань. Ця високонавантажена система дозволяє користувачам вбудовувати таргетовані опитування на свої сайти, збирати та аналізувати статистику, розподілену за такими параметрами, як вік, гендер, країна тощо. Опитування можуть бути налаштовані згідно з унікальними потребами користувачів, зокрема потребами у зовнішньому вигляді та в локалізації.

Однією з ключових переваг цього проєкту є його швидкодія, яка дозволяє збирати дані з найменшими затримками, що особливо важливо при потребі оперативного аналізу результатів опитувань. Завдяки оптимізації серверної частини, користувачі можуть отримати надійну та продуктивну систему, що відповідає сучасним вимогам та дозволяє ефективно взаємодіяти з аудиторією.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Під соціологічним дослідженням у світовій практиці розуміють будь-яке дослідження (незалежно від застосованого методу збору й аналізу даних), що формує, розвиває, уточнює або заперечує ту чи іншу теорію в галузі соціології. В Україні цей термін часто є тотожним із опитуванням громадської думки, але ця думка хибна, адже насправді репрезентативне опитування є лише одним із багатьох методів соціального дослідження [1, с. 22-23].

Дослідники проводять їх для розширення знань в різних галузях або перевірки припущень, що в свою чергу приносить користь суспільству і закриває частину його потреб. Безпосередньо зібрана від людей інформація допомагає створювати цілеспрямовані заходи реагування на потреби суспільства урядом чи бізнесом.

Проведення соціальних досліджень можна поділити на два способи: кількісний та якісний. У випадку якісних, дані збираються в природному середовищі респондента. Це сприяє зміцненню довіри, а отже, дає змогу отримати реальні та точні дані. Елементами якісного соціального дослідження є питання з відкритою відповіддю та інтерв’ю. Збір даних від цільової групи може відбуватися за допомогою фокус-груп, інтерв’ю, етнографічних досліджень, кейс-стаді тощо.

Для кількісного дослідження з цільової аудиторії відбирається вибірка, яка гарантує, що зібрані дані будуть репрезентативними для всього цільового ринку. Дані для них збираються за допомогою опитувань, анкетувань, інтерв’ю, експериментальних досліджень, тощо. Методом кількісного методу збору даних є отримання числових даних, які можна легко проаналізувати за допомогою статистичних програм.

Серед методів кількісних досліджень, особливо в Україні, превалює проведення опитувань (анкетувань та інтерв’ю). Їхні вдосконалення, такі як імовірнісна вибірка та стандартизовані вимірювання, дозволяють дослідникам бути впевненими в тому, що вибірка не є упередженою і що вони отримують порівнянну

інформацію про кожну особу з відібраної сукупності. Таким чином, імовірнісна вибірка за допомогою опитувальника дозволяє зробити описові твердження про характеристики великої популяції [1, ст. 90]. Жоден інший метод не може зробити такого твердження.

Ще однією перевагою такого опитування є те, що воно, в певному сенсі, є гнучким – можна поставити багато запитань на одну тему, та легко додати нові в процесі розробки чи тестування. Цей фактор дає дві переваги: можна включити велику кількість змінних, що дозволяє статистично контролювати потенційні помилки, а це, в свою чергу, означає, що гіпотетичні причинно-наслідкові зв'язки можуть бути перевірені.

Однак слід усвідомлювати, що опитування як метод проведення досліджень мають певні недоліки. При зборі даних з великої вибірки часто доводиться розробляти узагальнені відповіді до питань, що зазвичай означає втрачену частку потенціалу опитування, і таким чином, висвітлення складних тем може виглядати поверхневим. Анкетування можуть вимірювати лише те, що люди готові сказати, і тому вони можуть не виявити ставлення, яке є соціально неприйнятним. Крім того, вони не надають жодних доказів, які б підтверджували ці переконання, а отже немає способу перевірити достовірність наданої відповіді.

Незважаючи на ці проблеми, популярність опитування в соціальних науках призвела до розробки різних методик проведення анкетувань та інтерв'ю. Серед основний переваг анкетування можна навести:

- можливість самостійного проходження;
- можливість розповсюдження телефоном, поштою або в електронному вигляді;
- зниження витрат на проведення завдяки більш структурованому шаблону з меншою варіацією відповідей.

Витрати ж при проведенні інтерв'ю будуть набагато вищими, оскільки воно потребує значної кількості людського ресурсу. Але інтерв'юер, як реальна людина, може зменшити проблему з неточними чи неправдивими відповідями на анкети,

оскільки його запитання дадуть змогу детально дослідити почуття респондентів, щоб виявити, що вони насправді мають на увазі під висловленими ідеями.

При створенні опитувань дослідники керуються принципами, що дозволяють збільшити кількість відгуків на них, зокрема:

- опитування має містити вступ з його метою, інструкцією про заповнення, вказівку на організацію що проводить дослідження, приблизну тривалість його проходження, а також подяку за участь;
- дизайн опитування має бути чітким і легким для читання;
- питання мають бути розташовані в правильному порядку і зрозумілі.

Зрозумілість також означає, що на запитання має бути легко відповідати. Це означає наявність знайомої для респондента мови анкети, відсутність двозначних термінів, та довгих запитань, оскільки вони можуть заплутати респондента та призвести по передчасного завершення проходження опитування. Також спроба об’єднання запитань для заощадження місця може призвести до непорозумінь або пропущених запитань.

Ще важливим моментом є наявність залишкового варіанту відповіді у запитаннях (на зразок варіантів «Інше», або «Не маю відповіді»). Це дозволяє класифікувати усіх респондентів, по визначених категоріях, навіть тих хто не вніс відповіді [3, ст. 4-5].

При проведенні дослідження, незалежно від його типу, дослідник повинен пройти такі етапи:

- визначення змісту опитування та формування запитань;
- попереднє тестування анкети; прийняття рішення про те, хто буде залучений до опитування (вибірка людей) і коли воно проводитиметься;
- аналіз та інтерпретація результатів;
- подання результатів у вигляді необхідних графіків чи таблиць.

Наприкінці дослідження відбувається аналіз зібраних даних та формування звітів. Аналіз даних є ґрунтовною роботою та залежить від характеру зібраних даних (зокрема аналіз якісних досліджень полягає у впорядкуванні та сегментуванні отриманих відповідей).

Сформовані звіти ж мають містити детальний опис усіх аспектів дослідження, зокрема мету, дослідницьке питання, гіпотезу, та методи, використані для збору та аналізу даних. Вони повинні висвітлювати результати у спосіб, який буде зрозумілим для читачів, зокрема використовувати діаграми, таблиці або графіки.

1.2 Виявлення та вирішення рішень

Проведення досліджень у вигляді анкетування є досить простим та звичним для соціуму способом кількісних соціологічних досліджень. Але зважаючи на те, що анкети передбачають собою збір та обробку великої кількості даних (здебільшого числових), доречно було б спростити частину цієї роботи завдяки впровадженню цифрових технологій.

Необхідно взяти до уваги те, що основні недоліки анкетування (узагальнення відповідей та відсутність доказів позиції) є об’єктивними та їхнє подолання потребує досліджень та загалом тривалої роботи людей з освітою в галузі соціології. Це означає що при розробці системи для полегшення їхньої роботи необхідно зосереджуватися не так на подоланні цих недоліків, як на вдосконаленні вже існуючих переваг, а також на пошуку вирішень дрібних точок напруги.

Як було зазначено при аналізі предметної галузі, однією зі значних переваг опитувальника є його гнучкість: можливість додавання нових питань чи варіантів відповідей без значної шкоди для його логічної структури (що, правда, може бути складнішим для великих наборів питань). Також було вказано на важливість правильної послідовності питань для залучення респондента в процес. З цього можна зробити висновок що першим необхідним функціоналом системи є можливість швидко переглянути всю структуру опитування що створюється, за потреби відредагувати його в будь-якому місці, та змінити послідовність запитань чи варіантів відповідей. Це забезпечить найкращий досвід респондента.

Наступним пунктом можна виділити важливість зовнішнього вигляду анкети. Раніше було наголошено на те, що чіткий та легкий для читання дизайн сприяє більшій кількості проходжень опитувальника. Для забезпечення цієї умови

потрібно переконатися що шаблони анкет, які буде надано клієнтам, мають достатні відступи, зокрема на мобільних пристроях; що текст в них не може зливатися з фоном; що доступні шрифти є легкими для читання. Також було б доречним додати можливість увімкнення респондентом дизайну для людей з порушеннями зору, який би був доступний завжди, без додаткового втручання людини, що складає опитувальник.

Особливу увагу необхідно звернути на потребу респондента в зрозумілій мові опитувальника. Не важко здогадатися що людина взагалі не зможе пройти його, якщо він складений незнайомою мовою. Отже такий випадок є потенційно пустою статистикою переглядів анкет, та втраченим респондентом. Щоб уникнути цієї ситуації, важливо втілити в системі можливість локалізації питань, які складає клієнт, а також автоматичний підбір мови за країною респондента. Гарним інструментом був би перемикач мови на самому опитуванні, що дозволив би запам’ятати вибір користувача для подальших проходжень.

Однією з основних задач дослідника при підготовці дослідження є визначення вибірки опитуваних, яка найкраще відобразить мету його роботи. Наприклад анкетування про досвід життя під час війни варто було б надавати для проходження саме людям з України, а не Італії. Для цього в системі повинен бути передбаченим функціонал таргетування. Його задача – показувати респонденту тільки ті опитування, в налаштуваннях яких обрано параметри, яким він безпосередньо відповідає. Разом з цією функцією, не завадило б додати можливість показувати користувачам з різних таргетних груп – різні опитування, на одному й тому ж місці на сайті.

Зібрана статистика є дуже цінним результатом проведення опитування, але без належної обробки вона є лише даними, що збережені в електронному варіанті і ніким не використовуються. Щоб отримані дані були оброблені просто лежати в базі даних, а має використовуватися для проведення аналізу спеціалістами. Аналіз може відбуватися за повним набором даних, які збирає система, або за зрізами по різних характеристик (як от за країною, чи типом пристрою). Необхідно надати можливість користувачу переглядати статистику по зрізах в режимі реального часу,

завантажувати її в зручних для нього форматах (xlsx, txt, csv, pbix, тощо), а також будувати найбільш поширені графіки за набором даних.

Щоб звернутися до функціоналу продуктів-конкурентів для актуалізації рішень, можна дослідити можливості що надає платформа SurveyLegend. Ця система є прямим суперником на ринку продукту що розробляється, адже до її функцій зокрема належать:

- підтримка локалізації;
- автоматичне збереження при створенні та редагуванні;
- різні типи запитань (текст, числа, випадний список, слайдер, рейтинг, завантаження файлів);
- сторінки привітання та подяки за проходження;
- логіка розгалуження питань;
- кольорові палітри, ефект розмитого фону, вибір зображення для фону;
- захист від спаму;
- підтримка роботи в режимі інформаційної дошки на вулиці;
- аналітика в реальному часі та різні варіанти графіків;
- завантаження результатів в різних форматах;
- експорт результатів на Google Drive;

Навіть з наведеної частини функціоналу помітно дуже широкий вибір інструментів в системі. Зокрема підтримка великої кількості типів запитань може стати одним з ключових факторів при виборі SurveyLegend серед конкурентів. Ця особливість може знадобитися при проведенні опитувань різних вікових категорій населення. Наприклад учням старших класів буде цікавіше відповідати на запитання з картинками, виставляти рейтинг, тощо.

Деякі з функцій, як от «режим інформаційної дошки», видаються надмірними для проєкту на початку його існування, адже потребує значного часу розробки, але не буде використовуватися більшою частиною клієнтів.

На скриншоті, що зображує процес створення опитування в особистому кабінеті SurveyLegend (див. рис. 2.1) можна помітити наявність інструментів, про які йшла мова в переліку функцій.

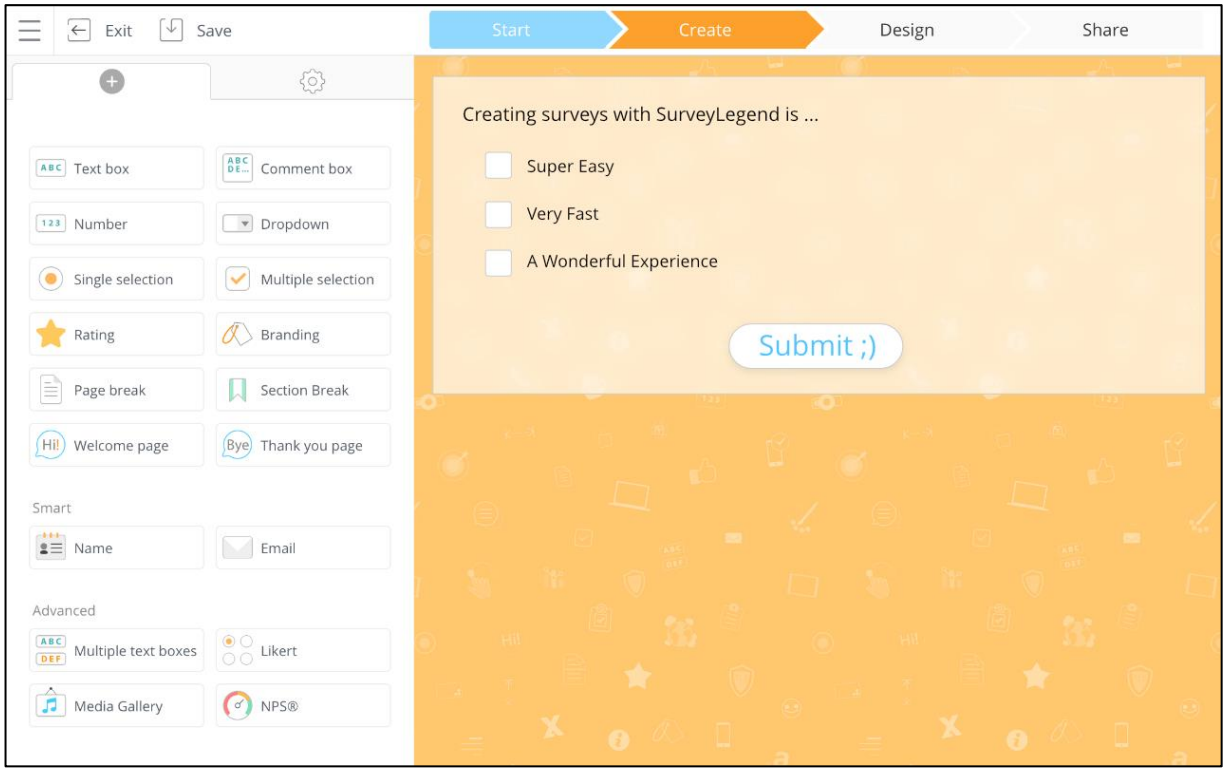

Рисунок 1.1 – Інтерфейс створення опитування в системі SurveyLegend

Інтерфейс платформи можна назвати приємним для використання, адже він має свою обмежену кольорову гаму, невелику кількість тексту та достатньо великі елементи керування. Але як зауваження можна навести те, що панель інструментів ліворуч займає велику частину екрану, і була б доречною можливість зменшувати її, прибираючи текст (адже піктограми чітко показують їх призначення).

Ще однією перевагою платформи можна навести наявність функціоналу для створення вбудовуваних (embedded) опитувань [4], тобто таких, які можна досить безшовно інтегрувати на будь-які веб-сторінки. Сервіс надає великий інструментарій з редагування зовнішнього вигляду таких анкет, хоча серед опцій відсутнє настроювання позиції блоку на сторінці. Це унеможливлює їх використання на сайтах, де було б доречно постійно тримати опитування перед очима користувача: у вигляді плаваючого елементу, або як накладання поверх основного вмісту, з можливістю закрити його.

1.3 Постановка задачі

Для вирішення виявлених проблем у проведенні соціологічних опитувань повинна бути спроєктована серверна частина системи для їх автоматизації. Вона безпосередньо має надавати респондентам змогу переглядати та давати відповіді на опитувальники, створення яких покладається на роботу інших частин системи. Серед доступного функціоналу мають бути:

а) отримання набору опитувань:

1) розташування невеликого HTML-коду на будь-якому сайті в місці, де має відображатися набір опитувань;

2) упізнання користувача за наявними ознаками (такими як файли cookie, IP-адреса, тощо);

3) повернення пустого набору за умови несплати послуг клієнтом;

4) таргетування опитувань на користувача за створеними налаштуваннями;

5) відсіювання опитувань за іншими ознаками (такими як заборона проходити його користувачу більше одного разу);

б) генерація розмітки набору опитувань:

1) генерація трекерів подій для відстеження етапів проходження опитувань);

2) протидія шахрайським трекерам завдяки обмеженню їхньої роботи у часі а також симетричному шифруванню;

3) заміна стилів шаблону відповідно до таких, які обрані для даного набору опитувань;

4) вибір мови набору опитувань відповідно до поширеної мови в країні з якої було зроблено запит;

5) додавання додаткової логіки (як от приховування набору опитувань після повного його проходження);

За проведеним раніше аналізом предметної області важко визначити обширну цільову авдиторію користувачів системи, адже найбільш значущою її частиною є

соціологи, що проводять дослідження, чи рекламні агентства, які збирають зворотній відгук від власної авдиторії. Також частину користувачів можуть становити викладачі та студенти спеціальностей, що пов’язані зі збором даних з встановленої вибірки респондентів.

Але незважаючи на те, ким є люди що користуються продуктом, до їхньої спільної характеристики можна віднести вміння користуватися комп’ютером на середньому та вище рівні. Під цим розуміється досвід у використанні різноманітних професійних програм, як от Excel чи PowerBI. Більш того, очікується, що користувачі матимуть базові навички аналізу даних, інтерпретації результатів та представлення висновків у зрозумілій формі.

Крім того, система повинна бути масштабованою та гнучкою, щоб задовольнити різноманітні потреби користувачів, від невеликих опитувань до серйозних досліджень. Це дозволить охопити ширший спектр галузей, де збір та аналіз даних є життєво важливим, наприклад, маркетинг, соціальні науки, державний сектор та інші.

Отже важливою задачею при розробці серверу є передбачення його швидкодії та придатності до обробки значної кількості запитів на секунду. Це необхідно для забезпечення гарного користувацького досвіду респондентів та мінімальних втрат статистики через можливі затримки в мережі.

Для реалізації потрібної швидкодії важливий правильний вибір мов програмування, бібліотек, сховищ даних, принципів розробки, тощо. Саме для цього було вирішено що серверна частина має бути розроблена за допомогою мови Golang, для збору статистики повинна використовуватися сховище Kafka, а для її збереження – стовпчикова база даних Vertica. Читання інформації про опитування та їх налаштування має відбуватися з бази даних MS SQL Server. Але важливою характеристикою системи має бути незалежність від конкретних сховищ даних. Вона повинна бути достатньо абстрагованою для зміни будь-якої компоненти, і використовувати лише загальні інтерфейси.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Під час виконання кваліфікаційної роботи необхідно спроектувати високонавантажену серверну частину системи для проведення соціальних опитувань. Вхідною точкою для перегляду опитування респондентом повинне бути завантаження сайту із встановленим на ньому кодом набору опитувань, який клієнт може отримати в клієнтському веб-застосунку системи, та розмістити на власному сайті чи сайті партнера.

При завантаженні сторінки відповідний код повинен зробити запит на створювану серверну частину, відповідь від якої відобразить підібрані опитування респонденту. Розмітка шаблонів опитувальника повинна забезпечувати користувачам гарний досвід та інтерфейс, швидко завантажуватися, не блокуючи інші елементи на веб-сторінці, а також мати функціонал, що дозволить збирати інформацію про проходження швидко та без високого навантаження на мережу.

Також код опитувальника, вставлений на сайтах клієнтів, повинен бути універсальним для набору опитувань та не вимагати його заміни чи редагування при зміні налаштувань самого набору чи його зовнішнього вигляду. Розміщення самого коду не повинно вимагати від клієнта знань програмування вище базового рівня (мінімальне розуміння структури HTML-розмітки).

Зважаючи на мінімальну кількість дій, якими користувач може взаємодіяти з системою, необхідно зосередити основну увагу розробки на функціоналі серверу. До нього має належати:

- використання кешу бази даних, збереженого в пам’яті, замість постійних запитів в неї. Це в рази збільшує швидкість обробки запитів, та є звичною практикою у високонавантажених системах, як от в рекламних технологіях (ad-tech). Для підтримання актуальності даних – раз в N часу кеш оновлює інформацію;
- ідентифікація користувача на кожному запиті за набором ознак: файли cookie з ідентифікатором, User-Agent, IP-адреса, тощо;

- перевірка статусу оплати послуг користувачем, на чий набір опитувань прийшов запит;
- перевірка запиту на шахрайство (перевірка User-Agent на бота, тощо);
- таргетування опитувань на користувача за налаштуваннями (мінімальний план – таргетування за країною);
- перевірка на отримання повторного запиту на пройдене користувачем опитування. За умови що опитування дозволено проходити лише один раз – воно відсіюється;
- генерація коду опитувальника, що повертатиметься на веб-сторінку. Код складається з розмітки запитань, стилів розмітки, та Javascript-скриптів, які відповідають за перемикання запитань та надсилання статистики. Всі ці елементи обираються з шаблону, прикріпленого до набору опитувань;
- заміна стилів шаблону відповідно до обраних клієнтом. До цих стилів можуть належати кольори фону, шрифт, колір кнопок, тощо;
- генерація трекерів подій для відстеження етапів проходження опитувань. Загалом система має підтримувати 6 видів подій: перегляд набору опитувань, перегляд питання, відповідь на питання, початок проходження опитування, кінець проходження опитування, проходження всіх опитувань в наборі;
- протидія шахрайським трекерам завдяки обмеженню їхньої роботи у часі, відстеженню дублікатів, а також симетричному шифруванню. Симетричне шифрування в даному випадку має передбачати шифрування набору даних (як от ідентифікатор опитування, ідентифікатор питання, тощо), що унеможливить їхню підробку шахраєм. А для запобіганню накручуванню статистики за допомогою одного й того ж трекера, для нього має проставлятися обмежений час, протягом якого він придатний (наприклад 10 хвилин), а протягом цього часу будуть відловлюватися можливі дублікати трекерів;
- генерація тексту опитувальника відповідно до мови що є найбільш поширеною в країні, з якої було зроблено запит. Визначення геолокації респондента має відбуватися за його IP-адресою. Визначення мови для відповідної території – за

допомогою сторонніх бібліотек чи API. Потрібно передбачити правильне відображення спеціальних символів в браузері (як от апострофа);

- додавання блоку-подяки за проходження опитування;
- додавання додаткової логіки, обраної користувачем. До такої може належати функція приховування набору опитувань після повного його проходження;
- перетворення набору опитувань відповідно до типу його розташування: повноекранне, в тексті чи плаваюче поверх тексту;
- запис подій з трекерів у стовпчикову базу даних. Цей тип NoSQL БД найкраще підходить для зберігання великих наборів статистики, адже надає швидке читання та вбудований аналіз даних;

Проходження опитувань повинно добре працювати як на нових та потужних пристроях, так і на застарілих. Мінімальними системними вимогами мають бути:

а) операційна система:
   1) Windows 7 або новіша версія;
   2) macOS 10.10 або новіша версія;
   3) Ubuntu 16.04 або новіша версія;
   4) Android 5.0 (Lollipop) або новіша версія;
   5) iOS 10 або новіша версія.

б) веб-браузер:
   1) Google Chrome (остання версія);
   2) Mozilla Firefox (остання версія);
   3) Microsoft Edge (остання версія);
   4) Safari (остання версія для macOS та iOS).

в) процесор:
   1) для комп’ютерів – двоядерний процесор з тактовою частотою 1 ГГц або вище;
   2) для планшетів та смартфонів – чотириядерний процесор з тактовою частотою 1 ГГц або вище.

г) оперативна пам’ять: 1 ГБ ОЗУ або більше;

д) інтернет-з’єднання зі швидкістю завантаження не менше 1 Мбіт/с;

е) роздільна здатність екрану:

1) для комп’ютерів: 1024x768 пікселів або вище;

2) для планшетів та смартфонів: адаптивний дизайн для різних роздільних здатностей.

Також, як було і зазначено у попередньому пункті, необхідно зберегти високу продуктивність серверної частини. Тож система повинна бути спроектована таким чином, щоб мати змогу масштабуватися та одночасно обслуговувати значну кількість користувачів без погіршення швидкості роботи. Також написаний код має бути високоефективним, а отже кількість виділень пам’яті в процесі роботи має бути мінімізована. Також доцільно зменшувати розмір даних, які передаються мережею, адже це зменшить час і на їхню передачу, і вартість підтримки роботи системи.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Серверна частина системи передбачає єдину роль, що взаємодіє з нею під час роботи – користувач сайту клієнта. Під час проходження опитування його можна називати респондентом. На рисунку 4.1 зображена діаграма прецедентів для користувача з цією роллю.

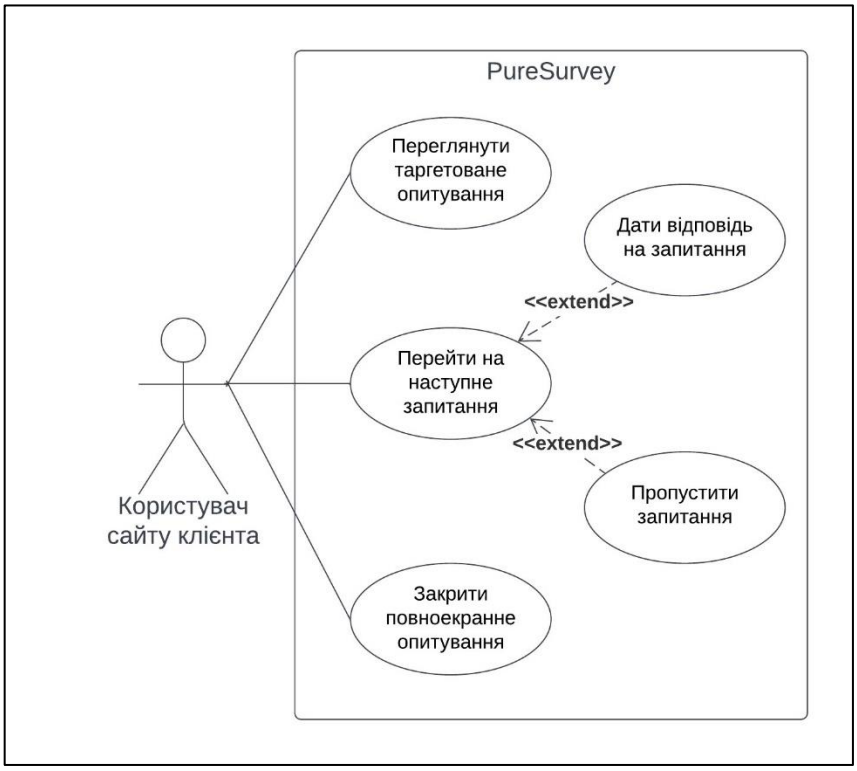

Рисунок 3.1 – Діаграма прецедентів серверної частини

Як видно з діаграми, до можливих дій користувача в серверній частині належить небагато функцій, так як ця частина керується лише даними з БД, і є самодостатньою. Але все ж до наявних можливостей можна віднести перегляд таргетованого опитування на сайті, з розміщеним клієнтом кодом, перехід по запитаннях (що передбачає їх пропуск або надання відповіді) та закриття повноекранного опитування. Додаткової інтеракції з системою респондент не має.

Щодо роботи серверної частини, для її проєктування було створено діаграму активності (див. рис. 4.2), яка надає уявлення щодо основного циклу відпрацювання запиту на отримання опитувань.

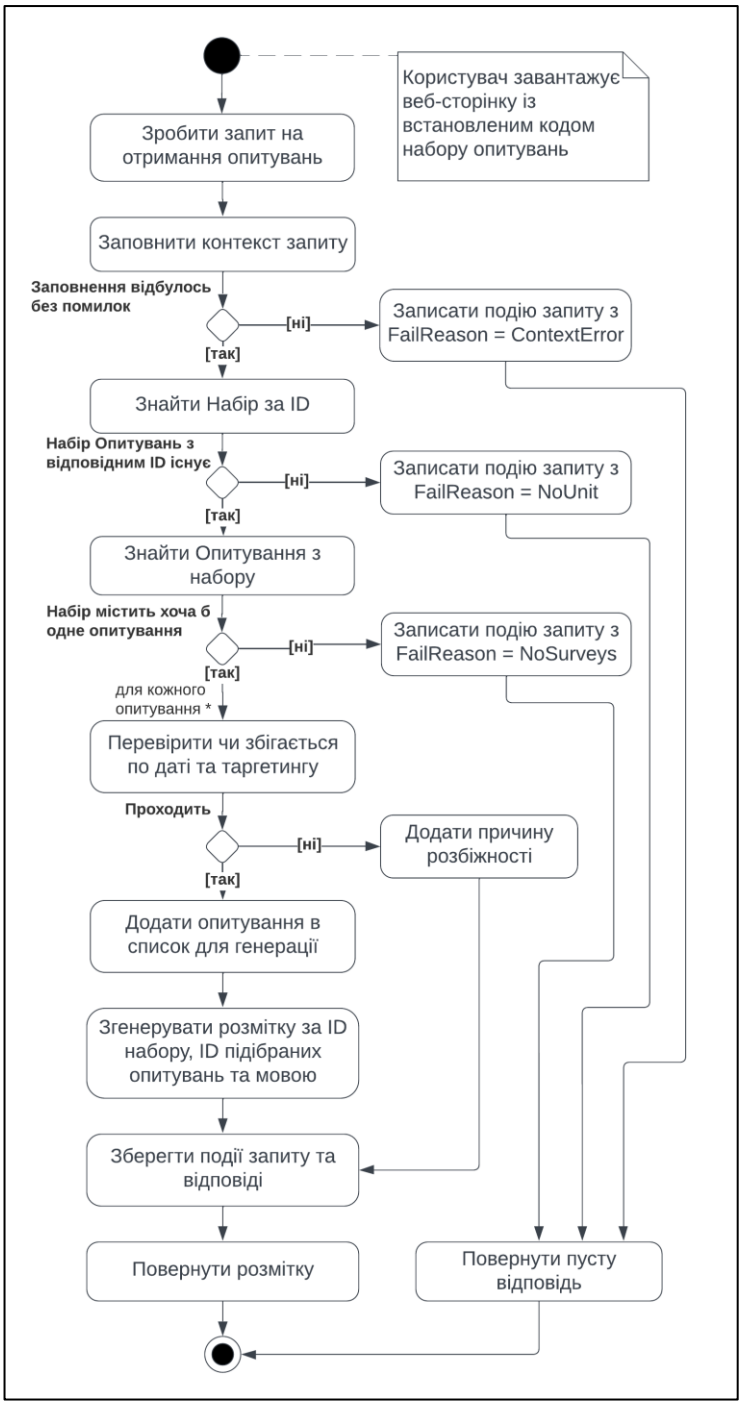

Рисунок 3.2 – Діаграма активності обробки запиту на опитування

На діаграмі помітно що основні операції при обробці запиту стосуються його валідації. Для цього, при заповненні контексту (зчитуванні даних із параметрів запиту та його заголовків), виконуються певні перевірки на його коректність, після

чого із отриманого ідентифікатора Набору Опитувань, відбувається пошук його та Опитувань, що вкладені в нього. Помилка, знайдена при якійсь з цих операцій, має власний тип, який проставляється у подію запиту, яка потім записується у базу даних.

Якщо на жодному з попередніх кроків не було виявлено помилки, то кожне з опитувань проходить таргетування та перевірку на дату, до якої воно може працювати. Таргетування в системі є перевіркою заповненого контексту на відповідність правилам, що були вказані при створенні опитування.

Всі опитування, що пройшли відповідні перевірки, підлягають генерації розмітки, яка є окремим комплексним процесом. Ті опитування, що з якоїсь причини не пройшли далі, мають бути записані в подію разом з самою причиною. Після завершення генерації, в базу даних має бути збережено подію запиту та подію відповіді, кожна з яких має власний набір даних, після чого, на сайт повертається сформована розмітка.

Створення розмітки має покладатися на окремий сервіс, проєкт структури якого було реалізовано у вигляді діаграмі пакетів, яку зображено на рисунку 4.3.

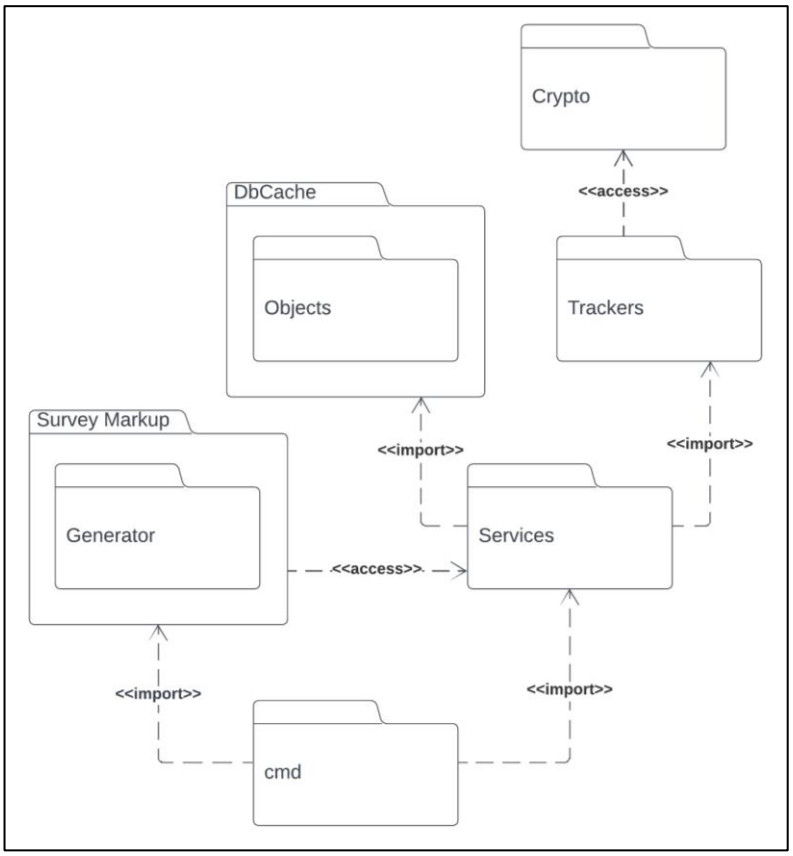

Рисунок 3.3 – Діаграма пакетів сервісу для генерації розмітки

Основним пакетом сервісу виступає cmd – точка входу у застосунок. Він імпортує 2 пакети, які містять основний функціонал: Services та SurveyMarkup. Всередині пакету Services мають інстанціюватися та зберігатися екземпляри сервісів, необхідних для роботи. Пакет cmd імпортує даний функціонал щоб передати в SurveyMarkup ці екземпляри для використання.

Зокрема, серед пакетів, імпортованих в Services, можна навести DbCache та Trackers. Перший – містить функціонал, пов'язаний з вичиткою кешу основної бази даних у пам'ять. Він містить вкладений пакет, що описує структуру об'єктів, які зберігаються в базі. Пакет Trackers же надає можливість створення трекерів подій, при генерації яких необхідна можливість шифрувати певні дані для протидії шахрайству. Це реалізується за допомогою використання функціоналу пакету Crypto.

Відповідно всі ці пакети використовуються у вкладеному в SurveyMarkup пакеті – Generator, за посередництва Services, який їх імпортує. За запитом, він створює HTML-розмітку опитувальника, використовуючи актуальні дані з кешу БД та генеруючи трекери подій.

3.2 Проєктування архітектури ПЗ

Для реалізації серверної частини системи було обрано сервісно-орієнтовану архітектуру. Цей підхід передбачає розбиття всієї функціональності на окремі сервіси, які можуть взаємодіяти один з одним через визначені інтерфейси [5, ст. 1].

Проектована система складається із трьох частин-сервісів, розділених по принципу єдиної відповідальності (Single Responsibility). До них належать:

- обробник запитів – точка доступу до системи з сайтів респондентів. Серед його функціоналу присутнє таргетування опитувань, протидія шахрайству, відсіювання дублікатів запитів, тощо;
- генератор розмітки опитувань – сервіс що відповідає за створення HTML-розмітки на основі обраних клієнтом налаштувань. Запит до нього приходить із обробника, після проходження логіки відсіювання;

- записувач статистики – сервіс, що отримує дані про події від обробника, обробляє їх у вигляд, придатний до запису у стовпчикову БД та записує в базу великими порціями, для гарної швидкодії.

Для зберігання та передачі даних між частинами використовуються три сховища даних. До них належать:

- реляційна база даних SQL Server – містить в собі усі дані про основні сутності системи, наприклад: Опитування, Зовнішній Вигляд Набору, Таргетинг;
- column-oriented база даних Vertica – слугує для зберігання великих об’ємів статистичних даних, пов’язаних із запитами та проходженням опитувань;
- розподілене сховище подій Kafka – використовується для передачі від обробника запитів до записувача статистики усіх подій, що фіксуються та зберігаються системою;

Діаграму розгортання серверної частини зображено на рисунку 4.4.

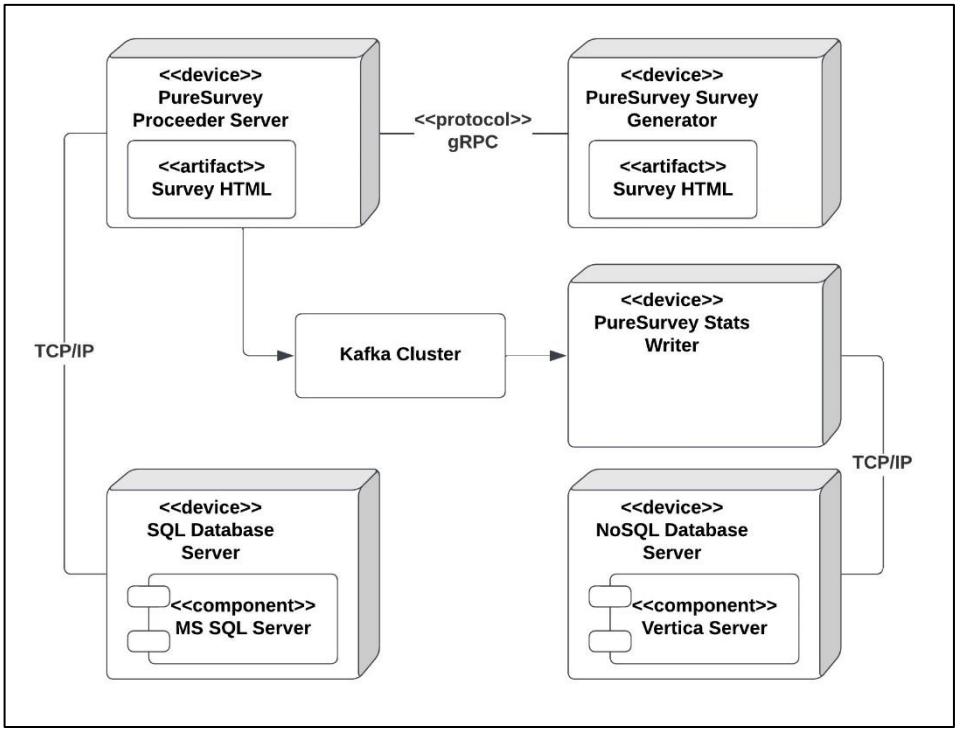

Рисунок 3.4 – Діаграма розгортання серверної частини

У процесі створення даної системи було прийнято рішення не покладатися на REST-підхід для комунікації між сервісами. Одним з основних недоліків REST є його орієнтованість на передачу даних у форматі JSON або XML, що є неефективним для великих обсягів даних або двосторонньої комунікації в режимі

реального часу. Це робить REST менш придатним для сценаріїв, які вимагають високої пропускної здатності та низької латентності, а відповідно непридатним до забезпечення вимог, встановлених у пункті 2.

Замість цього, для комунікації між обробником запитів та генератором розмітки, був використаний gRPC – сучасний підхід, заснований на HTTP/2, з певними перевагами над REST. На відміну від REST, він використовує бінарний протокол, заснований на Protocol Buffers, що забезпечує компактніше представлення даних [6], відповідно кращу продуктивність.

Незважаючи на те, що gRPC працює поверх HTTP/2, він уникає деяких недоліків, притаманних традиційному HTTP. Зокрема в HTTP кожен запит ініціює нове з’єднання між клієнтом та сервером, яке закривається після отримання відповіді. Такий підхід призводить до значних накладних витрат, особливо в середовищах з високою інтенсивністю трафіку, оскільки встановлення та закриття з’єднань є ресурсномісткими операціями.

На відміну від цього, gRPC використовує довготривалі HTTP/2 з’єднання, що дозволяє передавати численні запити та отримувати відповіді через одне стійке з'єднання. Це зменшує накладні витрати та підвищує ефективність передачі даних.

Приклад створення ініціалізації єдиного екземпляру з’єднання на весь життєвий цикл обробника запитів наведено далі:

```
func (s *Service) Init() error {
	var err error
	s.conn, err = grpc.Dial(s.srvGeneratorAddr, grpc.WithTransportCredentials(insecure.NewCredentials()))
	if err != nil {
		log.Fatalf("could not connect to SurveyMarkupGenerator: %v", err)
	}

	s.client = model.NewSurveyMarkupGeneratorClient(s.conn)
	return nil
}
```

Також, зважаючи на потребу підтримувати високу пропускну здатність та забезпечувати надійність збереження даних, обробник запитів не передає отримані події безпосередньо записувачу статистики, натомість вони спершу потрапляють до черги повідомлень Kafka.

Використання Kafka як проміжної ланки між обробником запитів та системою збереження статистики має кілька ключових переваг. По-перше, це забезпечує відмовостійкість та надійну доставку повідомлень, оскільки Kafka гарантує, що жодна подія не буде втрачена, навіть у випадку тимчасової недоступності або збою одного з компонентів системи [7].

Крім того, Kafka підтримує концепцію груп споживачів (consumer groups), що дозволяє кільком сервісам одночасно споживати події з однієї черги. Це відкриває можливості для горизонтального масштабування системи збереження статистики, оскільки кілька екземплярів сервісу можуть працювати паралельно, розподіляючи навантаження між собою та забезпечуючи підвищену пропускну спроможність.

Використовуючи групи споживачів, кожен екземпляр сервісу записувача статистики буде отримувати різні партії подій з черги Kafka, запобігаючи дублюванню обробки та забезпечуючи збалансоване навантаження. Такий підхід не лише покращує продуктивність системи, а й забезпечує відмовостійкість, оскільки у випадку відмови одного з екземплярів записувача статистики, інші екземпляри продовжуватимуть споживати та обробляти події з черги Kafka без втрати даних.

Під час розробки та тестування даної системи, для зручності та спрощення процесу, всі її компоненти запускалися на одній локальній машині, використовуючи Docker-контейнер. Проте, система була спроектована з урахуванням можливості розподіленого розгортання її компонентів на різних серверах або кластерах. Наприклад, обробник запитів, шина Kafka, записувач статистики та інші частини можуть бути розподілені на різні вузли, залежно від вимог до продуктивності та фактичного навантаження.

3.3 Проєктування структури зберігання даних

Збереження статистики, яка включає в себе запити на опитування, відповіді на ці запити та проходження самих опитувань, відбувається в базі даних зі стовпчиковим зберіганням (column-oriented) Vertica. У загальному, стовпчикові СУБД відрізняються від традиційних реляційних самим способом фізичного збереження даних. У реляційних СУБД дані зберігаються у вигляді рядків, які

можна ідентифікувати за допомогою первинного ключа. В стовпчикових же – дані зберігаються у вигляді окремих колонок.

Vertica, та інші СУБД цього типу, є більш ефективними для операцій читання, оскільки для виконання запиту їм достатньо завантажувати лише відповідні колонки, а не цілі рядки [8]. Це скорочує обсяг зчитувань даних з диску, що підвищує продуктивність системи.

Також ці СУБД часто використовують більш ефективні методи стиснення даних, оскільки значення в одній колонці, як правило, мають більш однорідний характер, ніж значення в різних рядках таблиці. Це дозволяє зменшити розмір даних, що зберігаються на диску, та покращити швидкодію операцій читання.

Зазвичай, для зберігання даних в стовпчикових базах даних, використовуються так звані «плоскі», тобто денормалізовані, таблиці. Сама їхня суть допомагає допомагає уникнути операцій JOIN під час звернення до даних, що забезпечує дуже швидке читання. Хоча денормалізація може призвести до надмірності даних, відсутності їх цілісності та ускладнити процес оновлення, ці недоліки компенсуються перевагами у швидкості доступу до даних, що є критично важливим для аналітичних систем.

Кожна подія, отримана обробником, записується в базу даних як окремий рядок (набір стовпців). Частина стовпців називаються вимірами (dimensions) і відповідають за такі ознаки події, як ідентифікатори сутностей, з якими вона пов'язана, час, коли подія була зафіксована, тощо. Інша частина стовпців називається метриками і містить значущі дані, такі як кількість подій, що відбулися, час, витрачений на обробку події, та інше.

Таблиці структуровані таким чином, щоб актуальну статистику за будь-яким виміром можна було отримати, згрупувавши за ним записи та агрегувавши їх за потрібними метриками (найчастіше за допомогою операції суми).

Для зберігання інформації про події системи, було створено дві основні таблиці: tracking_stats і completion_stats, та дві допоміжні: tracking_stats_flat і completion_stats_flat. Приклад запиту для створення tracking_stats наведено далі:

```
CREATE TABLE pure_survey.tracking_stats
(
```

```
    date date,
    "timestamp" timestamp,
    hour int NOT NULL,
    unit_id int NOT NULL,
    geo varchar(16),
    lang varchar(8),
    gender varchar(1),
    unit_requests int,
    unit_responses int,
    unit_views int,
    unit_ends int,
    survey_id array[int8] NOT NULL,
    survey_responses array[int8],
    mismatched_survey_responses array[int8],
    mismatched_survey_reason array[int8],
    survey_starts array[int8],
    survey_ends array[int8]
)
PARTITION BY tracking_stats.date::DATE
GROUP BY CALENDAR_HIERARCHY_DAY(tracking_stats.date::DATE, 2, 2);
```

Помітно що структура таблиці містить в собі стовпці з типом array, адже подія повернення набору опитувань може містити в собі кілька цих опитувань. Так само як і подія відповіді на запитання може включати в себе декілька варіантів відповіді. Отже ці дані необхідно зберігати в такому ж, первинному, вигляді, адже записування події за допомогою кількох рядків у підсумку дасть спотворені дані для тих вимірів, які буде продубльовано.

Але використання масивів не дає забезпечити можливість швидкого читання з таблиць, адже операції GROUP BY чи WHERE будуть виконуватися неефективно у випадку таких стовпців. Отже, за відсутності вбудованих у Vertica інструментів, які допомогли б подолати цю проблему (як наприклад оператор ARRAY JOIN в СУБД Clickhouse), було вирішено записувати згорнуту та розгорнуту інформацію в окремі таблиці (як от tracking_stats та tracking_stats_flat).

Також для пришвидшення читання даних за різними вимірами, в базі даних були створено декілька проєкцій. Проєкція у Vertica це спосіб зберігання копії даних для пришвидшення їх обробки. Проєкція схожа на перегляд (VIEW) у багатьох інших СУБД, але відрізняється тим, що дані перегляду обчислюються щоразу, коли на них надходить запит, а дані проєкції зберігаються на диску у фізичному вигляді.

Проєкції зберігають дані у форматі, оптимізованому для конкретних типів запитів, наприклад, групування за певними вимірами. З них Vertica автоматично

вибирає найбільш ефективну проєкцію для виконання запиту, що значно підвищує продуктивність.

Прикладом створеної проєкції можна навести completion_stats_flat_gender_proj, запит для створення якої наведено далі:

```
CREATE PROJECTION pure_survey.completion_stats_flat_gender_proj
(
 date,
 question_id,
 gender ENCODING RLE,
 option_id,
 answered
)
AS
 SELECT completion_stats_flat.date,
        completion_stats_flat.question_id,
        completion_stats_flat.gender,
        completion_stats_flat.option_id,
        sum(completion_stats_flat.answered) AS answered
 FROM pure_survey.completion_stats_flat
 GROUP BY completion_stats_flat.date,
          completion_stats_flat.question_id,
          completion_stats_flat.gender,
          completion_stats_flat.option_id;
```

Ця проєкція зберігає в собі результат агрегації даних таблиці completion_stats_flat з групуванням за стовпцями question_id, option_id та gender, що значно зменшує загальну можливу кількість записів (вона становить K*L*M*3, де K – кількість днів, в які записана статистика, L – кількість запитань, M – кількість варіантів відповідей, та 3 – кількість доступних опцій гендеру).

Варто підмітити використання ENCODING RLE для стовпця gender. Цей оператор явно вказує Vertica, який тип стискання чи кодування даних потрібно використовувати для того чи іншого стовпця. RLE – це тип кодування, який перетворює послідовність однакових значень, у рядок, який містить саме значення і кількість його входжень [9]. Його варто застосовувати для стовпців, що мають невелику кількість можливих значень, та якщо він є відсортованим. Гендер в наведеній проєкції добре підходить під це визначення.

Для перевірки того що оптимізатор Vertica використовує створену проєкцію тоді коли це необхідно, було перевірено план виконання запиту, який явно має читати дані з неї. Запит для перегляду плану наведено далі:

```
explain select question_id, option_id, gender, sum(answered) as
answered
```

```
from pure_survey.completion_stats_flat
group by question_id, option_id, gender;
```

Скриншот плану виконання наведено на рисунку 4.5.

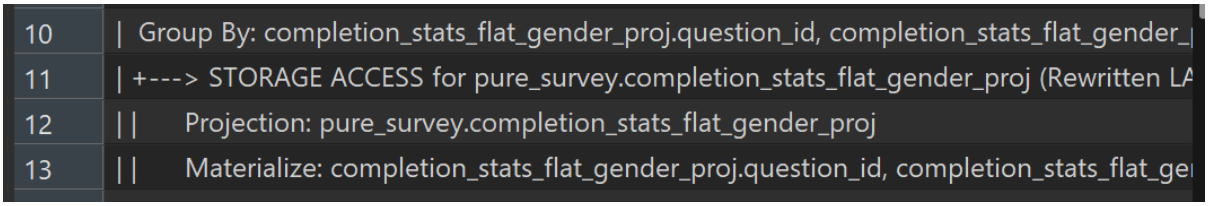

Рисунок 3.5 – Частина плану виконання запиту на статистику

В результаті, було отримано задовільний результат, адже при матеріалізації стовпців було використано необхідну проєкцію, що видно на 13-му рядку на зображенні.

3.4 Створення UI/UX системи

Під час розробки зовнішнього вигляду та інтерфейсу керування опитування, було проведено аналіз досвіду користувача інших популярних опитувальників на ринку. Це дозволило зрозуміти загальноприйняті практики та очікування користувачів, а також виявити потенційні проблеми та обмеження.

Серед вимог до зовнішнього вигляду та інтерфейсу опитування не було виявлено нічого надзвичайно складного або незвичного. Більшість вимог були досить стандартними та зосереджувалися на забезпеченні зручності використання, доступності та інтуїтивності для респондентів.

Серед типових вимог, які висуваються до зовнішнього вигляду та інтерфейсу опитування було обрано:

- адаптивний дизайн, для коректного відображення на різних пристроях: комп'ютерах чи смартфонах;
- інтуїтивна навігація, для легких переходів між питаннями та окремими опитуваннями;
- візуальна привабливість, яка хоч і не є критичною, але сприяє кращому залученню респондентів та підвищенню задоволення від процесу опитування;
- відсутність можливості зробити щось неправильно, із запобіганням повторних відповідей.

Ще однією важливою вимогою до дизайну опитувальника є наявність елементів, які можна кастомізувати з кабінету клієнта. До таких можна віднести кольори фону чи шрифту, сам шрифт або колір елементів керування.

З дотриманням наведених правил було створено мінімальний макет опитування, переглянути який можна на рисунку 4.6. На ньому зображено необхідні елементи керування у вигляді радіокнопок, кнопки для переміщення до наступного чи попереднього запитання, тексти самого запитання та відповідей, а також елементи дизайну.

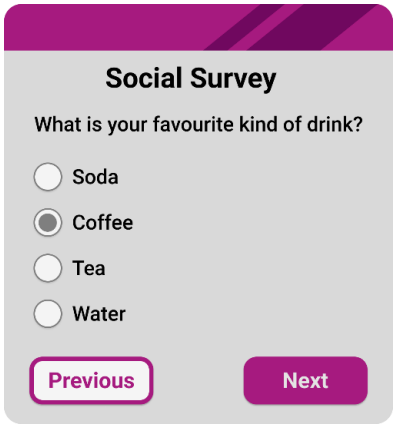

Рисунок 3.6 – Макет розмітки опитування

Для забезпечення його адаптивності варто передбачити прості правила, за якими розмітка може змінюватися. До таких можна віднести необхідність збереження елементів дизайну подібними до первинного варіанту, можливість зменшення відступів між елементами, якщо вони здаються непропорційними, та зміну самої верстки, якщо елементи стають занадто малими (див. рис. 3.7).

Рисунок 3.7 – Макет розмітки опитування для мобільних екранів

У створеному макеті було вирішено змінити взаємне розташування кнопок з рядку на стовпець, що надало достатній розмір для взаємодії з ними.

Важливою частиною розмітки опитувальника є збирання з нього статистики без впливу на досвід користувача (UX), тобто без підвисань самої анкети чи інших компонентів на сайті.

Для відстеження самого факту успішної доставки та відображення набору опитувань було вирішено використати «піксель». «Піксель» (зокрема в рекламних технологіях) – це HTML-тег <img/>, розміром 1x1 піксель, в атрибуті «src» якого знаходиться трекер, що відстежує подію завантаження цього тегу. Зазвичай передбачається що завантаження цього тегу супроводжує завантаження розмітки, яка розташована поруч з ним, з чого можна робити висновок про показ розмітки.

Але також завантаження зображень, за специфікацією HTML, відбувається незважаючи на те, чи є воно видимим для користувача [10]. Отже розташування окремих «пікселів» на окремих блоках запитань не підходить для відстеження події перегляду запитання, адже всі вони будуть завантаженні при первинному відображенні опитувальника. Отже такі події вимагають використання інших засобів для надсилання трекерів.

Таким засобом було обрано інструмент HTML, що доступний у всіх останніх версія браузерів – Navigator.sendBeacon(). Він гарантує надійне асинхронне надсилання запиту на сервер без відстеження відповіді на нього та без впливу на завантаження сторінки, що робить його досить легким та непомітним при взаємодії. Зокрема він рекомендується для завантаження невеликих аналітичних подій на сервер.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Рішення для продуктивності системи

Для забезпечення швидкодії системи було використано принципи, що допомагають зменшити кількість перевиділень пам’яті та операцій вводу/виводу. До таких можна віднести:

- використання in-memory кешу бази даних з періодичним оновленням інформації в ньому. Це дозволяє зберігати всі, необхідні для роботи сервісу, дані в оперативній пам’яті, що значно прискорює доступ до них порівняно з постійним зверненням до бази даних на диску;
- використання пулів об’єктів, які вимагають частих створень. До них належать JSON-парсери, об’єкти типу strings.Builder, парсери заголовку User-Agent, тощо. Це допомагає повторно використовувати створені об’єкти замість виділення пам’яті на нові. Реалізацію пулів було виконано за допомогою вбудованої в Golang структури sync.Pool, яка надає можливість складати та діставати з неї об’єкти, а також автоматично їх збуває (dispose), якщо вони певний час не використовуються;
- використання gzip-стискання розмітки опитувальника перед поверненням її в браузер користувача. Це дає змогу заощадити час, який займатиме завантаження опитувальника у респондента, і додатково зменшити витрати проєкту на мережевого постачальника. Так невелике опитування на 4 запитання вимагає передачі всього 2.4 кілобайтів замість 7.8 Кб без стискання, що становить зменшення об’єму в 3.25 разів. А анкета на 6 запитань, що займає вже 8.8 Кб, зменшується в 3.43 рази, що показує гарне масштабування gzip-стискання для HTML-файлів, адже вони містять багато повторюваних символів;
- перетворення HTML-розмітки в компактну – minifying. До цього процесу входить видалення непотрібних пробілів та символів переносу рядка, видалення булевих значень атрибутів, видалення коментарів, та переведення можливих значень до нижнього регістру (це надає можливості для кращого gzip-

стискання). Це не змінює функціонал опитувальника, але дає змогу зекономити близько 30% його розміру на передачі;

- використання каналів для ефективної комунікації між горутинами, замість блокуючих операцій обробки повідомлень.

Ці та інші оптимізації дозволяють системі ефективно обробляти великі обсяги запитів та забезпечувати високу швидкодію роботи з можливістю масштабування.

4.2 Провайдер сервісів

Для забезпечення п’ятого принципу SOLID, принципу інверсії залежності, всі складові застосунків були побудовані на абстракціях за допомогою використання інтерфейсів сервісів, а не їхніх конкретних імплементацій.

Зазвичай впровадження залежностей (dependency injection) передбачає наявність DI-контейнера, який керує життєвим циклом об’єктів протягом роботи застосунку. Але в Golang немає стандартних засобів для його використання, а сторонні бібліотеки зазвичай мають неочевидний синтаксис.

Зважаючи на те, що більшість сервісів системи можуть існувати в єдиному екземплярі, та самих сервісів існує порівняно невелика кількість, для подолання проблеми було вирішено створити провайдер сервісів, який буде єдиним місцем, що ініціалізуватиме потрібні реалізації інтерфейсів, та надаватиме доступ до кожного з ним за допомогою getter-методів.

Приклад створення продюсера подій наведено в коді далі:

```
func NewProvider(appConfiguration *configuration.AppConfiguration) servicescontracts.IServiceProvider {

	...
	eventProducer := kafka.NewProducer(appConfiguration.Events)
	err = eventProducer.Init()
	if err != nil {
		log.Fatalf(err.Error())
	}	...
	provider := &Provider{
		...
		eventProducer:  eventProducer,
		...
	}
```

```
	return provider
}
```

При створенні продюсера, йому передається конфігурація програми, яка містить зібрані в одному місці налаштування всіх сервісів. Якщо при ініціалізації створеного об’єкту виникає помилка, вона відразу ж повертається з конструктору.

Витягувати створені імплементації з провадеру можна за допомогою простих методів, що повертають інкапсульований в провайдері об’єкт, приховуючи його конкретний тип за інтерфейсом. Приклад такого методу наведено в наступному уривкові коду:

```
func (p *Provider) GetEventContextFiller() contextcontracts.IRequestFiller {
	return p.eventContextFiller
}
```

В кінці життєвого циклу провайдера необхідно викликати його метод Dispose(). В ньому виконується закриття ресурсів (зокрема з’єднань) коли вони більше не потрібні. В мові Golang зазвичай доречно викликати такі функції відразу після створення екземпляру класу, з додаванням команди defer, яка відкладає її виклик до завершення виконання функції, в якій було його зроблено.

Вигляд методу Dispose() в провайдері обробника запитів наведено далі:

```
func (p *Provider) Dispose() error {
	if p.surveyMarkupService != nil {
		err := p.surveyMarkupService.Close()
		if err != nil {
			return err
		}
	}

	if p.eventProducer != nil {
		p.eventProducer.CloseConnection()
	}

	return nil
}
```

Тут обов’язково проводиться перевірка потрібних об’єктів на nil, після чого на них викликаються відповідні методи для очистки ресурсів, що вони використовують.

4.3 Створення розмітки опитувальника

Для генерацію HTML-коду опитувальників на основі шаблонів використовується структура Generator пакету surveymarkup. Основна його мета

класу – створення динамічних опитувальників з урахуванням локалізації, різних варіантів питань та відповідей, а також трекерів для відстеження дій користувачів.

Діаграму активності основних етапі генерації наведено на рисунку 4.8.

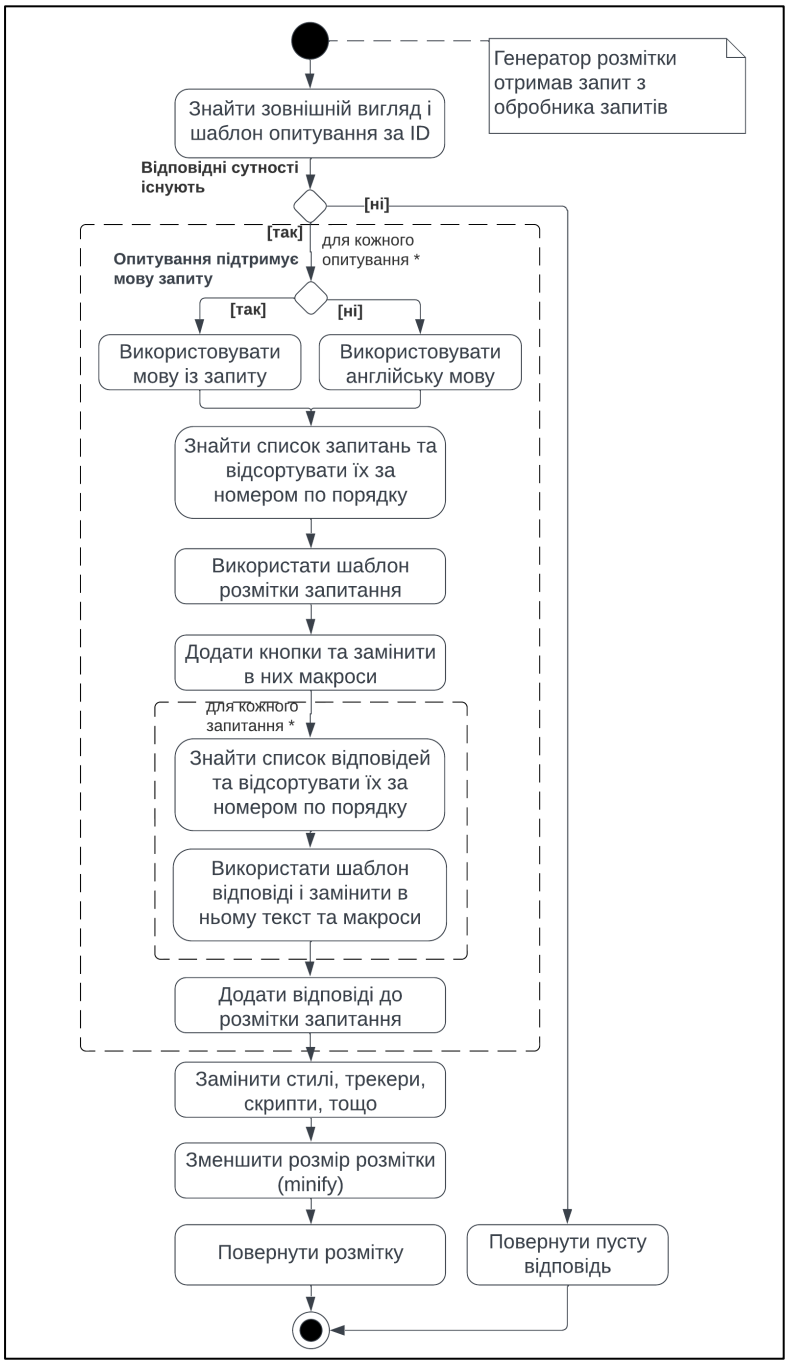

Рисунок 4.1 – Макет розмітки опитування для мобільних екранів

Код починається з отримання шаблону і зовнішнього вигляду для даного набору опитувань з бази даних. Для кожного опитування створюється HTML-код, що включає питання та варіанти відповідей, розташовані по порядку.

Далі до коду додаються трекери для відстеження переглядів питань, відповідей, початку та завершення опитування. Всі частини HTML і JavaScript коду

об'єднуються в кінцеву розмітку, яка мінімізується для зменшення розміру та покращення продуктивності.

Клас використовує внутрішні сервіси для отримання необхідних даних та інших потреб. До них належать:

- DbRepo – репозиторій для доступу до кешованих даних з БД, таких як опитування, питання, варіанти відповіді, шаблони тощо;
- TrackersGenerator – генератор трекерів для відстеження дій користувачів, таких як перегляди питань, відповіді та завершення опитування;
- StringBuilderPool – пул для ефективного створення рядків, використовується для об'єднання частин HTML-коду;
- AppConfiguration – конфігурація додатка, що містить налаштування, такі як URL-адреси обробника подій;
- Minifier – сервіс для мінімізації HTML та JavaScript-коду для зменшення розміру кінцевих сторінок.

Цей клас забезпечує динамічну генерацію опитувальників, дозволяє легко адаптуватися до різних конфігурацій і мовних налаштувань, а також забезпечує ефективне відстеження дій користувачів для подальшого аналізу.

4.4 Шифрування трекерів для відстеження подій

Шифрування трекерів для протидію шахрайству відбувається через клас Encryptor. Він містить у собі єдиний публічний метод, який приймає вказівник на об'єкт Tracker, що містить всі дані, пов’язані із подією, та повертає зашифрований рядок.

Спочатку даний метод формує рядок для шифрування за допомогою допоміжного методу getStringFromEvent(). Потім цей рядок шифрується за допомогою GCM шифрування, яке забезпечує допоміжний метод з пакету crypto, після чого кодується у base64, придатному для передачі в URL. Для шифрування використовується ключ, отриманий з конфігурації застосунку, де він зберігається у вигляді рядку шістнадцятирічних символів (hex).

Метод getStringFromEvent формує рядок, який містить інформацію про подію. Так як менший розмір трекера означає менше шансів що він не надійде на сервер за умови слабкого інтернету в респондента, було вирішено не серіалізувати структуру в JSON, адже він додає багато символів, що є зайвими, коли структура рядку є відомою та незмінною. До таких можна віднести назви полів, лапки навколо рядкових значень, квадратні дужки для масивів, тощо.

Тож для серіалізації трекетів було визначено такі правила:

- кількість полів є обмеженою, а їхня послідовність – сталою;
- значення полів розмежовується комою;
- якщо тип поля – масив, його елементи розмежовуються крапкою з комою;
- якщо тип поля – словник, його ключі відмежовуються від значень двокрапкою, самі значення крапкою з комою, а окремі входження в словник – знаком слеш.

Серіалізована інформація включає в себе тип події, ідентифікатор набору опитувань, термін дії трекеру та ідентифікатори опитувань, запитань та відповідей, які можуть бути валідними у цьому трекері.

Код методу наведено далі:

```
func (e *Encryptor) getStringFromEvent(event *model.Tracker) string {
	params := [5]string{}

	params[0] = strconv.Itoa(int(event.EventType))
	params[1] = strconv.Itoa(event.UnitId)
	params[2] = strconv.FormatInt(event.ValidTo, 10)

	if event.ValidSurveys != nil && len(event.ValidSurveys) > 0 {
		validSurveysString := join(event.ValidSurveys, ";")
		params[3] = validSurveysString
	}

	if event.ValidQuestions != nil && len(event.ValidQuestions) > 0 {
		validQuestionsString := join(event.ValidQuestions, ";")
		params[4] = validQuestionsString
	}

	if event.ValidQuestionsWithAnswers != nil &&
len(event.ValidQuestionsWithAnswers) > 0 {
		var validQuestionsWithAnswers []string
		for key, val := range event.ValidQuestionsWithAnswers {
			questionWithAnswers := fmt.Sprintf("%v:%v", key,
join(val, ";"))
```

```
                    validQuestionsWithAnswers                    =
append(validQuestionsWithAnswers, questionWithAnswers)
                }

                params[5] = strings.Join(validQuestionsWithAnswers, "/")
            }

            return strings.Join(params[:], ",")
        }
```

Ідентифікатори валідних сутностей використовуються в цьому рядку, з тієї причини що клієнтська частина опитувальника дещо змінює сам трекер, додаючи до нього ідентифікатори сутностей для яких відбулася подія. Це дає змогу створити єдиний трекер для події перегляду запитання, а не створювати їх 10 чи 20 штук, чим перевантажуючи код опитувальника. Відповідно шифрування ID валідних сутностей дасть змогу перевірити чи є прикріплений ID серед них, і чи можна вірити такому трекеру.

4.5 Робота записувача статистики

Записувач статистики є швидкою в роботі та легкою частиною системи. Він збирає події з шини Kafka, перетворює їх у формат, придатний для запису у Vertica, та складає їх туди великими порціями.

На початку виконання програми відбувається ініціалізація клієнтів Kafka, та Vertica. Підключення до них відбувається в кілька спроб, кількість яких визначається в конфігурації застосунку. Також клієнт споживача Kafka отримує з конфігурації інформацію про назву його групи споживачів (щоб мати можливість горизонтально масштабувати запис статистики за допомогою розгортання додаткових копій цього сервісу) та назви топіків, на які він має підписатися.

Після ініціалізації всіх клієнтів, EventConsumer починає збирати всі повідомлення, що містяться в Kafka. За умови отримання помилки, наприклад, неможливості прочитати повідомлення через його пошкодження, EventConsumer записує помилку в логи та переходить до наступного повідомлення, щоб не блокувати весь процес обробки подій. Якщо ж повідомлення було прийнято успішно, EventConsumer складає його в канал Messages.

З каналу Messages повідомлення споживаються окремою горутиною MessagesProceeder, яка відповідає за їх перетворення у формат, придатний для запису у Vertica. Це включає такі операції, як розбиття повідомлень на окремі поля, перетворення типів даних тощо. Перетворені повідомлення групуються у великі пакети для ефективного запису у Vertica. Приклад методу для обробки події пов'язаної з проходженням опитування наведено далі:

```
func (p *Proceeder) proceedCompletionEvent(message []byte) {
	if p.completionEventsBuffer.Len() == 0 {
		p.completionEventsBuffer.Write(completionEventsHeader)
	}

	event := &pb.CompletionEvent{}
	err := proto.Unmarshal(message, event)
	if err != nil {
		fmt.Print(err.Error())
		return
	}

	timestamp := time.Unix(event.Timestamp, 0)

	writeDate(p.completionEventsBuffer, timestamp)
	writeSeparator(p.completionEventsBuffer)

	writeTimestamp(p.completionEventsBuffer, timestamp)
	writeSeparator(p.completionEventsBuffer)

	writeInt(p.completionEventsBuffer, timestamp.Hour())
	writeSeparator(p.completionEventsBuffer)
	...
	if int(event.EventType) == int(enums.ETQuestionView) {
		writeString(p.completionEventsBuffer, "1")
	} else {
		writeString(p.completionEventsBuffer, "0")
	}
	writeEndLine(p.completionEventsBuffer)

	p.completionEvents++

	if p.completionEvents ==
p.config.DbConfiguration.WritingBatchSize {
		p.dbClient.BulkInsert(p.completionEventsBuffer,
"pure_survey.completion_stats")
		p.completionEvents = 0
	}
}
```

Отримане повідомлення форматовано за допомогою Protobuf, а отже потребує десеріалізації в об'єкт. Якщо при десеріалізації виникла помилка, наприклад якщо воно було серіалізовано з іншої структури даних, то таке повідомлення вважається хибним, та не обробляється.

Якщо ж із повідомленням все добре, то воно підлягає запису у буфер байтів, що лежить в обробнику повідомлень. Буфер байтів – це стандартна структура Golang, яка добре підходить для подальшого складання даних у Vertica, адже реалізує інтерфейс io.Writer, що дозволяє передавати дані у вигляді потоку.

Події записуються в буфер у csv форматі, тобто окремі її поля розділяються комами, а окремі події розділяються символом нового рядку (line feed). Якщо перед записом події буфер є пустим, в нього необхідно вставити csv-заголовки, які повинні збігатися з назвою стовпців таблиці. Це знадобиться для автоматичного визначення необхідних стовпців при записі таблицю без необхідності їх явного вказування.

Коли в буфер було складено N рядків (в поточних налаштування проєкту N=10000) обробник повідомлень починає процес запису даних у Vertica. Для швидкої вставки великого об’єму даних в базу, використовується вбудована команда COPY. Вона приймає дані з stdin потоку, який натомість можна підмінити будь-яким іншим, а в даному випадку – сформованим буфером байтів.

Після збереження даних StatsWriter очищує буфер та продовжує свою роботу. Такі буфери існують для кожної таблиці, в яку відбуватиметься запис, відповідно запис у буфери також відбувається в окремих методах, за окремою логікою.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розуміючи важливість забезпечення високої продуктивності та стабільності застосунку, під час розробки серверної частини було приділено окрему увагу його тестуванню навантаженням (load testing). Метою було гарантувати, що система здатна витримувати значні навантаження, забезпечуючи при цьому прийнятний рівень продуктивності та надійності.

На етапі планування даного тестування було визначено мінімальні вимоги до системи. Зокрема, веб-сервер, розгорнутий в єдиному екземплярі, повинен підтримувати обробку 1000 запитів на секунду, при цьому максимально допустима кількість помилкових відповідей не має перевищувати 5%, а медіанний час обробки одного запиту не повинен бути більший за 400 мілісекунд.

Тестування відбувалося з використанням інстументу k6, який надає значну кількість засобів для проведення різноманітних перевірок. Зокрема, для проведення мінімального тесту отримання розмітки опитувань, було визначено налаштування цього тесту, в яких зазначена кількість одночасних користувачів: 400, та час навантаження: 10 секунд. Код тесту наведено далі:

```
import http from 'k6/http';

export const options = {
  vus: 400,
  duration: '10s',
};

export default function () {
  http.get('http://localhost:5000/unit?unitId=1');
}
```

Вивід k6 після завершення виконання тестування наведено на рисунку 5.1.

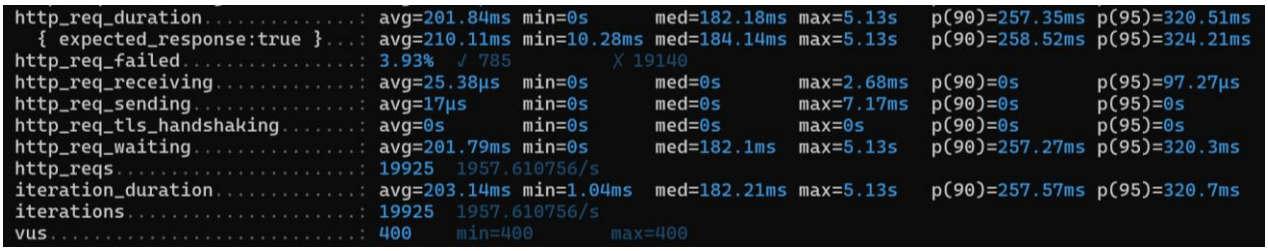

Рисунок 5.1 – Вивід інструменту k6 після тестування системи навантаженням.

Результатами тесту стало виконання 19925 запитів за 10 секунд, 3.93% з яких отримали помилку. Медіанний час відповіді на запит склав 182.18 мілісекунд. Це

задовольняє всі встановлені вимоги, та перевищує необхідну кількість запитів на секунду (QPS) та час відповіді майже в 2 рази, що означає можливість переглянути ці вимоги, та встановити жорсткіші рамки, для забезпечення якості системи.

Незважаючи на важливість тестування навантаженням для забезпечення якості високопродуктивної серверної частини, мануальне тестування відіграє важливу роль у її розробці, адже дозволяє перевірити функціональність системи з точки зору кінцевого користувача, імітуючи реальні сценарії використання. Це допомогло виявити потенційні проблеми, пов'язані з користувацьким інтерфейсом, зручністю використання та загальним досвідом роботи з системою, які непомітні при використання самих лише автоматизованих тестів.

До мануального тестування взаємодії респондента при проходженні опитувальника входила перевірка роботи його елементів керування та власне відображення анкети на сайті.

Опис тесту:	Проходження опитування		
Власник тесту:	Масенко Ярослав Володимирович		
Дата створення:	17.05.2024		
Мета тесту:	Перевірити коректність проходження опитування респондентом		
Пристрій:	Dell Inspiron 15 3000, Intel Core i5-8265U, 8 Гб DDR 4		
Операційна система та браузер:	Windows 10 Pro (64 bit), Google Chrome (версія 92.0.4515.159)		
Завантаження опитування			
№	Опис випадку	Очікуваний результат	Висновок
1	Користувач відкриває веб-сторінку з кодом набору опитувань	Усі питання, та стилі відображаються правильно. Опитувальник виглядає зручним для читання	Опитувальник правильно відображений
2	Клієнт змінює колір кнопки набору опитувань	При завантаженні сторінки через 1 хвилину після змін, колір кнопки збігається з налаштованим	Стилі опитувальника успішно змінені
4	Користувач намагається маніпулювати трекерами, надсилаючи повторний трекер перегляду набору	Система виявляє дублікат події та записує подію з приміткою про дублікат	Шахрайські дії успішно заблоковані

	опитувань через 5 секунд		
5	Користувач відкриває сайт з іншої країни, мова якої була додана в переклади опитування	Опитувальник автоматично відображається мовою, що відповідає геолокації користувача	Мова опитувальника успішно підібрана
6	Користувач завершує опитування, в якому клієнт встановив текст-подяку за проходження	Користувач бачить повідомлення з подякою за участь	Блок-подяку успішно відображено
7	Користувач завершує всі питання в наборі, в налаштуваннях якого виставлено автоматичне приховування	Набір опитувань більше не відображається на сторінці, забезпечуючи чистоту інтерфейсу	Опитувальник успішно прихований
8	Користувач заходить на сторінку з встановленим опитувальником з повноекранним форматом	Опитувальник відображається відповідно до обраного формату, зберігаючи зручність користування та надаючи можливість себе закрити	Формат опитувальника успішно працює
10	Користувач відкриває опитувальник на смартфоні	Опитувальник стає вужчим, але все ще придатним для проходження та зручним для читання	Опитувальник успішно адаптовано
11	Користувач бачить текст опитувальника зі спеціальними символами (апострофами та діакритичними знаки)	Спеціальні символи відображаються коректно, забезпечуючи правильне розуміння та читабельність тексту	Спеціальні символи успішно оброблені
12	Користувач переходить між питаннями, обираючи відповіді	Перемикання між питаннями та надсилання відповідей працюють плавно, без візуальних помилок та помилок в консолі	Функції перемикання та надсилання працюють
13	Користувач намагається повторно використати трекери після завершення	Трекери не працюють після встановленого часу, а записуються в	Обмеження часу працює успішно

	встановленого часу їх дії (через 40 хвилин).	статистику з приміткою про вичерпаний час	

Даний список тестів не є вичерпним для забезпечення якості бізнес-логіки всієї системи, але дає можливість переконатися у гарному користувацькому досвіді респондентів при проходженні опитувань.

ВИСНОВКИ

Впровадження сучасних цифрових інструментів для проведення соціологічних досліджень є важливим кроком у забезпеченні ефективності та швидкодії цих процесів. Розроблена серверна частина програмної системи для проведення соціальних опитувань дозволяє користувачам вбудовувати таргетовані опитування на свої сайти, збирати та аналізувати статистику, розподілену за різними параметрами.

Ключовими перевагами розробленої системи є її високонавантаженість, швидкодія та можливість налаштування опитувань згідно з унікальними потребами користувачів. Завдяки ретельному вибору мов програмування, бібліотек, сховищ даних та принципів розробки, система забезпечує надійну та продуктивну роботу, відповідаючи сучасним вимогам та дозволяючи ефективно взаємодіяти з аудиторією.

Використання мови Go для розробки серверної частини, Kafka для збору статистики та стовпчикової бази даних Vertica для її зберігання дозволяє досягти високої продуктивності та масштабованості системи. Абстрагування від конкретних сховищ даних забезпечує гнучкість та можливість заміни компонентів у майбутньому.

Розроблена система успішно реалізує поставлені вимоги, такі як генерація розмітки набору опитувань, протидія шахрайським трекерам, локалізація опитувань та додавання додаткової логіки. Вона забезпечує ефективний збір даних, що є важливим для проведення якісних соціологічних досліджень.

Загалом, впровадження даної системи може стати важливим кроком у напрямку цифровізації соціологічних досліджень в Україні та забезпеченні їх відповідності сучасним технологічним стандартам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Богдан О. Що варто знати про соціологію та соціальні дослідження? Посібник-довідник для громадських активістів та всіх зацікавлених / Олена Богдан; консультант-рецензент Володимир Паніотто – К: Дух і Літера, 2015. – 380 с. : іл.

2. Cargan L. Doing Social Research – Rowman & Littlefield Publishers, 2007.– 345 с.

3. Treiman D. J. Quantitative Data Analysis: Doing Social Research to Test Ideas. – John Wiley & Sons, 2009. – 480 с.

4. Embed Your Surveys in a website or blog, using an IFrame. [Електронний ресурс] – URL: https://www.surveylegend.com/user-guide/embed-surveys-forms-website-blog/ (дата звернення: 10.06.2024).

5. Papazoglou M. P. Service oriented architectures: approaches, technologies and research issues [Текст] / Mike P. Papazoglou // The VLDB Journal. – 2007. – №16 – 389-415 с.

6. What are the Benefits of Using Protocol Buffers? [Електронний ресурс] – URL: https://protobuf.dev/overview/#benefits (дата звернення: 10.06.2024).

7. Message Delivery Guarantees. [Електронний ресурс] – URL: https://docs.confluent.io/kafka/design/delivery-semantics.html (дата звернення: 10.06.2024).

8. The Secrets behind Vertica. [Електронний ресурс] – URL: https://www.vertica.com/secrets-behind-verticas-performance/ (дата звернення: 10.06.2024).

9. Encoding Types. [Електронний ресурс] – URL: https://docs.vertica.com/24.1.x/en/sql-reference/statements/create-statements/create-projection/encoding-types/ (дата звернення: 10.06.2024)

10. <img>: The Image Embed element. [Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img (дата звернення: 10.06.2024)

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

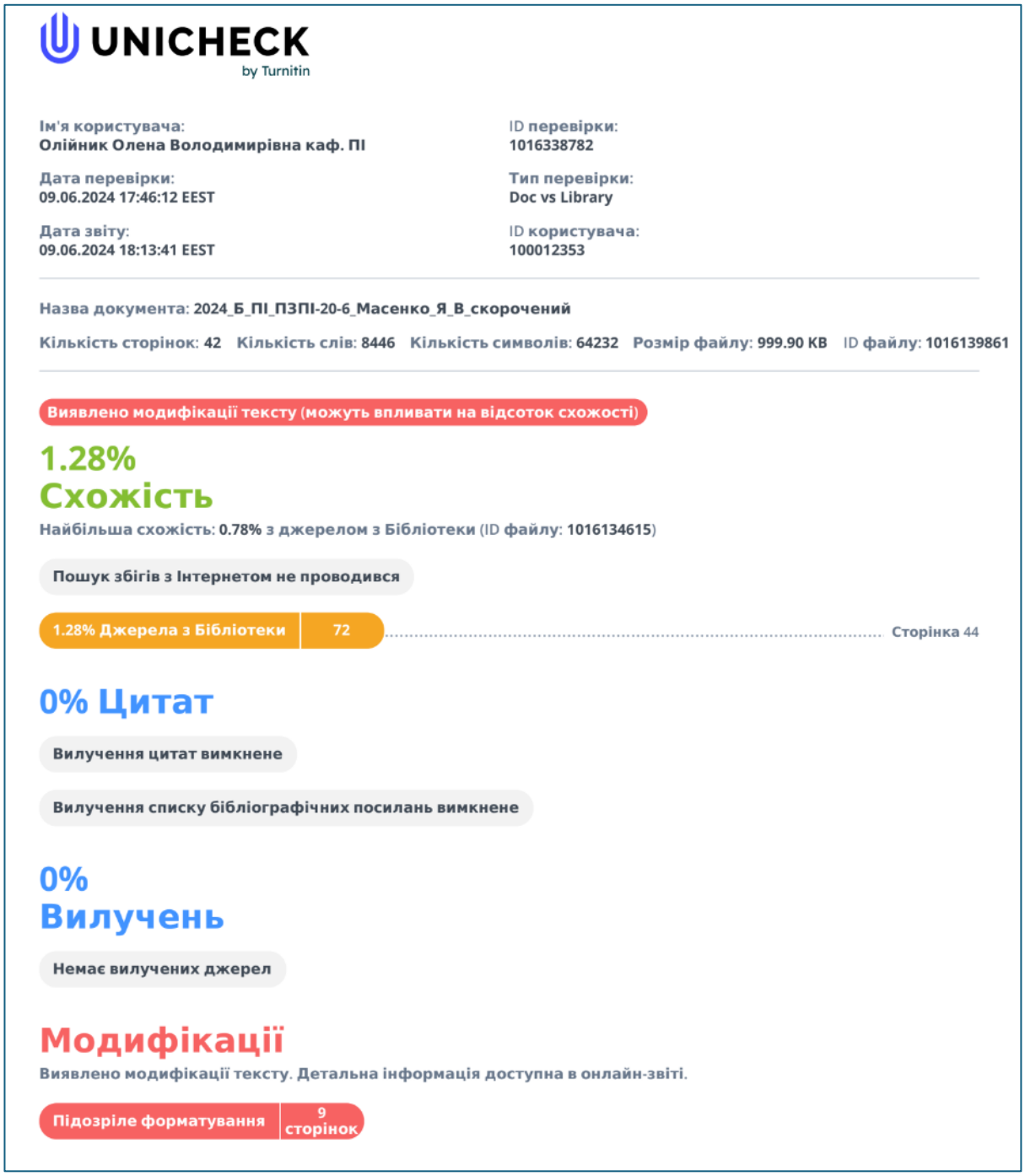

Рисунок А.1 – Результаті перевірки на унікальність тексту в базі ХНУРЕ

ДОДАТОК Б

Слайди презентації

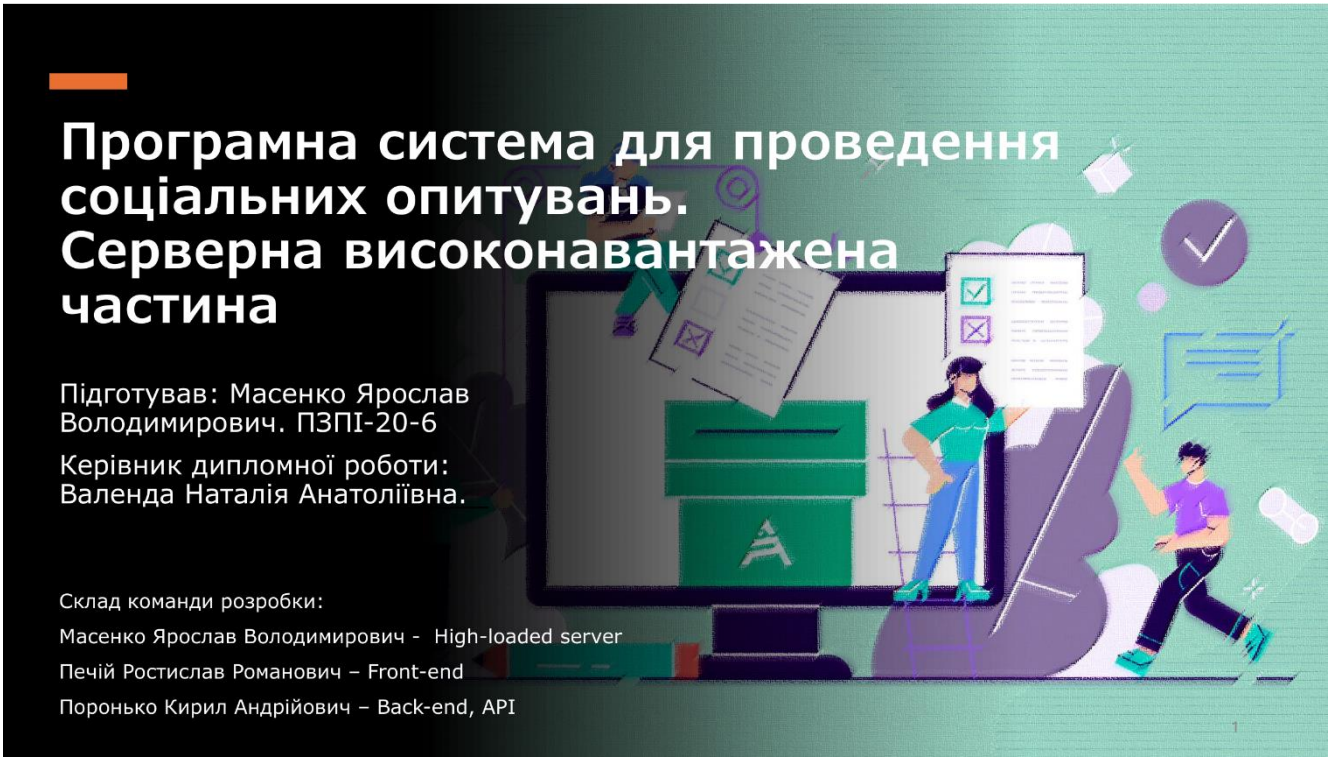

Рисунок Б.1 – Слайд 1

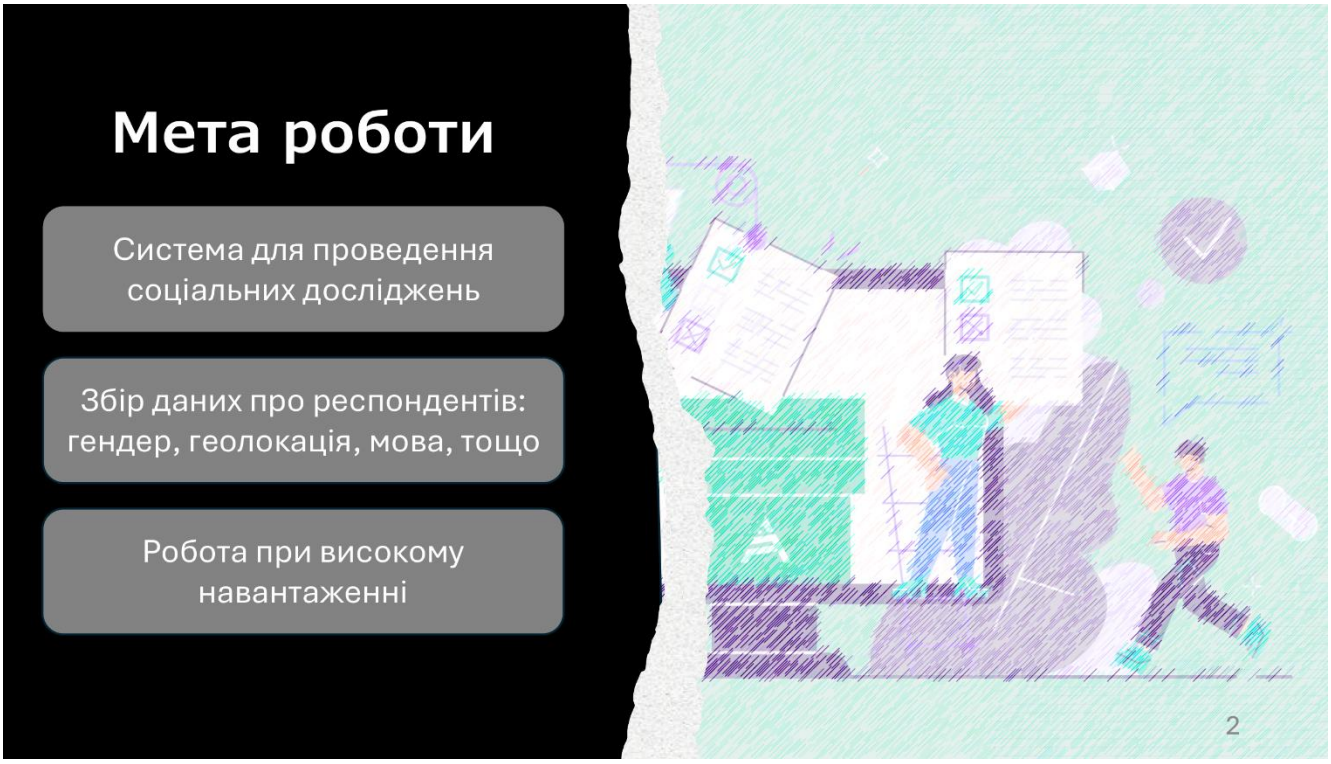

Рисунок Б.2 – Слайд 2

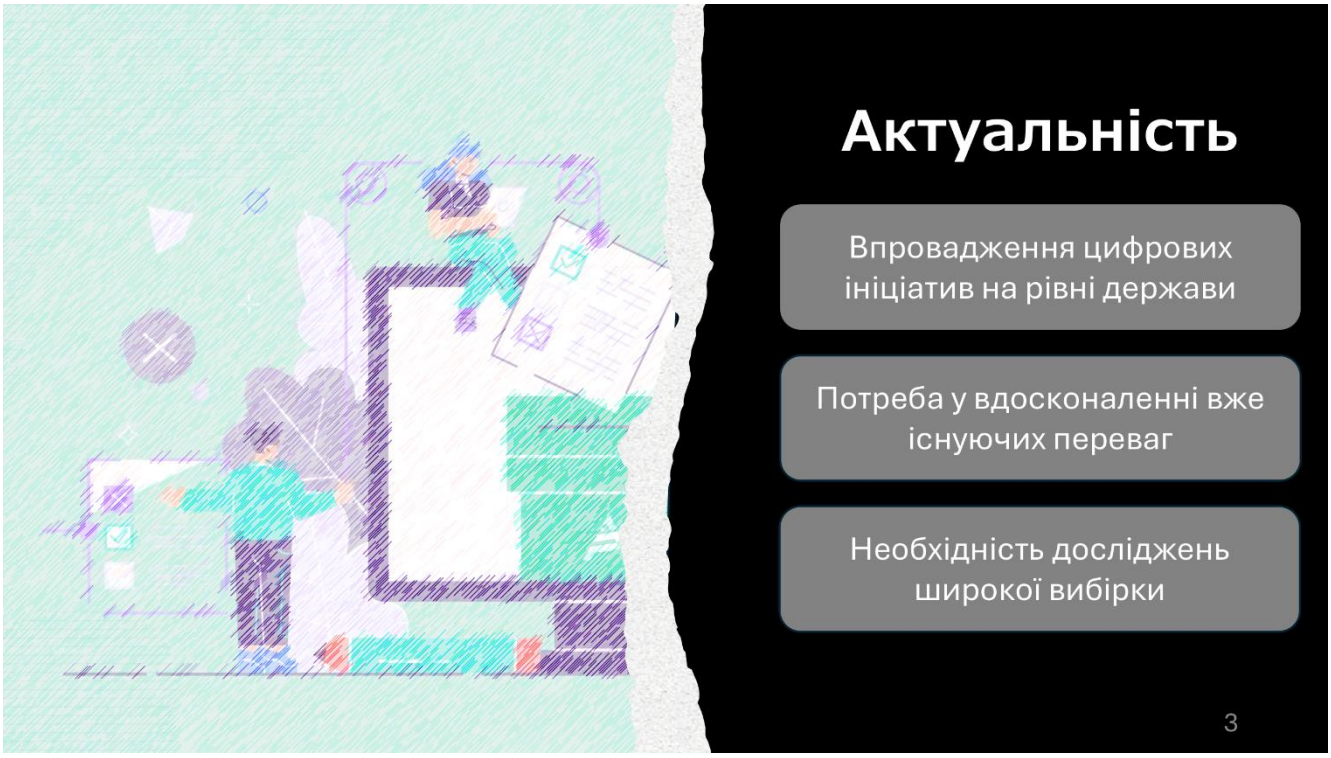

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

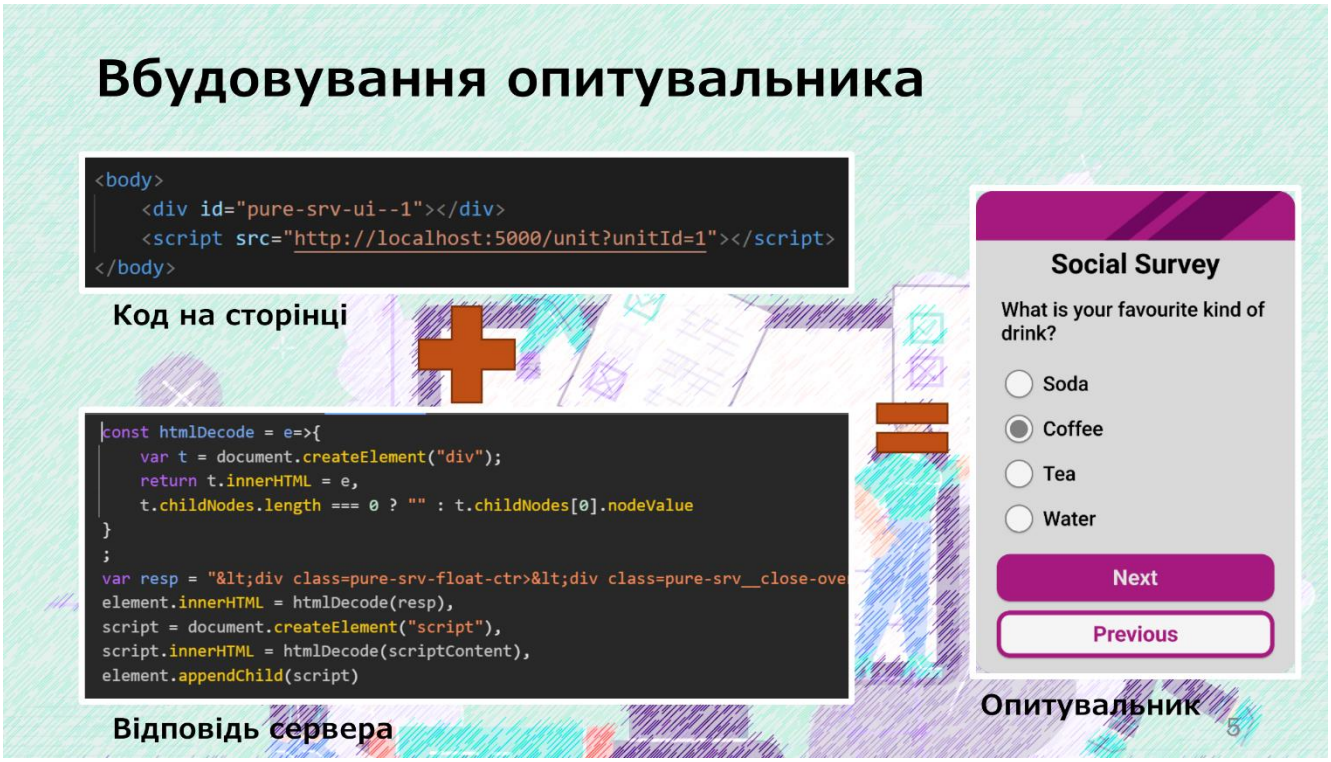

Рисунок Б.5 – Слайд 5

Рисунок Б.6 – Слайд 6

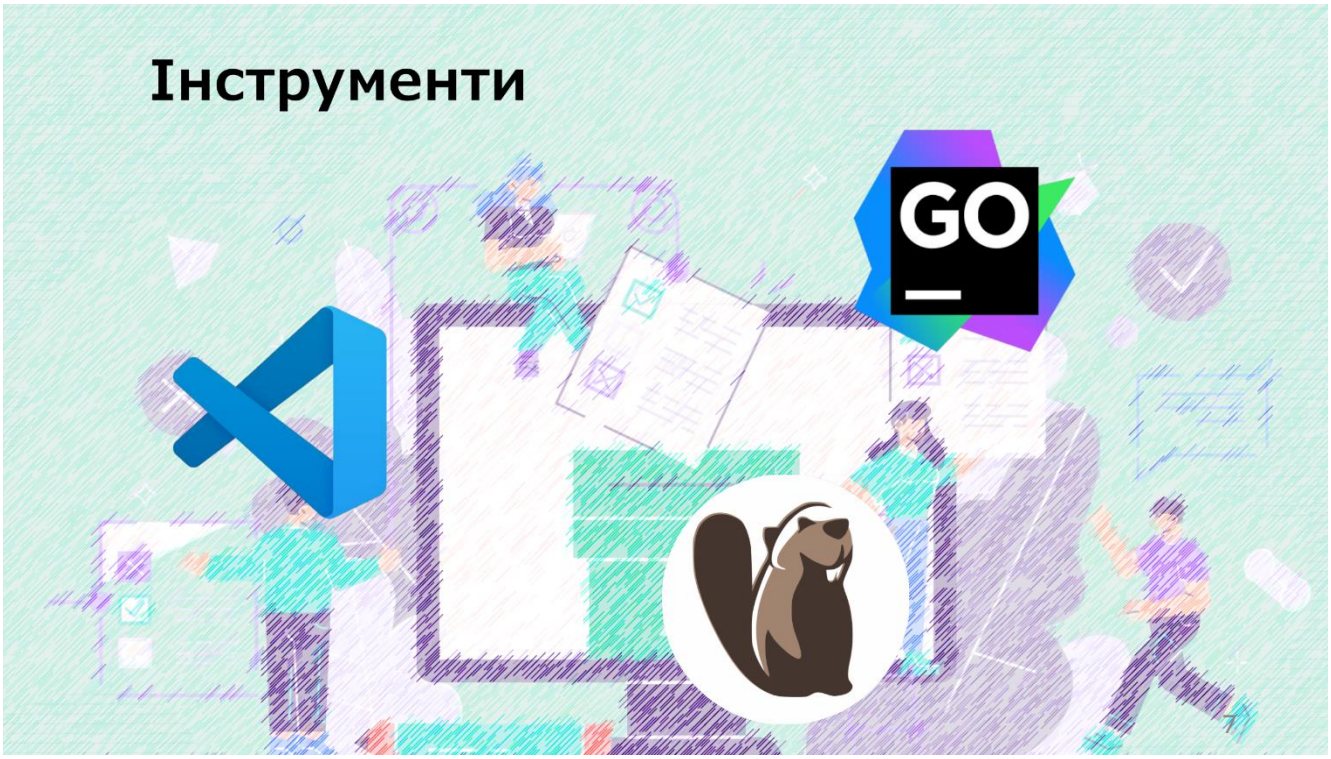

Рисунок Б.7 – Слайд 7

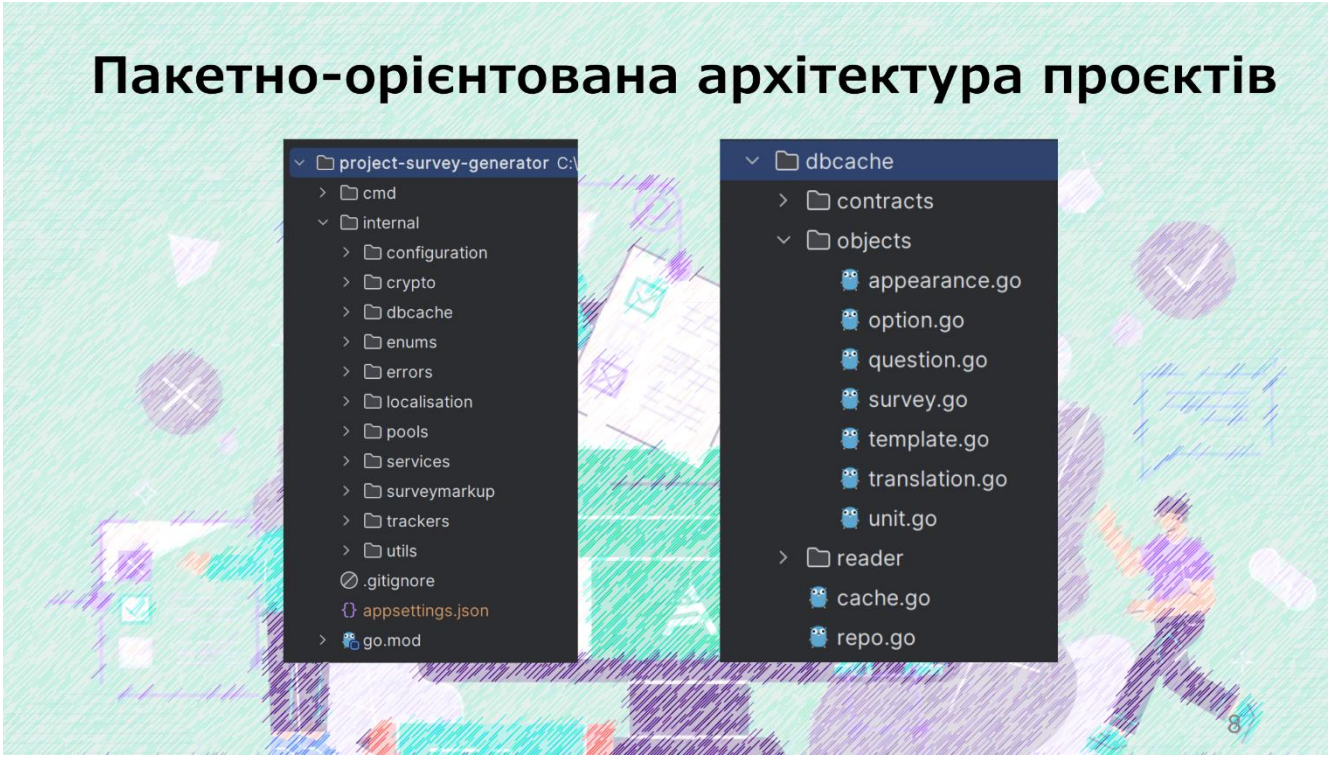

Рисунок Б.8 – Слайд 8

Рисунок Б.9 – Слайд 9

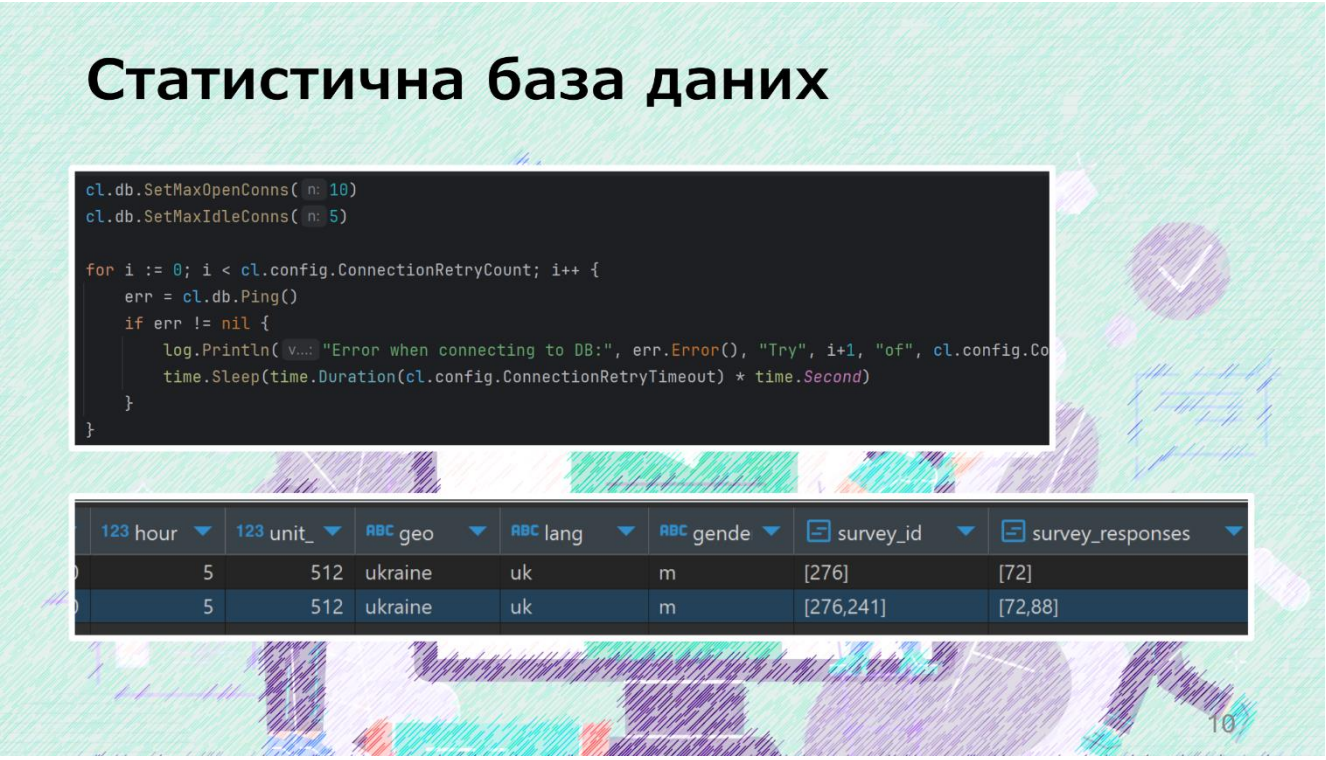

Рисунок Б.10 – Слайд 10

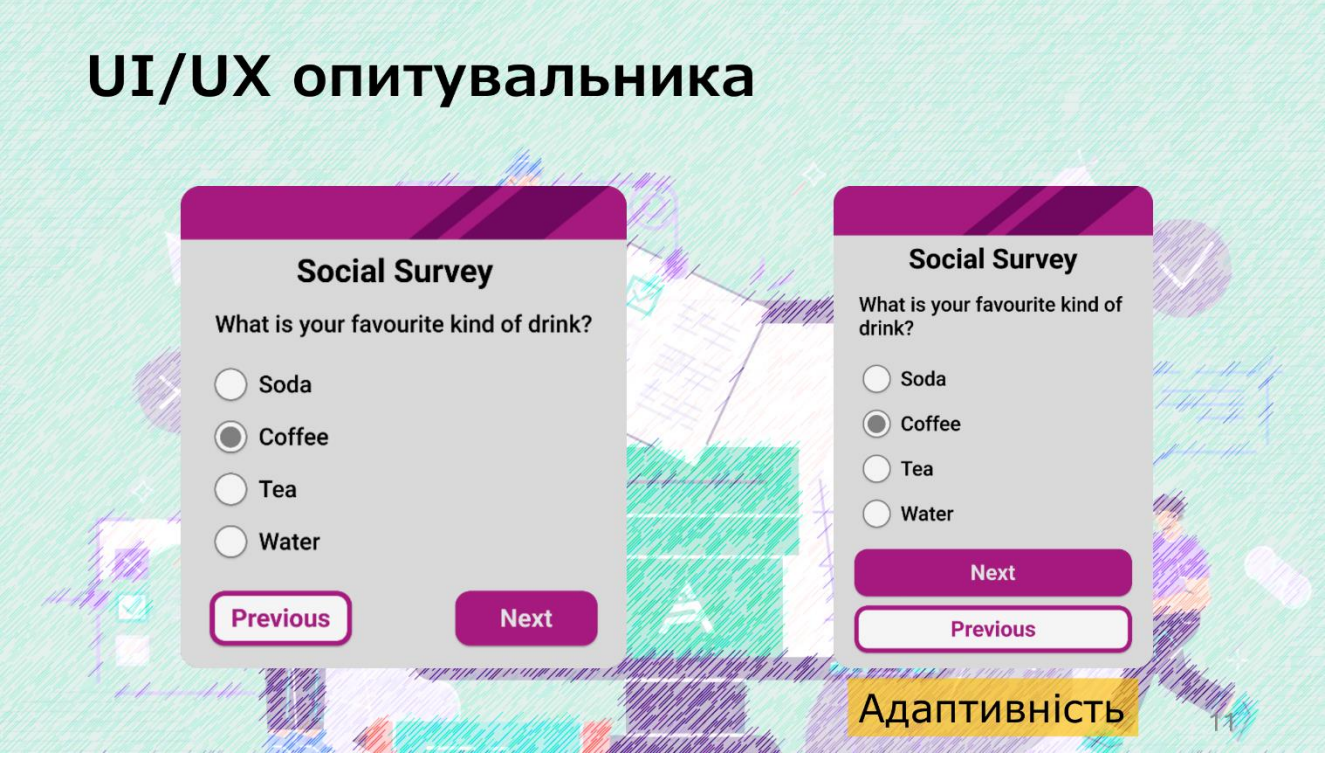

Рисунок Б.11 – Слайд 11

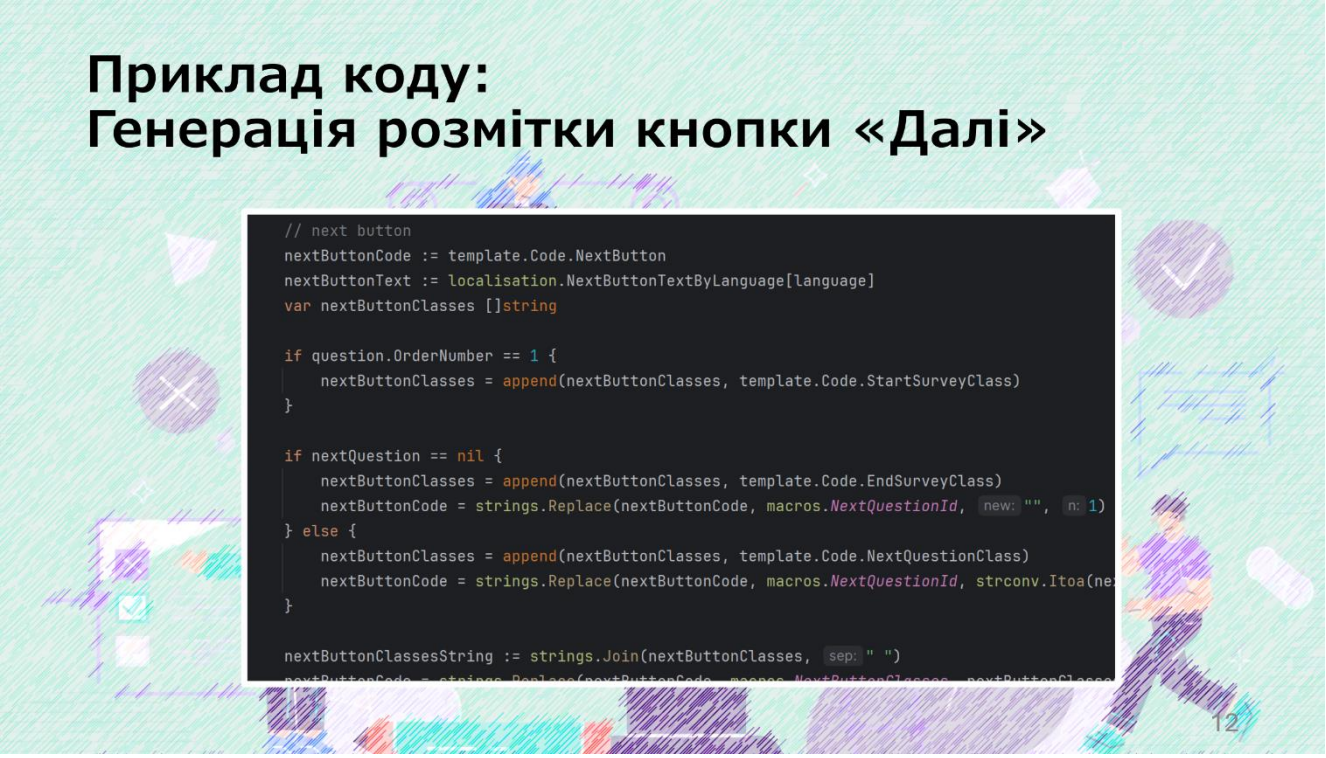

Рисунок Б.12 – Слайд 12

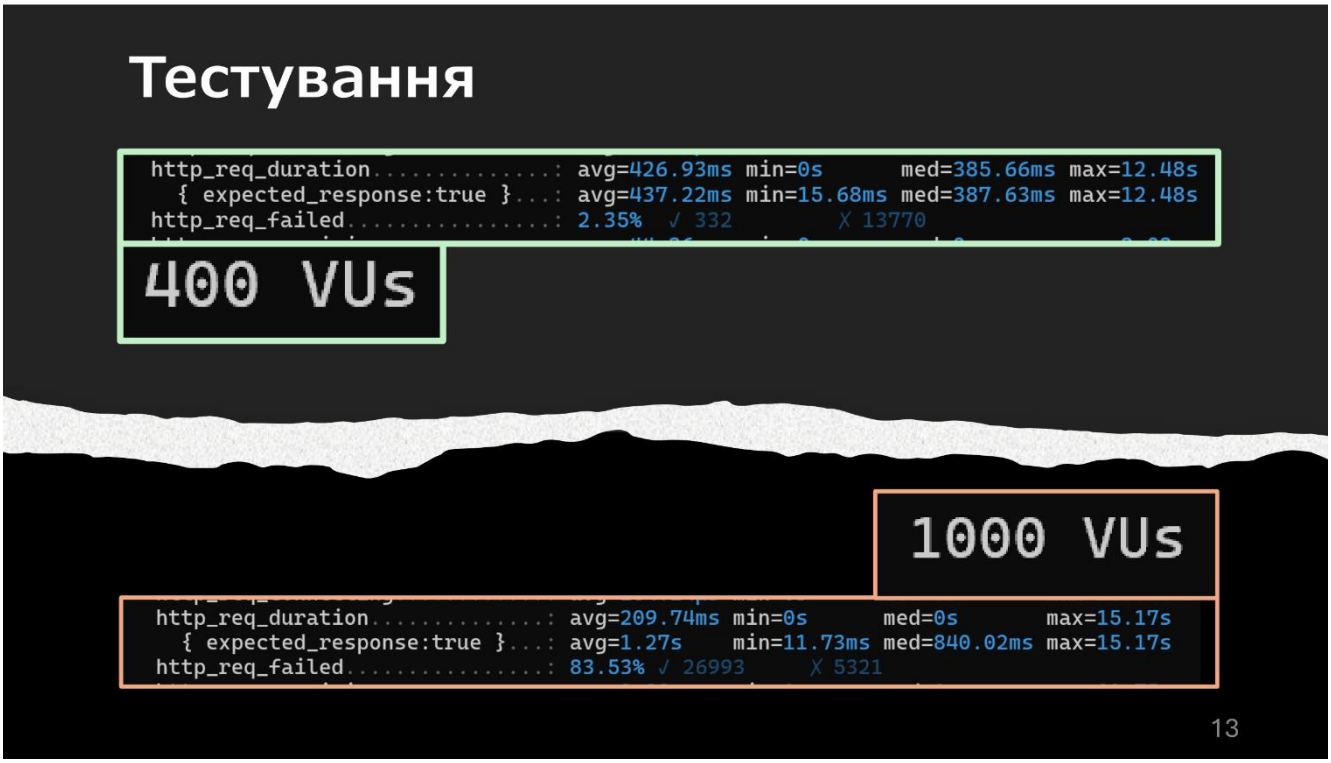

Рисунок Б.13 – Слайд 13

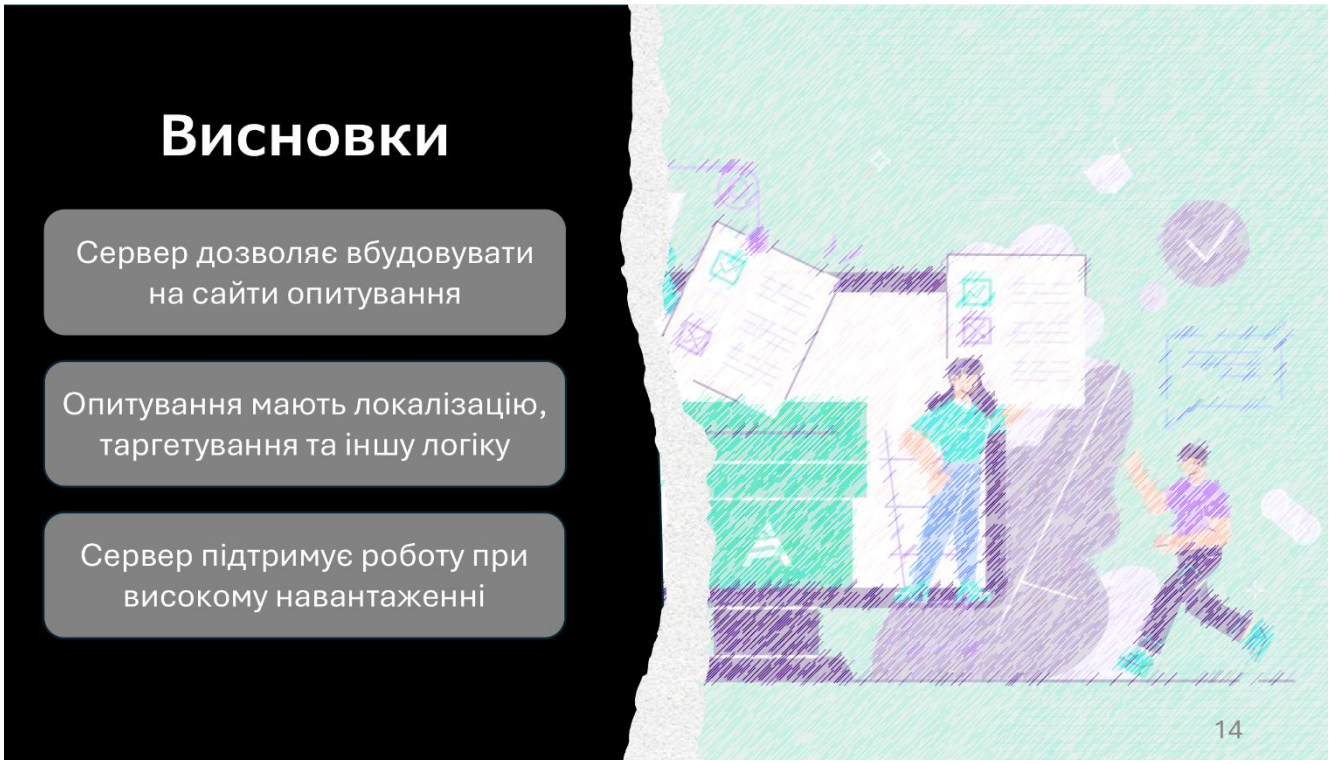

Рисунок Б.14 – Слайд 14

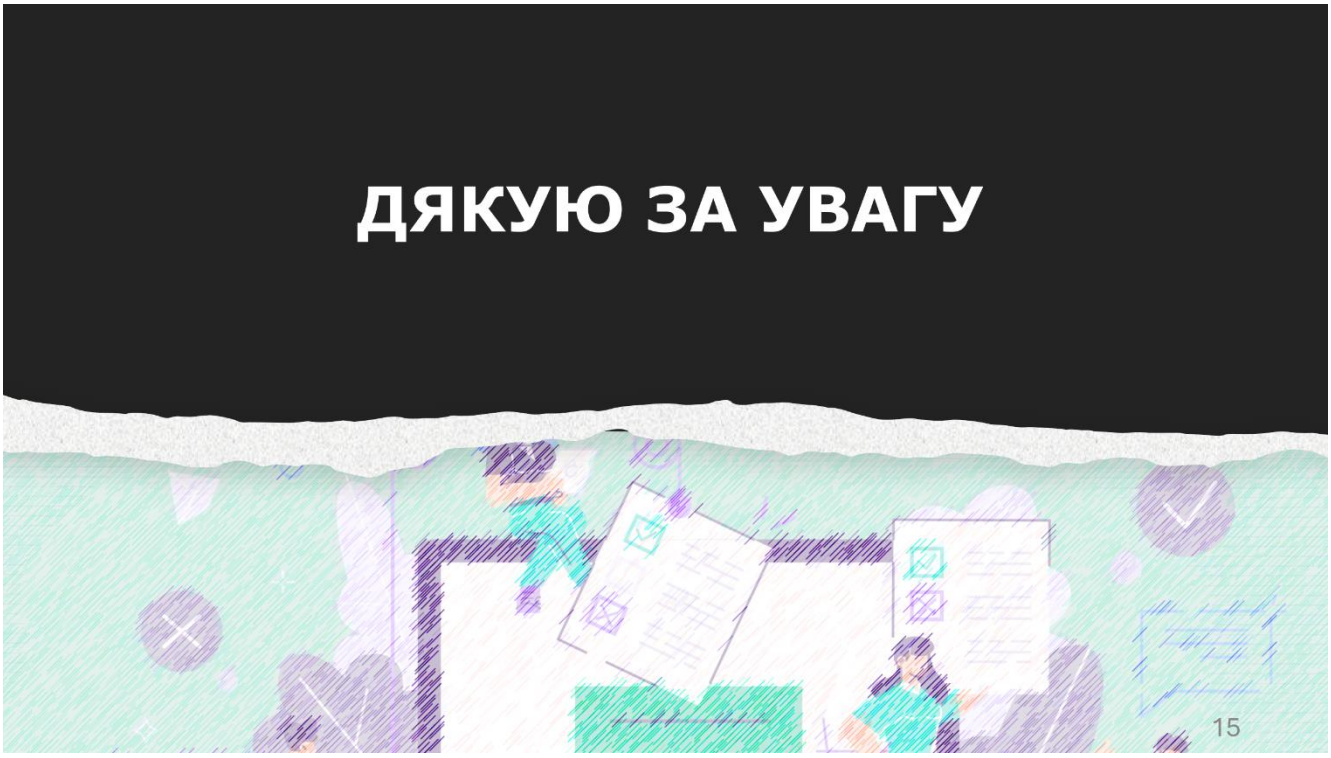

Рисунок Б.15 – Слайд 15

ДОДАТОК В

Специфікація ПЗ

СПЕЦИФІКАЦІЯ ПЗ

До системи для проведення соціальних опитувань

Виконали:

Поронько Кирил Андрійович

Печій Ростислав Романович

Масенко Ярослав Володимирович

Рисунок В.1 – Сторінка 1 специфікації ПЗ

1. Вступ

1.1 Огляд продукту

Ціль цього документа полягає у створенні програмної системи для проведення опитувань «Pure Survey», яка дозволяє користувачам створювати різноманітні опитування, обираючи певні налаштування, з можливістю вбудовувати опитування на інші вебсайти, та переглядати статистику відповідей на опитування.

1.2 Межі

«Pure Survey» це програмна система, яка дозволяє користувачам створювати свої власні соціальні опитування, використовуючи гнучкі інструменти редагування.

Зареєстровані користувачі можуть оглянути та використовувати функціонал зі створення опитувань, змінювати відповідну аудиторію, на яку націлене опитування та їх редагувати.

Користувачі можуть бачити інформацію про створені опитування, групи в яких вони містяться, а з активною підпискою – також переглядати статистику обрану користувачем з певними відібраними даними по певних опитуваннях. Також експортувати створені опитування у вигляді скрипту для їх використання на різних веб-сайтах.

Додатково користувач може додати певні кошти на свій особистий рахунок профілю для публікації опитувань в мережі «Інтернет», відповідно використовуючи кошти за певну кількість отриманих відгуків.

1.3 Означення та абревіатури

Table 1 - Означення

Термін	**Означення**
Таргетинг	Цільова аудиторія для якої буде доступне опитування. Залежить від країни користувача
Зареєстрований користувач	Користувач, який створив профіль у системі та може використовувати функції створення опитувань
Опитування	Елемент опитування створений юзером, залежний від налаштувань таргетингу. Опитування можна вбудовувати в інші вебсайти

Рисунок В.2 – Сторінка 2 специфікації ПЗ

Група опитувань	Декілька однаково налаштованих опитувань в контексті зовнішнього вигляду та налаштувань правил опитування, які відрізняються таргетингом та вмістом питань
Front-end/Клієнт	Графічний інтерфейс, який відображається користувачеві за допомогою HTML, CSS і JavaScript, щоб користувачі могли переглядати дані та взаємодіяти з ними
Серверна частина/АПІ	Інтерфейс взаємодії між клієнтом і сервером, який також взаємодіє з базою даних та відповідає за базову частину програмної системи.
Високонавантажена серверна частина	Інтерфейс взаємодії між клієнтом і сервером, який також взаємодіє з базою даних та відповідає за опрацювання взаємодії з опитуваннями.
Темплейт	Зовнішній вигляд групи опитувань, задається у форматі html-коду.

2. Загальний опис

2.1 Перспективи продукту

Програмна система складається з таких частин: веб-додаток користувача, веб-сервер для додатку, основний сервер бази даних для додатку, додаткова високонавантажена серверна частина і база даних для додаткової системи

Основна база даних міститься в СКБД MSSQL, яка містить в собі дані користувачів та опитувань. Користувач не може напряму комунікувати з сервером бази даних, але може використовувати основне API системи.

При потребі даних клієнт формує запит і посилає його на API веб-серверу. Даний веб-сервер інтерпретує запит і повертає помилку, якщо запит не може бути виконаний, інакше він формує та надсилає запит на сервер бази даних, або на інший додатковий веб-сервер відповідно до запиту користувача. База даних повертає помилку або дані, після чого веб-сервер перерозподіляє цю інфомацію необхідним чином і надсилає клієнтові. Тільки клієнт відповідає за надання даних у графічному представленні.

Веб-сервер може додавати або змінювати дані під час обміну інформації. API має бути прихована від користувачів, але доступна для інших частин системи і за її межами.

Рисунок В.3 – Сторінка 3 специфікації ПЗ

Додатковий веб-сервер використовує відповідно створену для нього базу даних і відповідає за внесення згенерованих подій під час проходження опитування і їх зберігання у додаткову базу даних, яка міститься СКБД Vertica. Також ця частина системи відповідальна за генерацію скриптів для вбудовування на інші веб-сайти.

2.2 **Функції продукту**

За допомогою клієнта зареєстровані користувачі зможуть створювати свої групи опитувань, налаштовуючи їхній зовнішній вигляд, правила проходження опитування, та створення одиниць опитування в групах, для яких будуть доступні налаштування контенту опитування та таргетингу. Опитування можна влаштовувати в інші вебсайти, де й будуть проходити більшість проходжень опитувань. Також буде реалізовано функціонал оплати підписки задля можливості активації функцій системи, таких як вбудовування опитувань на інші веб-сайти.

Адміністратори зможуть створювати та змінювати темплейти для груп опитувань та налаштовувати систему.

2.3 **Характеристики користувачів**

У системі існує 3 типи користувачів: зареєстровані користувачі з підпискою і без неї та адміністратори.

Зареєстрованим користувачам без активної підписки доступний такий функціонал системи як: створення опитувань, груп опитувань, налаштування опитувань та зовнішній вигляд.

Зареєстрованим користувачам з активною підпискою доступний весь функціонал системи, таких як: створення опитувань та отримання скриптів для вбудовування в інші вебсайти, перегляд їх статистики.

Незареєстрованим користувачам доступні лише функції логіна та реєстрації.

Адміністратори зможуть оперувати темплейтами які будуть доступні в системі.

2.4 **Загальні обмеження**

Веб-програма клієнта обмежена веб-браузером, встановленим на пристрої користувача. Оскільки існує кілька веб-браузерів і різні пристрої з

Рисунок В.4 – Сторінка 4 специфікації ПЗ

різною роздільною здатністю екрана, інтерфейс, швидше за все, не буде однаковим для кожного з них.

Підключення до Інтернету також є обмеженням для програми. Програма підключається до веб-сервера через Інтернет, і немає способу отримати інформацію без зв'язку між клієнтом і сервером.

Сервер обмежений характеристиками системи баз даних і сервера баз даних. Коли є багато запитів, база даних може бути змушена поставити їх у чергу, що збільшує час, необхідний для отримання даних.

2.5 **Припущення й залежності**

Основне припущення полягає в тому, що більшість сучасних браузерів працюють однаково та підтримують більш-менш новітні технології, такі як HTML5. Якщо браузер не підтримує нові технології, інтерпретація коду та виконання інструкцій можуть призвести до помилок.

3. **Конкретні вимоги**

3.1 **Вимоги до зовнішніх інтерфейсів**

3.1.1 **Інтерфейс користувача**

Зареєстровані користувачі можуть створювати опитування та оплачувати підписку, після чого отримувати скрипти для вбудовування в інші вебсайти, переглядати їх статистику.

3.1.2 **Апаратний інтерфейс**

Для відображення інтерфейсу користувача до пристрою має бути підключений дисплей.

3.1.3 **Програмний інтерфейс**

Щоб мати доступ до веб-системи, користувач повинен використовувати програмне забезпечення для веб-перегляду. Він повинен підтримувати Javascript, HTML5, CSS3.

3.1.4 **Комунікаційний протокол**

Щоб користувач мав доступ до веб-системи, необхідно встановити підключення до Інтернету.

Рисунок В.5 – Сторінка 5 специфікації ПЗ

3.2 **Функції продукту**

3.2.1 Реєстрація

3.2.1.1 Опис

Незареєстровані користувачі (відвідувачі) не мають дозволу на вхід до основного контенту системи, тому що це може призвести до хакерських атак. Щоб розпочати процес реєстрації, користувач повинен натиснути «створити обліковий запис». Це перемістить його до «реєстраційної форми».

3.2.1.2 Вхідні дані

Текст: Електронна пошта, пароль (8-50 символів, з перевіркою складності), ім’я(2-50 символів) прізвище (2-50 символів).

3.2.1.3 Обробка

При натисканні кнопки реєстрації клієнт надсилає запит із усіма вхідними даними на веб-сервер, який спілкується з API.

Веб-сервер перевіряє правильність логіна, пароля та електронної пошти. Якщо правильно – запитує сервер бази даних знайти користувачів із таким логіном або електронною поштою. Якщо знайдено – користувач залишається незареєстрованим і клієнт показує повідомлення з описаною причиною. В іншому випадку пароль буде хешовано, а всі дані, включаючи хеш перевірки, збережуться у базі даних.

3.2.1.4 Вихідні дані

Якщо форма була заповнена успішно, після відправки клієнт видає повідомлення про те, що акаунт було успішно створено та перейде на сторінку логіна.

Якщо форма була заповнена невдало, перед формою з'являється повідомлення з відповідним текстом.

3.2.1.5 Обробка помилок

Повідомлення з помилкою з'явиться перед формою і не буде надіслано, якщо:

- неправильна електронна адреса;
- підтвердження електронною поштою не можна надіслати на адресу електронної пошти;
- користувач із цією електронною адресою вже зареєстрований;
- немає зв'язку з базою даних;
- невалідний пароль.

3.2.1.6 Обмеження

Неавторизовані/незареєстровані користувачі.

Рисунок В.6 – Сторінка 6 специфікації ПЗ

3.2.2 Авторизація

3.2.2.1 Вступ

Неавторизовані користувачі (відвідувачі) можуть увійти, якщо вони мають зареєстрований профіль у системі. Це принесе їм привілеї. Щоб увійти, користувач повинен натиснути «увійти» в заголовку сторінки (в меню). Це перемістить його до «форми входу».

3.2.2.2 Вхідні дані

Текст: email (до 50 символів), пароль (8-50 символів).

3.2.2.3 Обробка

Веб-сервер хешує пароль і запитує базу даних для пошуку користувача з такою електронною поштою. Якщо знайдено та підтверджено користувача, у випадку якщо хешований пароль збігається – вхід виконано успішно. Це переносить дані користувача, що зберігаються в базі даних, до сеансу на стороні клієнта.

3.2.2.4 Вихідні дані

Якщо вхід виконано успішно, система вітає авторизованого користувача.

3.2.2.5 Обробка помилок

Якщо вхід не вдається, перед формою входу з’являється повідомлення з відповідним текстом.

3.2.2.6 Обмеження

Неавторизовані/незареєстровані користувачі.

3.2.2 Створення групи опитувань

3.2.2.1 Вступ

Користувачі зможуть створювати групу опитувань, яка є базою для кожного опитування, тобто для групи вказуються певні правила проходжень опитувань в цій групі, їх зовнішній вигляд, а саме: колір, розмір та шрифт. Після створення, користувачі можуть експортувати скрипт для вбудування опитування в інші вебсайти

3.2.2.2 Вхідні дані

Користувач заповнює форму створення групи опитувань, а саме такі налаштування як:

– дозвіл на повідомлення після завершення опитування;

Рисунок В.7 – Сторінка 7 специфікації ПЗ

- дозвіл на зникнення форми після проходження;
- кількість проходжень опитувань з одного пристрою за день;
- кількість проходження одного опитування одним пристроєм.

Також користувач обирає тип відображення, шаблон, задає певні параметри налаштування

3.2.2.3 Обробка

Після налаштування відображення опитування та правил проходження, юзер підтверджує форму. Після підтвердження форми клієнт відправляє запит на сервер зі створення групи опитувань та повертає статус створення та оновлює список груп опитувань з новоствореною групою.

3.2.2.4 Вихідні дані

Якщо якісь параметри було введено неправильно клієнт видає повідомлення про помилку. Якщо усе пройшло успішно – клієнт відображає нову групу опитувань у списку.

3.2.2.5 Обробка помилок

Якщо якісь параметри було введено неправильно клієнт видає помилку про те, що саме неправильно введено. Якщо було правильно введено усі дані, але була помилка на стороні сервера, то про це видається відповідна помилка. Якщо усе пройшло успішно – клієнт видає повідомлення про успішне створення та відображає нову групу опитувань у списку.

3.2.2.6 Обмеження

Доступно лише для зареєстровних користувачів/

3.2.3 Перегляд набору опитувань

3.2.3.1 Вступ

Для зареєстрованого користуача доступна функція перегляду усіх груп опитувань, для цього на відповідній сторінці буде відображено список груп опитувань, які було створено юзером. Також користувач зможе переглянути вміст кожної групи опитувань натиснувши на певну групу та переглянути її дані: усі опитування групи, тип відображення, правила проходження тощо.

3.2.3.2 Вхідні дані

Для перегляду усіх груп опитувань користувач повинен перейти до відповідної сторінки усіх груп, щоб перейти до інформації певної групи опитувань потрібно натиснути на потрібний елемент групи опитувань, що направить користувача на сторінку інформації про конкретну групу опитувань.

Рисунок В.8 – Сторінка 8 специфікації ПЗ

3.2.3.3 Обробка

Після переходу на сторінку усіх груп опитувань, клієнт відправляє запит на отримання усіх груп опитувань поточного юзера, які відображає після отримання відповіді. Після натискання на конкретну групу опитувань, клієнт відправляє запит до сервера про отримання інформації про задану групу опитувань та відображає дані.

3.2.3.4 Вихідні дані

Усі дані про групи опитувань користувач бачить на екрані гаджета, а саме таку інформацію як: назва групи, тип відображення, кількість опитувань в ній.

Усі дані про конкретну групу опитувань користувач бачить на екрані гаджета, а саме таку інформацію як:

- правила проходження (дозвіл на повідомлення після завершення опитування, дозвіл на зникнення форми після проходження, кількість проходжень опитувань з одного пристрою за день, кількість проходження одного опитування одним пристроєм)
- тип відображення (колір, шрифт, розмір, місце розташування, шаблон)
- список опитувань у групі (назва та базова інформацію про опитування)

3.2.3.5 Обробка помилок

Якщо в групі немає опитувань – повідомляємо користувача про це

3.2.3.6 Обмеження

Авторизовані користувачі.

3.2.4 Видалення набору опитувань

3.2.4.1 Вступ

Користувачу доступна функція видалення групи опитувань, у випадку, коли вона стає йому непотрібна.

3.2.4.2 Вхідні дані

Для видалення групи опитувань користувачу потрібно перейти до сторінки конкретної групи опитувань та натиснути на кнопку видалення, потім виконати підтвердження, натиснувши на відповідну кнопку.

3.2.4.3 Обробка

Рисунок В.9 – Сторінка 9 специфікації ПЗ

Після натискання користувачем на кнопку підтвердження видалення, клієнт виконує запит на видалення групи опитувань на сервер, після чого отримує результат запиту.

3.2.4.4 Вихідні дані

Кнопка видалення буде знаходитися на екрані користувача на сторінці конкретної групи опитувань. Повідомлення про успішне видалення чи помилку буде відображено модальним вікном на екрані користувача.

3.2.4.5 Обробка помилок

У випадку успішного видалення буде відображено відповідне повідомлення, у випадку помилки буде відображена причина помилки.

3.2.4.6 Обмеження

Авторизовані користувачі. Створена група опитувань для видалення

3.2.5 Редагування опитувань

3.2.5.1 Вступ

Користувачу доступна функція редагування опитування. Він може змінювати контент питання, їх переклад, варіанти відповіді та налаштування тагретингу.

3.2.5.2 Вхідні дані

Для редагування опитування, користувач має перейти на сторінку опитування та натиснути на кнопку редагування. Після цього змініти дані про таргетинг та питання і варіанти відповіді.

3.2.5.3 Обробка

Після відправки користувачем форми зі зміненими даними опитування, клієнт відправляє запит на сервер з оновлення даних про опитування, результат операції повертається на клієнт та залежно від результату відображається у користувача на екрані.

3.2.5.4 Вихідні дані

Якщо результат редагування успішний – система каже про успішну операцію користувачу

3.2.5.5 Обробка помилок

У випадку помилки при редагуванні, клієнт видасть помилку про це, відповідно до типу помилки, яку передає сервер.

3.2.5.6 Обмеження

Авторизований користувач, створене опитування для редагуваня.

Рисунок В.10 – Сторінка 10 специфікації ПЗ

3.2.6 Перегляд опитувань

3.2.6.1 Вступ

Користувач має можливість продивитися інформацію про опитування в групі, усі створені опитування та дані про конкретне опитування. У списку він має бачити таку інформацію, як: назва, кількість питань тощо. У деталізованих даних про опитування, користувач має змогу передивитися список питань, їх переклад, таргетинг та дані про групу опитувань, в якій він знаходиться з даними про групу, таку як тип відображення, налаштування тощо.

3.2.6.2 Вхідні дані

Для перегляду списку опитувань, користувачу потрібно перейти на сторінку певної групи опитувань, щоб подивитися спсиок опитувань у групі або на сторінку усіх опитувань, щоб переголянути усі створені. Щоб переглянути дані про конкретне опитування треба натиснути на певне опитування в списку, після чого користувач перейде на сторінку опитування.

3.2.6.3 Обробка

При переході користувача на сторінку усіх опитувань або опитувань у групі, клієнт відправляє відповідний запит на сервер, та, після отримання відповіді – відображає результат на сторінці. При переході до сторінки конкретного опитування, клієнт також посилає запит на АПІ, та відображає дані після отримання результату.

3.2.6.4 Вихідні дані

Усі дані про список опитувань чи дані про конкретне опитування будуть відображені на екрані користувача одразу по завантаженню відповідної сторінки.

3.2.6.5 Обробка помилок

У випадку помилки під час запиту на АПІ, на екрані користувача буде відображено відповідне повідомлення. Якщо у користувача немає опитувань, він не зможе перейти до сторінки конкретного опитування, а на сторінці спсика опитувань він побачить повідомлення про відсутність опитувань.

3.2.6.6 Обмеження

Авторизований користувач.

3.2.7 Створення опитувань

3.2.7.1 Вступ

Користувачу доступна функція додавання опитування. Він може додати питання, їх переклад, варіанти відповіді, переклади, та налаштування тагретингу.

Рисунок В.11 – Сторінка 11 специфікації ПЗ

3.2.7.2 Вхідні дані

Для додавання опитування, користувач має перейти на сторінку усіх опитувань групи, в яку він хоче додати опитування та натиснути на кнопку «Додати». Після цього обрати дані про таргетинг, додати питання, варіанти відповіді, їх переклад.

3.2.7.3 Обробка

Після відправки користувачем форми з даними опитування, клієнт відправляє запит на сервер з додавання нових даних про опитування, результат операції повертається на клієнт та залежно від результату відображається у користувача на екрані.

3.2.7.4 Вихідні дані

Якщо результат додавання успішний – система каже про успішну операцію користувачу та у списку одразу з’явиться нове опитування.

3.2.7.5 Обробка помилок

У випадку помилки при редагуванні, клієнт видасть помилку про це, відповідно до типу помилки, яку передає сервер.

3.2.7.6 Обмеження

Авторизований користувач, доступнийбезкоштовний слот для створення опиитування або куплена підписка.

3.2.8 Видалення опитувань

3.2.8.1 Вступ

Користувачу доступна функція видалення опитування, у випадку, коли воно стає йому непотрібне.

3.2.8.2 Вхідні дані

Для видалення опитування користувачу потрібно перейти до сторінки конкретного опитування та натиснути на кнопку видалення, потім виконати підтвердження, натиснувши на відповідну кнопку.

3.2.8.3 Обробка

Після натискання користувачем на кнопку підтвердження видалення, клієнт виконує запит на видалення опитування на сервер, після чого отримує результат запиту.

3.2.8.4 Вихідні дані

Рисунок В.12 – Сторінка 12 специфікації ПЗ

Кнопка видалення буде знаходитися на екрані користувача на сторінці конкретного опитування. Повідомлення про успішне видалення чи помилку буде відображено модальним вікном на екрані користувача.

3.2.8.5 Обробка помилок

У випадку успішного видалення буде відображено відповідне повідомлення, у випадку помилки буде відображена причина помилки.

3.2.8.6 Обмеження

Авторизовані користувачі. Створене опитування для видалення.

3.2.9 Додавання запитань до опитування

3.2.9.1 Вступ

Користувачу доступна функція додавання питання в опитування. Він може створити нове питання обравши варіанти відповіді та переклади для певних обраних мов.

3.2.9.2 Вхідні дані

Для додавання нового питання користувачу потрібно перейти до сторінки редагування певного опитування та натиснути на кнопку додавання нового питання. Після натискання на кнопку з'являється вікно створення нового питання, де користувач вписує текст питання та обирає відповідні варіанти відповідей та їх типи, а також написати текст питання на обраних мовах опитування.

3.2.9.3 Обробка

Після натискання кнопки збереження питання, клієнт виконує запит на додавання нового питання на сервер, після чого отримує результат запиту на додавання.

3.2.9.4 Вихідні дані

Кнопка додавання питання буде відображена на сторінці опитування. Після успішного додавання, користувач побачить повідомлення про це та також буде відображено щойно створене питання.

3.2.9.5 Обробка помилок

У випадку неуспішного додавання питання користувачеві буде відображена помилка з текстом причини результату запиту.

3.2.9.6 Обмеження

Авторизований користувач, створене опитування.

Рисунок В.13 – Сторінка 13 специфікації ПЗ

3.2.10 Видалення запитань з опитування

3.2.10.1 Вступ

Користувачу доступна функція видалення питання з опитування у разі його неактуальності.

3.2.10.2 Вхідні дані

Для видалення питання користувачу потрібно перейти до сторінки редагування певного опитування та натиснути на кнопку видалення питання. Після натискання на кнопку з'являється вікно підтвердження, де користувач підтверджує видалення.

3.2.10.3 Обробка

Після натискання кнопки підтвердження видалення, клієнт виконує запит на видалення питання у вигляді редагування опитування на сервер, після чого отримує результат запиту.

3.2.10.4 Вихідні дані

Кнопка видалення буде відображена на сторінці редагування опитування біля кожного питання. Після видалення питання, користувач побачить повідомлення та питання зникне з опитування.

3.2.10.5 Обробка помилок

У разі помилки при запиті або неправильно введених даних користувач побачить відповідне повідомлення.

3.2.10.6 Обмеження

Авторизований користувач, створене опитування та питання в ньому.

3.2.11 Редагування запитань в опитуванні

3.2.11.1 Вступ

Користувачу доступна функція редагування питання в опитуваннях. Він може редагувати створене питання змінюючи варіанти відповіді та переклади для певних обраних мов.

3.2.11.2 Вхідні дані

Для редагування створеного питання користувачу потрібно перейти до сторінки редагування певного опитування та натиснути на редагування відповідного питання. Після натискання на кнопку з'являється вікно редагування питання, де користувач змінює текст питання та змінює, додає або видаляє відповідні варіанти відповідей і їх типи, а також змінити чи додати текст питання на обраних мовах опитування.

3.2.11.3 Обробка

Рисунок В.14 – Сторінка 14 специфікації ПЗ

Після натискання кнопки збереження питання, клієнт виконує запит на оновлення даниї певного питання на сервер, після чого отримує результат запиту на зміну даних.

3.2.11.4 Вихідні дані

Кнопка редагування питання буде відображена на сторінці опитування біля певного питання. Після успішного редагування, користувач побачить повідомлення про це, і на сторінці опитування буде відображене щойно відредаговане питання .

3.2.11.5 Обробка помилок

У випадку неуспішного додавання питання користувачеві буде відображена помилка з текстом причини результату запиту

3.2.11.6 Обмеження

Авторизований користувач, створене опитування та питання в ньому.

3.2.12 Оплата підписки за користування послугами

3.2.12.1 Вступ

Щоб користуватися функціями активації опитувань на інших сайтах, користувачу потрібно купити підписку.

3.2.12.2 Вхідні дані

Щоб активувати підписку, користувачу потрібно натиснути на кнопку активації, відбувається редірект на вікно оплати PayPal, в якій користувач вводить свої дані для оплати та натискає на кнопку «сплатити».

3.2.12.3 Обробка

Після натискання кнопки оплати підписки користувача відбувається запит на створення підписки PayPal, після чого відбувається перехід на іншу сторінку з оплатою на PayPal систему і посилається запит на оплату підписки з відповідним станом оплати.

3.2.12.4 Вихідні дані

Після успішно проведеної оплати користувач може побачити стан активації підписки у своєму профілі з відповідною датою кінця статусу підписки.

3.2.12.5 Обробка помилок

У випадку неуспішної оплати, сторонній сервіс повідомляє користувача про результат оплати, при цьому на сайті буде відображена помилка про статус неуспішної оплати підписки.

Рисунок В.15 – Сторінка 15 специфікації ПЗ

3.2.12.6 Обмеження

Авторизований користувач, діюча картка PayPal.

3.2.13 Перегляд статистики по опитуванню

3.2.13.1 Вступ

Користувачу доступна функція перегляду статистики по обраному опитуванню. Він може переглядати та фільтрувати дані, отримані з відгуків клієнтів, використовуючи візуальні діаграми та засоби фільтрації.

3.2.13.2 Вхідні дані

Для перегляду статистики опитування користувачу потрібно перейти до сторінки статистики і натиснути на відповідну кнопку для перегляду статистики, після чого користувач буде переправлений на сторінку з відображеними діаграмами, які користувач може обрати, а також змінити поля та фільтрацію за допомогою відповідних засобів на сторінці.

3.2.13.3 Обробка

Після натискання кнопки на перегляд статистики, клієнт виконує запит на отримання даних по обраному питанню на високонавантажений сервер, після чого формуються відповідні дані для їх перегляду на сторінці.

3.2.13.4 Вихідні дані

Кнопка перегляду статистики опитування відображена на сторінці опитування біля основної інформації. Після успішного отримання даних, користувач побачить сторінку з відображеними даними у вигляді діаграм, і на цій ж сторінці користувач зможе відфільтрувати дані за допомогою наявних інструментів.

3.2.13.5 Обробка помилок

У випадку неуспішного отримання даних опитування користувачеві буде відображена помилка з текстом причини результату запиту

3.2.13.6 Обмеження

Авторизований користувач, опитування та відгуки на це опитування.

3.2.14 Перегляд таргетингів

3.2.14.1 Вступ

Користувач може переглядати усі створені таргетинги для опитувань двома способами: при створенні опитування, коли користувач обирає таргетинг, та окремо зайшовши на сторінку усіх таргетингів.

3.2.14.2 Вхідні дані

Рисунок В.16 – Сторінка 16 специфікації ПЗ

Перегляд таргетінгів можливий після переходу на сторінку усіх таргетингів та під час створення опитування у вигляді форми.

3.2.14.3 Обробка

В обох випадках процес обробки однаковий, а саме: клієнт відправляє запит про усі створені таргетинги на сервер та, після отримання відповіді, відображає список таргетингів.

3.2.14.4 Вихідні дані

Список таргетінгів буде відображено на екрані користувача під час створення опитування та після переходу на сторінку усіх тагретингів.

3.2.14.5 Обробка помилок

У випадку помилок клієнт повідомить користувача про те, що сталася помилка з текстом відповідним до типу помилки. Якщо не було створено жодних таргетингів, то про це буде повідомлено користувача.

3.2.14.6 Обмеження

Авторизований користувач

3.2.15 Створення таргетингів

3.2.15.1 Вступ

Для налаштування аудиторії, на яку буде націлено опитування, користувач зможе створювати таргетинги, обравши цільові країни.

3.2.15.2 Вхідні дані

Для створення таргетингу користувачу потрібно натиснути на відповідну кнопку та заповнити форму, де потрібно обрати цільові країни та придумати назву таргетингу.

3.2.15.3 Обробка

Під час підтвердження та відправки форми створення таргетингу, клієнт посилає запит на сервер, де створюється таргетинг, та відображає на клієнті результат залежно від відповіді.

3.2.15.4 Вихідні дані

У випадку успішного створення, про це буде повідомлено користувача.

3.2.15.5 Обробка помилок

Якщо під час створення було отримано помилку – вона відображається на екрані користувача.

3.2.15.6 Обмеження

Рисунок В.17 – Сторінка 17 специфікації ПЗ

Авторизований користувач.

3.3 **Класи/Об’єкти**

3.3.1 User

3.4.1.1 Атрибути

- Email – text
- Hashed password – text
- First name – text
- Last name – text
- Role – text

3.3.2 SurveyUnit

3.3.2.1 Атрибути

- Name – text
- One survey take per device – int
- Maximum surveys per device – int
- Message after no surveys – bool
- Hide after no surveys – bool

3.3.3 UnitAppearance

3.3.3.1 Атрибути

- Type – text
- Params – text
- State – text
- Name – text
- Template code – text
- Default params – text

3.3.4 Survey

3.3.4.1 Атрибути

- Name – text
- DateBy – text

Рисунок В.18 – Сторінка 18 специфікації ПЗ

- Params – text
- Countries – text
- QuestionType – text
- OrderNumber – int
- Translate – text

3.4 **Нефункціональні вимоги**

3.4.1 **Продуктивність**

Будь-яка сторінка повинна бути завантажена менш ніж за 5 секунд;

3.4.2 **Надійність**

Система не повинна виходити з ладу, коли користувач вводить неправильні параметри.

3.4.3 **Доступність**

Система повинна бути доступною для використання з будь-якого місця.

3.4.4 **Безпека**

- Конфіденційні дані повинні бути захищені;
- Паролі мають бути хешованими.

3.4.5 **Придатність до обслуговування**

Повинна бути панель адміністратора з реалізованими деякими функціональними вимогами.

3.4.6 **Портативність**

Сервіс має бути доступним для використання з мобільних браузерів.

3.5 **Зворотні вимоги**

Веб-сервіс не надає жодних додаткових конфіденційних даних користувача, окрім тих, які вводить користувач.

3.6 **Обмеження дизайну**

- всі користувацькі інтерфейси повинні бути перекладені українською мовою;
- колір # 2196F3 використовується для панелей (наприклад, головного меню);

Рисунок В.19 – Сторінка 19 специфікації ПЗ

- колір #FFFFFF для тла або тексту, якщо текст написаний на темних кольорах;
- колір #000000 для тексту, якщо він написаний на світлих кольорах;
- Pure Survey використовує шрифти: Impact для заголовків, Century Gothic для підзаголовків;
- дизайн має бути мінімалістичним.

3.7 **Вимоги до логічної бази даних**

Будемо використовувати MS SQL для основної функціональності системи, оскільки важлива безпека та цілісність даних та СКБД Vertica для збереження даних стстистики опитувань.

Рисунок В.20 – Сторінка 20 специфікації ПЗ

ДОДАТОК Г

Приклад програмного коду (dbcache.Cache)

```
package dbcache

import (
	"database/sql"
	"encoding/json"
	"project-survey-generator/internal/dbcache/objects"
	"project-survey-generator/internal/enums"
)

type Cache struct {
	Units       map[int]*objects.Unit
	Surveys     map[int]*objects.Survey
	Templates   map[int]*objects.Template
	Appearances map[int]*objects.Appearance
	Questions   map[int]*objects.Question
	Options     map[int]*objects.Option

	SurveysByUnit     map[int][]*objects.Survey
	QuestionsBySurvey map[int][]*objects.Question
	OptionsByQuestion map[int][]*objects.Option

	TranslationsByQuestionLine
map[int]map[string]*objects.Translation
	TranslationsByOption     map[int]map[string]*objects.Translation
}

func (c *Cache) fillUnits(rows *sql.Rows) error {
	for rows.Next() {
		var id, appearanceId int
		var hideAfterNoSurveys bool
		var message string

		err := rows.Scan(&id, &appearanceId, &hideAfterNoSurveys,
&message)
		if err != nil {
			return err
		}

		unit       :=       objects.NewUnit(id,       appearanceId,
hideAfterNoSurveys, message)
		c.Units[id] = unit
	}

	return nil
}

func (c *Cache) fillSurveys(rows *sql.Rows) error {
	for rows.Next() {
		var id int

		err := rows.Scan(&id)
		if err != nil {
			return err
```

```
            }

            survey := objects.NewSurvey(id)
            c.Surveys[id] = survey
      }

      return nil
}

func (c *Cache) fillSurveyInUnits(rows *sql.Rows) error {
      for rows.Next() {
            var surveyId, surveyUnitId int

            err := rows.Scan(&surveyId, &surveyUnitId)
            if err != nil {
                  return err
            }

            survey := c.Surveys[surveyId]
            if survey != nil {
                  c.SurveysByUnit[surveyUnitId]                                =
append(c.SurveysByUnit[surveyUnitId], survey)
            }
      }

      return nil
}

func (c *Cache) fillTemplates(rows *sql.Rows) error {
      for rows.Next() {
            var id int
            var code, defaultParamsString string

            err := rows.Scan(&id, &code, &defaultParamsString)
            if err != nil {
                  return err
            }

            var templateCode objects.TemplateCode
            err = json.Unmarshal([]byte(code), &templateCode)
            if err != nil {
                  return err
            }

            var defaultParams map[string]string
            err      =      json.Unmarshal([]byte(defaultParamsString),
&defaultParams)
            if err != nil {
                  return err
            }

            template    :=    objects.NewTemplate(id,    &templateCode,
defaultParams)
            c.Templates[id] = template
      }

      return nil
}
```

```
func (c *Cache) fillAppearances(rows *sql.Rows) error {
	for rows.Next() {
		var id, aType, templateId int
		var paramsString string

		err := rows.Scan(&id, &aType, &templateId, &paramsString)
		if err != nil {
			return err
		}

		var params map[string]string
		err = json.Unmarshal([]byte(paramsString), &params)
		if err != nil {
			return err
		}

		appearance := objects.NewAppearance(id,
enums.EnumAppearanceType(aType), templateId, params)
		c.Appearances[id] = appearance
	}

	return nil
}

func (c *Cache) fillQuestions(rows *sql.Rows) error {
	for rows.Next() {
		var id, qType, surveyId, orderNumber, questionLineId int

		err := rows.Scan(&id, &qType, &surveyId, &orderNumber,
&questionLineId)
		if err != nil {
			return err
		}

		question := objects.NewQuestion(id,
enums.QuestionType(qType), surveyId, orderNumber, questionLineId)
		c.Questions[id] = question
		c.QuestionsBySurvey[surveyId] =
append(c.QuestionsBySurvey[surveyId], question)
	}

	return nil
}

func (c *Cache) fillOptions(rows *sql.Rows) error {
	for rows.Next() {
		var id, questionId, orderNumber int

		err := rows.Scan(&id, &questionId, &orderNumber)
		if err != nil {
			return err
		}

		option := objects.NewOption(id, questionId, orderNumber)
		c.Options[id] = option
		c.OptionsByQuestion[questionId] =
append(c.OptionsByQuestion[questionId], option)
```

```
	}

	return nil
}

func (c *Cache) fillQuestionTranslations(rows *sql.Rows) error {
	for rows.Next() {
		var id, questionLineId int
		var lang, translationLine string

		err	:=	rows.Scan(&id,	&lang,	&translationLine,
&questionLineId)
		if err != nil {
			return err
		}

		translation := objects.NewTranslation(id, translationLine,
lang, questionLineId)

		if c.TranslationsByQuestionLine[questionLineId] == nil {
			c.TranslationsByQuestionLine[questionLineId]		=
map[string]*objects.Translation{}
		}

		c.TranslationsByQuestionLine[questionLineId][lang]		=
translation
	}

	return nil
}

func (c *Cache) fillOptionTranslations(rows *sql.Rows) error {
	for rows.Next() {
		var id, optionId int
		var lang, translationLine string

		err := rows.Scan(&id, &lang, &translationLine, &optionId)
		if err != nil {
			return err
		}

		translation := objects.NewTranslation(id, translationLine,
lang, optionId)

		if c.TranslationsByOption[optionId] == nil {
			c.TranslationsByOption[optionId]				=
map[string]*objects.Translation{}
		}

		c.TranslationsByOption[optionId][lang] = translation
	}

	return nil
}
```