Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів»

Виконав: студент 2-го курсу, групи 1ПІ-20м

спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Бугайов В.Ю.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ
Коваленко О.О.
(прізвище та ініціали)
«____» _____2021 р.

Опонент: к.т.н., доц. каф. КН
Арсенюк І.Р.
(прізвище та ініціали)
«____» _____2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
(прізвище та ініціали)
«____» _____2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 13 » вересня 2021 р.

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

Бугайову Володимиру Юрійовичу

1. Тема роботи – удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів.

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «24» вересня 2021 р. № 277.

2. Строк подання студентом роботи
1 грудня 2021 р.

3. Вихідні дані до роботи:
Операційна система – Windows, Mac OS.
Мова програмування – C#, TypeScript, SQL.
Технології – ASP.NET Core, SQL Server.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз завдання і методів удосконалення процесів A/B тестування для ефективної адаптації веб-сайту; розробка структури та алгоритму системи для A/B тестування; розробка модулів платформи для A/B тестування; тестування додатку; економічна частина; висновки; додатки.

5. Перелік графічного матеріалу: титульний слайд; мета, об'єкт та предмет дослідження; наукова новизна та практична цінність; методи проведення користувацьких інтерфейсів; огляд платформ для A/B експериментів; загальний алгоритм роботи; архітектура додатку; вікна програми; результати тестування; основні результати роботи, заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Коваленко О. О., к.т.н., доцент кафедри ПЗ		
5	Ратушняк О. Г., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання 14 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз, вибір та обґрунтування актуальності розробки додатку	15.09.2021 – 27.09.2021	Вик.
2	Розробка структури та алгоритму роботи додатку	28.09.2021 – 15.10.2021	Вик.
3	Розробка та тестування модулів додатку	16.10.2021 – 01.11.2021	Вик.
4	Економічна частина	02.11.2021 – 11.11.2021	Вик.
5	Оформлення матеріалів до захисту магістерської кваліфікаційної роботи	12.11.2021 – 30.11.2021	Вик.

Студент _____ **Бугайов В. Ю.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О. О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Бугайов В. Ю. Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 128 с.

На укр. мові. Бібліогр.: 32 назви; рис.: 45; табл. 14.

У магістерській кваліфікаційній роботі «Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів» розроблено програмний додаток, який дозволяє інтегрувати підтримку проведення A/B тестування на обраному веб-сайті. У ході роботи було розроблено метод поєднання Бета тестування з A/B тестуванням. Отримав подальшого розвитку метод синтетичної контрольної групи для проведення A/B тестувань.

В ході виконання роботи було проаналізовано сучасні підходи проведення тестувань для ефективної адаптації до вимог користувачів та системи-аналоги. Також, було розроблено архітектуру додатку, графічний інтерфейс, проаналізовано засоби розробки. В результаті, розроблено програмні модулі з використанням середовища ASP.NET Core та протестовано роботу додатку.

Розроблений продукт містить зрозумілий інтерфейс, не вимагає будь-якого встановлення від кінцевого користувача, що дає змогу мати доступ до всіх даних у будь-який момент часу та в будь-якому місці.

Ключові слова: A/B тестування, адаптація веб-сайту, проведення експериментів, платформа для тестування.

ABSTRACT

Buhaiov V.Y. Improving A/B testing processes to effectively adapt the website to user requirments. Master's thesis in specialty 121 - Software engineering. Vinnitsa: VNTU, 2021. – 128 p.

In Ukrainian language. Bibliographer: 32 titles; fig.: 45; tabl. 14.

In the master’s thesis, the software application that allows you to integrate support for A/B testing on the selected website was developed. In the course of the work, a method was developed to combine Beta testing with A/B testing. Also, received further development method of the synthetic control group for A/B testing usage.

In the course of the work, modern approaches for testing effective adaptation to user requirements and similar systems were analyzed. Also, the application architecture and graphical interface were developed and development tools were analyzed. As a result, software modules were developed using the ASP.NET Core framework and the developed application was tested.

The developed product contains a clear interface, does not require any installation from the end-user, that allows you to access all the data at any time and in any place.

Keywords: A/B testing, website adaptation, launching experiments, platform for testing.

ЗМІСТ

ВСТУП 8

1 АНАЛІЗ ЗАВДАННЯ І МЕТОДІВ УДОСКОНАЛЕННЯ ПРОЦЕСІВ А/В ТЕСТУВАННЯ ДЛЯ ЕФЕКТИВНОЇ АДАПТАЦІЇ ВЕБ-САЙТУ 12

1.1 Аналіз стану питання проведення тестувань для ефективної адаптації веб-сайту 12

1.2 Використання Бета тестування при проведенні А/В експериментів 17

1.3 Використання методу синтетичної контрольної групи при А/В тестуванні 19

1.4 Порівняння з аналогами 21

1.5 Постановка задач дослідження 24

1.6 Висновки 26

2 УДОСКОНАЛЕННЯ МЕТОДІВ А/В ТЕСТУВАННЯ 28

2.1 Розробка архітектури системи для А/В тестування 28

2.2 Авторизація та автентифікація користувачів 32

2.3 Проектування бази даних 35

2.4 Удосконалення методу синтетичної контрольної групи для проведення А/В тестування 38

2.5 Розробка графічного інтерфейсу 41

2.6 Висновки 44

3 РОЗРОБКА МОДУЛІВ ПЛАТФОРМИ ДЛЯ А/В ТЕСТУВАННЯ 45

3.1 Аналіз засобів розробки модуля управління експериментами 45

3.2 Аналіз вибору СУБД 53

3.3 Програмна реалізація модулів системи для проведення А/В тестування 57

3.4 Висновки 62

4 ТЕСТУВАННЯ ДОДАТКУ 63

4.1 Аналіз методів і засобів тестування 63

4.2 Автоматизоване тестування розробленого додатку 66

4.3 Ручне тестування платформи для А/В експериментів 68

4.4 Розгортання та впровадження системи 75

4.5 Висновки 81

5 ЕКОНОМІЧНА ЧАСТИНА 82

5.1 Оцінювання комерційного потенціалу розробки 82

5.2 Прогнозування витрат на виконання науково-дослідної роботи 89

5.3 Розрахунок економічної ефективності науково-технічної розробки 95
5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності 97
5.5 Висновки 99
ВИСНОВКИ 100
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 102
Додаток А. Технічне завдання 106
Додаток Б. Протокол перевірки роботи 110
Додаток В. Лістинг програми 112
Додаток Г. Ілюстративна частина 124

ВСТУП

Обґрунтування вибору теми дослідження. Підвищення ефективності користування веб-сайтом сьогодні, як ніколи актуальна тема. Кількість користувачів Інтернету з кожним днем стрімко збільшується, компанії намагаються різними шляхами заохочувати нових відвідувачів на свої сайти. Основна їхня ціль, це зробити все для того, щоб користувач провів як найбільше часу на їхньому сайті, і в майбутньому повернувся ще не один раз.

Для досягнення успішності потрібно постійно удосконалювати користувацький досвід, перевіряти нові ідеї та гіпотези [1]. Кожна деталь на веб-сайті є важливою, навіть колір кнопки та в якому куті екрану вона розташована впливає на те скільки користувачів її натиснуть. Для того, щоб перевірити, які зміни є кращими для користувачів та прийняти певні рішення, потрібно володіти даними, на основі яких можливо прийняти дійсно об’єктивні рішення.

Існують декілька основних методів для перевірки певних змін на веб-сайту, а саме A/B тестування, мультиваріантне тестування та метод багаторуких бандитів.

Метод A/B тестування має на меті розділення користувачів на певні однакові групи, за допомогою методу випадкового розподілення [2]. Такі групи обов’язково не мають впливати одна на одну, це дасть змогу отримати набір якісних даних після проведення експерименту. Дані можливо легко проаналізувати, на відміну методу мільтиваріантного тестування, де можливе поєднання великої кількості змін у різні групи, що вимагають більш складних підходів до аналізу. На основі проаналізованих даних можливо зробити висновки чи порівнювані зміни були достатньо ефективними.

Такий підхід до перевірки гіпотез є ефективним, оскільки одночасно можливо запускати декілька експериментів на одному веб-сайті та отримувати швидко початкові результати експериментів, вже через декілька днів, на відміну від методу багаторуких бандитів, який вимагає тривалого

налаштування, та пізні результати. Також, даний підхід не вимагає оновлень версій веб-сайту, при наявності всієї інфраструктури для проведення тестувань, достатньо здійснити налаштування через користувацький інтерфейс та вказати код, що модифікує сторінки, що також зменшує кількість необхідних людських ресурсів. Такий підхід до тестування є корисним для вивчення користувачів певного ресурсу.

Актуальність даної роботи полягає в тому, що ефективна адаптація веб-сайтів до вимог користувачів є важливою умовою для успішного розвитку продуктів, а використання A/B тестування для перевірки гіпотез надає можливість отримувати швидку інформацію про успішність нових змін.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів.

Відповідно до поставленої мети в роботі вирішуються такі завдання:

- проаналізувати існуючі методи проведення експериментів на веб-сайтах;
- провести аналіз існуючих підходів до проведення A/B тестувань;
- виконати аналіз існуючих платформ для проведення A/B тестувань;
- розробити структуру та алгоритми програмного продукту;
- розробити інтерфейс додатку для проведення A/B тестування;
- виконати програмну реалізацію модулів системи проведення A/B тестування;
- провести тестування програмного продукту та проаналізувати отримані результати.

Об'єкт дослідження – процес удосконалення підходів до проведення A/B тестування з врахуванням рівня адаптації.

Предмет дослідження – методи удосконалення процесів проведення

A/B тестування.

Методи дослідження. У процесі досліджень використовувались: метод синтетичної контрольної групи, методи алгоритмізації та програмної реалізації під час розробки програмних модулів, методи теорії баз даних для розробки структури бази даних.

Наукова новизна отриманих результатів.

1. Вперше було запропоновано метод поєднання A/B тестування з Бета тестуванням, який полягає в можливості ідентифікації помилок на ранніх етапах проведення експериментів, що дозволяє підвищити рівень клієнтоорієнтованості та адаптації тестування.

2. Отримав подальшого розвитку метод синтетичного контролю для проведення A/B експериментів, який, на відміну від існуючих, дає змогу порівнювати дані між групами, які мають вплив одна на одну, використовувати додаткові групи для можливості додаткової перевірки відсутності впливу внесених змін на контрольну групу, що дозволяє підвищити рівень контролю та тестування.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено програмний продукт для проведення A/B тестування, що може бути використаний компаніями для проведення тестування на власних веб-сайтах.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: методи перевірки нових ідей за допомогою користувацьких тестувань на веб-сайтах, огляд сучасних систем для проведення експериментів.

Апробація матеріалів магістерської кваліфікаційної роботи. Результати досліджень були представлені на конференціях: Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси:

створення, використання, доступ». 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021; III Міжнародній науково-практичній конференції «Theoretical and empirical scientific research: concept and trends». 10 грудня 2021 р. – Оксфорд/Вінниця: LLC OXFORD SCIENCES LTD/European Scientific Platform, 2021.

Публікації. Результати роботи були опубліковані у збірках матеріалів міжнародної науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» [3] та III міжнародній науково-практичній конференції «Theoretical and empirical scientific research: concept and trends» [4].

1 АНАЛІЗ ЗАВДАННЯ І МЕТОДІВ УДОСКОНАЛЕННЯ ПРОЦЕСІВ А/В ТЕСТУВАННЯ ДЛЯ ЕФЕКТИВНОЇ АДАПТАЦІЇ ВЕБ-САЙТУ

1.1 Аналіз стану питання проведення тестувань для ефективної адаптації веб-сайту

На сьогоднішній день у світі більше 50% відсотків населення користується інтернетом, кількість користувачів на веб-сайтах постійно збільшується.

Генерування нових ідей – це не від'ємна частина розвитку будь-якого сучасного продукту. Виникає проблема, що коли є ідея добавити нові зміни на сайт невідомо, як вони вплинуть на досвід користування кінцевими користувачами веб-сайтом. Нова ідея на даному етапі є лише інтуїтивною догадкою, яка побудована на базі власного досвіду і бачення, які можуть не завжди співпадати з баченням аудиторії певного ресурсу. Тому, оскільки є певна невизначеність, бізнес може почати втрачати користувачів, у випадку, якщо зміни негативно вплинуть на користувачів. Також цикл розробки великих змін завжди потребує багато часу та коштів, адже крім безпосередньої імплементації в себе включає дизайн, тестування, інтеграцію та підтримку, у випадку негативного впливу нових змін цілий цикл розробки буде марно витраченим часом [5].

Для того, щоб дійсно підтвердити, що певна ідея або гіпотеза дійсно буде мати позитивний вплив на відвідувачів потрібно спробувати спочатку зробити перевірочні тести, наприклад перевірити розробку на певній частині користувачів і по можливості без внесення великих змін в основну частину коду продукту, адже можливо вони виявляться не потрібними. Основна ціль це перевірити нову ідею в найбільш швидкий та дешевий з всіх можливих варіантів. Чим більше різних ідей можливо протестувати тим краще. Оскільки час є не відновлювальним ресурсом, тому краще витратити його на створення реальних змін, які не просто існують, а дійсно вирішують реальні проблеми та користуються попитом.

На сьогоднішній день існують декілька основних видів для проведення такого роду тестувань:

- A/B тестування;
- мультиваріантне тестування;
- метод багаторуких бандитів.

A/B тестування, також відоме як split тестування, порівнює дві версії веб-сторінки (інколи більше) для визначення тої яка є найкращою в плані досягнення певних встановлених цілей [6]. Зазвичай, наявна велика різниця між порівнювальними сторінками і вони можуть бути повністю різними. Цей метод тестування також може бути використаний для порівняння двох версій електронних листів, оголошень. Варіанти для порівняння називаються варіаціями.

На даний момент, такі тести вважаються найпростішою формою контрольованих експериментів [7]. Хоча, додаючи до тесту додаткові варіанти, його складність зростає. A/B тести є корисними для розуміння користувачів і їх задоволення від властивостей таких як нові функції та продукти. Сучасні тести також використовуються для проведення складних експериментів, наприклад як одні користувачі впливають на інших, як онлайн сервіси впливають на дії користувачів. Багато професій використовують зібрані дані після проведення A/B тестів: аналітики, продуктові менеджери, розробники, маркетологи. Оскільки зібрана інформація дає змогу приймати певні рішення базуючись на реальних показниках, також ці дані дають розуміння компаніям про ріст, збільшення прибутку.

Версія A може бути версією яка використовується в даний момент, тоді як версія B певною мірою модифікованою версією A. У даному випадку відбудеться порівняння відносно версії A, тому така версія буде називатися контрольною.

Гарним прикладом може бути сайт який займається продажем товарів, для того, щоб оформити замовлення користувач проходить декілька сторінок

до моменту оформлення замовлення, і на кожній сторінці яку відвідує користувач, відбувається зменшення кількості відвідувачів. Приклад тесту відображено на рисунку 1.1. Незначне збільшення кількості відвідувачів за допомогою візуальної модифікації сторінок може значно вплинути на кількість продажів в цілому.

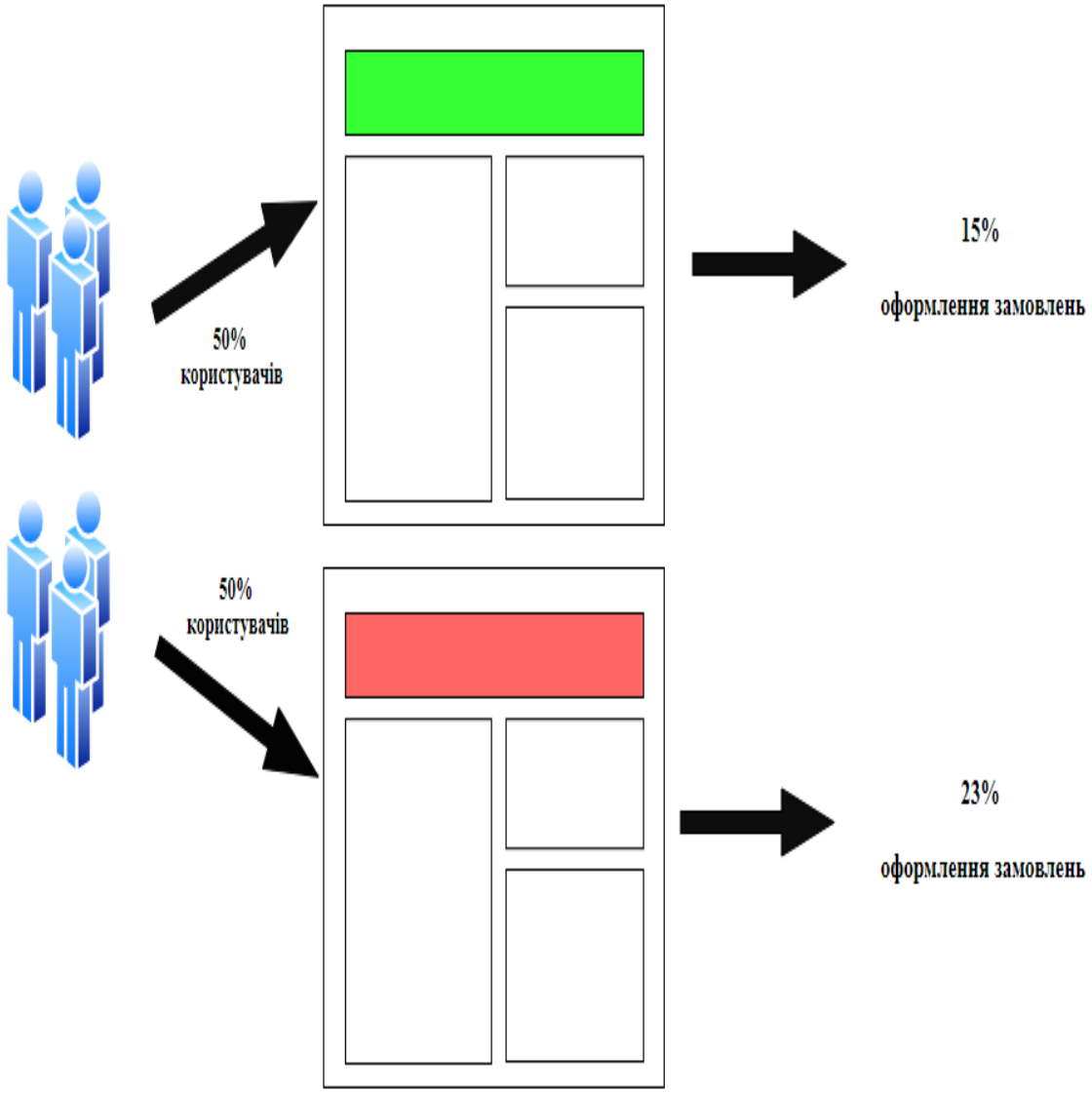


Рисунок 1.1 – Приклад A/B тесту

Істотні покращення можна побачити за допомогою таких елементів тестування як зміна тексту, макету, зображень та кольорів, але не завжди. В такому типі тестів, окремі користувачі завжди бачать одну з двох версій, оскільки основна ціль є знайти яка з двох версій є ліпшою.

Мультиваріантний тип тестування є подібним до A/B тестування, але може відбуватися тестування більше ніж двох версій одночасно чи використовуватися дві контрольних версії [8]. Основна ідея даного виду тестування це можливість визначити які об’єкти на сторінці здійснюють найбільший вплив на користувачів. Приклад мультиваріантного тесту відображено на рисунку 1.2.

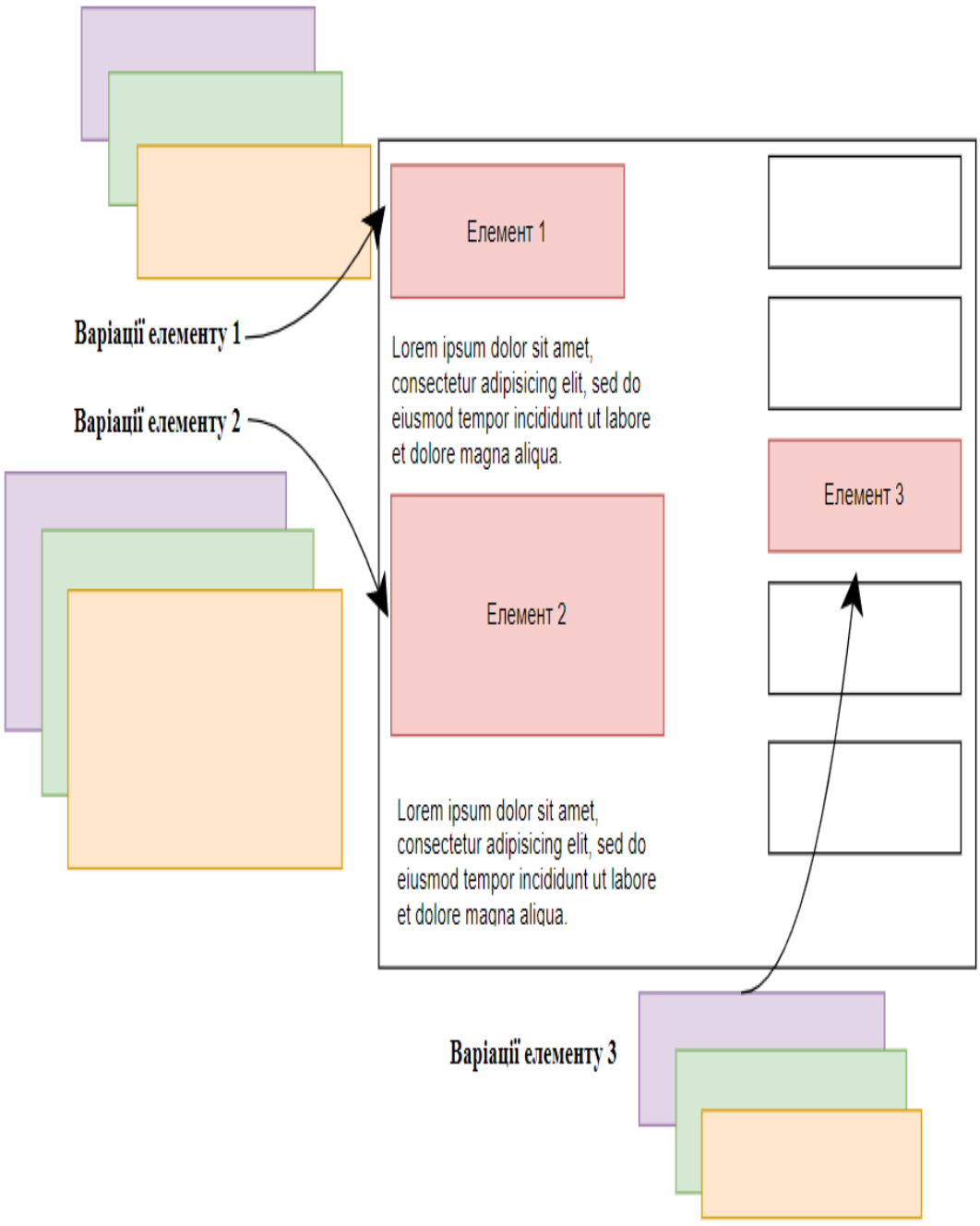


Рисунок 1.2 – Приклад мультиваріантного тесту

Даний тип тестування добре проводити після A/B тестування, для удосконалення користувацького досвіду. Для проведення мультиваріантного тестування спочатку потрібно мати велику кількість користувачів на сайті, оскільки для отримання значимих результатів потрібно, щоб кожну комбінацію відвідала достатня кількість користувачів.

Більшість компаній відають перевагу A/B тестуванню оскільки бажають протестувати значимі зміни, які потенційно сильно впливають на сторінку. Також A/B тести простіші в проведені та є простішими у подальшому аналізі, оскільки окремі варіації містять зміни по всій сторінці одночасно.

Метод багаторуких бандитів – це A/B тести, які обновляються в реальному часі на основі ефективності кожної варіації [9]. Основна ідея даного виду тестування, що після певного проміжку часу один з варіантів експерименту витіснить інші і буде мати весь трафік (рис 1.3). Основна перевага даного методу в тому, що пом’якшується втрата конверсії, яку бізнес відчуває при тестуванні потенційно гіршого варіанту. Варто відмітити, що для прийняття рішень потрібно, щоб був вибраний один або декілька показників на основі яких буде прийматися рішення.

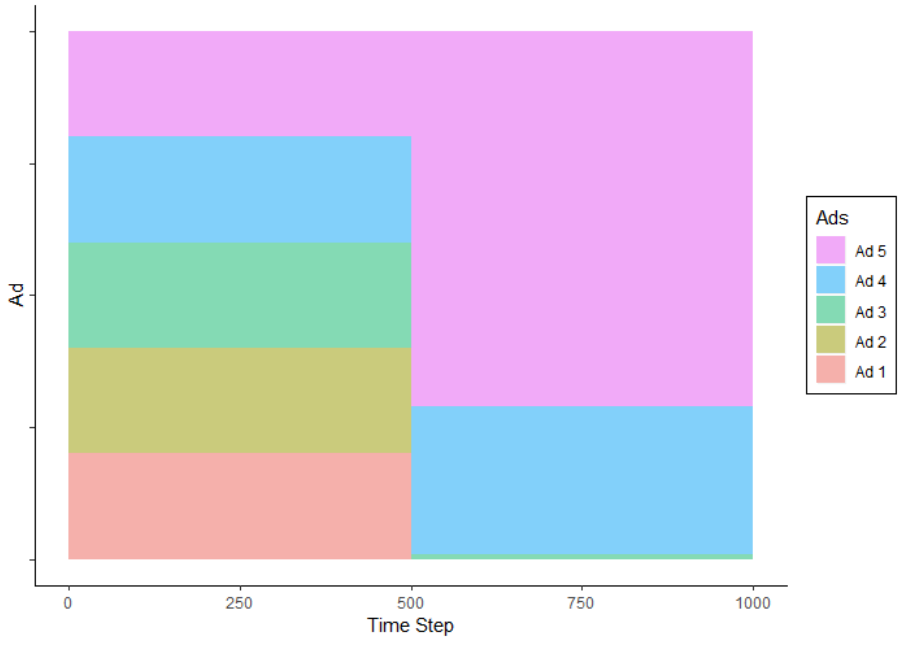


Рисунок 1.3 – Приклад розподілу трафіку з часом у методі багаторуких бандиті

Як бачимо, сучасні методи проведення тестування є досить розвинуті та мають свої переваги. У даній реалізації буде використовуватися метод A/B тестування, оскільки він не потребує великої кількості відвідувачів сайту, є досить простим для налаштування та подальшого аналізу.

1.2 Використання Бета тестування при проведенні A/B експериментів

При появі нової гіпотези завжди з’являється бажання, як найшвидше її перевірити у реальних умовах. Для швидкої перевірки ідей процес створення A/B експериментів є досить простий та не вимагає будь-яких створень нової версії продукту та оновлень версій веб-сайту на серверах. Достатньо лише заповнити необхідні дані через платформу та підставити код, що буде змінювати сторінки. Такий спрощений підхід до появи змін у реальних користувачів завжди збільшує кількість помилок, які не були виявлені на ранніх етапах розробки і вперше знаходяться вже під час того, як користувачі отримали певні зміни.

Помилки можуть мати різний вплив на показники успішності продукту, кожна з них залежить від того, на скільки вона виявилась видимою користувачам, наприклад, у випадку якщо не працює сторінка оплати замовлень, то бізнес повністю втрачає клієнтів та прибутковість.

На сьогоднішній день існує не багато методів, для запобігання таким помилкам. Одним з них є збільшення етапів тестування, а саме покриття коду, який використовується лише в експерименті, unit тестами. Цей тип тестів в цілому є досить швидкий та не вимагає додаткових налаштувань для запуску таких тестів, також вони мають високу стабільність. До мінусів такого підходу можна віднести те, що зазвичай код для експериментів пишеться досить спрощено, без підтримки майбутнього розширення. Тому, якщо варіація певні виявиться дійсно виграшною, основний код буде переписуватися відповідно до стандартів, а тести буде необхідно написати

знову. У випадку, якщо варіація виявиться програшною, час розробників витрачений на написання тестів виявиться марно витраченим.

Ефективним методом вирішення даної проблеми може бути поєднання проведення A/B тестування у поєднанні з Бета тестуванням.

Бета тестування має на меті випуск нового продукту на обмежену кількість користувачів, для знаходження дефектів [10]. Такий підхід дає змогу здійснити тестування у реальному світі з використанням реальних користувачів продукту. Якщо під час такого тестування з'являються дефекти, вони вплинуть лише на малу кількість користувачів і як результат на загальних показниках успішності компанії не будуть відображатися. На рисунку 1.4 зображено приклад розподілу трафіку з часом під час A/B експериментів та в Бета тестуванні.

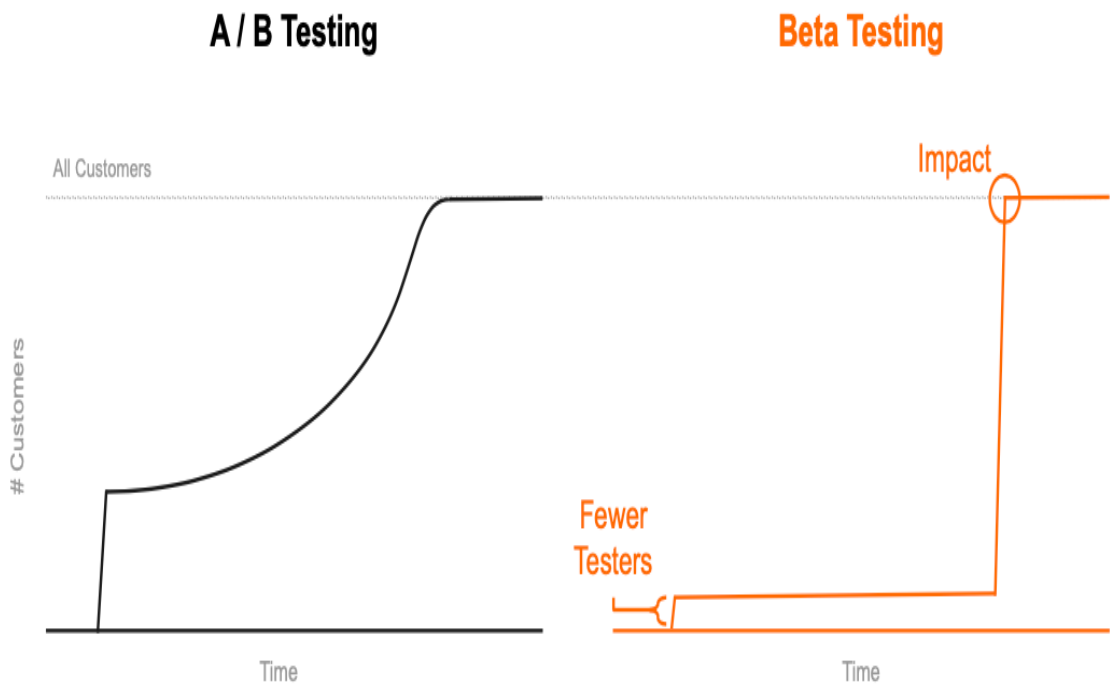


Рисунок 1.4 – Приклад розподілу трафіку з часом в експериментах

Цей метод можливо ефективно використати при проведенні A/B тестування. Потрібно встановити на змінену версію сторінки досить малий відсоток трафіку, в залежності, де саме сторінка знаходиться і скільки

користувачів в загальному її бачить, це можуть бути значення від 1 до 5 відсотків. І запустити тест на декілька днів у такому вигляді. Під час його проведення здійснювати активний моніторинг, і в випадку, якщо ніяких аномалій не зафіксовано, необхідно просто збільшити розподіл трафіку на реальний, який планувався у заданому експерименті.

1.3 Особливості методу синтетичної контрольної групи в A/B тестуванні

A/B тестування це є стандарт для встановлення чи є певні відносини причино-наслідкові. За допомогою випадкового призначення контрольної групи та варіації, гарантується, що в середньому розподіл між групами є рівним (рис 1.5). У такому випадку можливо зробити покращення для однієї групи і спостерігати різницю, що буде спричинена скоріш за все новими змінами.

Також є важливим мати можливість встановити причино-наслідковий зв’язок для даних де не можливо зробити випадковий розподіл. Найчастіше такі випадки трапляються, коли одна група має прямий вплив на іншу. На перший погляд, можливим варіантом вирішення даної проблеми може бути врахування всіх можливих факторів впливу, але у реальному світі передбачити всіх факторів не можливо.

Метод синтетичної контрольної групи – це статичний метод, що використовується для оцінки ефекту втручання в порівняльних дослідженнях.

Даний метод можливо використати при проведені тестування, оскільки він є досить ефективним методом, та водночас простим, створення контрольної групи на даних, що можливо спостерігати [11].

Для того, щоб скористатися даним методом при тестуванні потрібно спочатку вибрати окрему контрольну групу, що буде використовуватися для передбачення значень штучної контрольної групи. Дана група не повинна отримати ніякий вплив від змін, які будуть використовуватися у варіації.

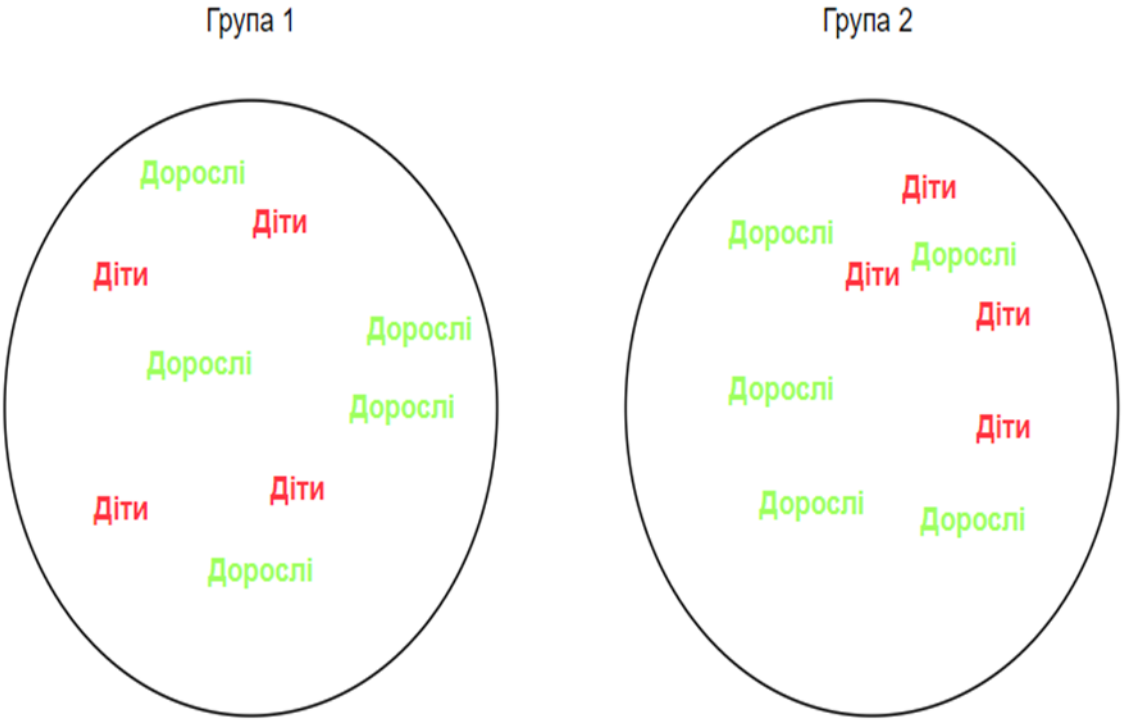


Рисунок 1.5 – Приклад рівного розподілу між групами

Після обрання групи, потрібно визначити предиктор, параметр, що відповідає за прогнозування. Він має спостерігатися у контрольній та у варіаційній групах. Це може бути будь що, що немає систематичний вплив змінами, які присутні у варіаційній групі.

Наступним етапом є обрання коефіцієнтів, що мають на меті зменшення відстані між предикторами в контрольній групі і в варіаційній. В результаті даного кроку, можливо отримати набір коефіцієнтів, що можуть зробити контрольну групу схожою на варіаційну.

Оскільки, було визначено, як відноситься контрольна група до варіаційної, можливо використати дані з контрольної та варіаційної групи для створення синтетичної групи, до якої не було добавлено змін (рис 1.6). В подальшому можливо проаналізувати групу з штучним контролем та варіаційну, по аналогії з аналізом звичайного A/B тесту.

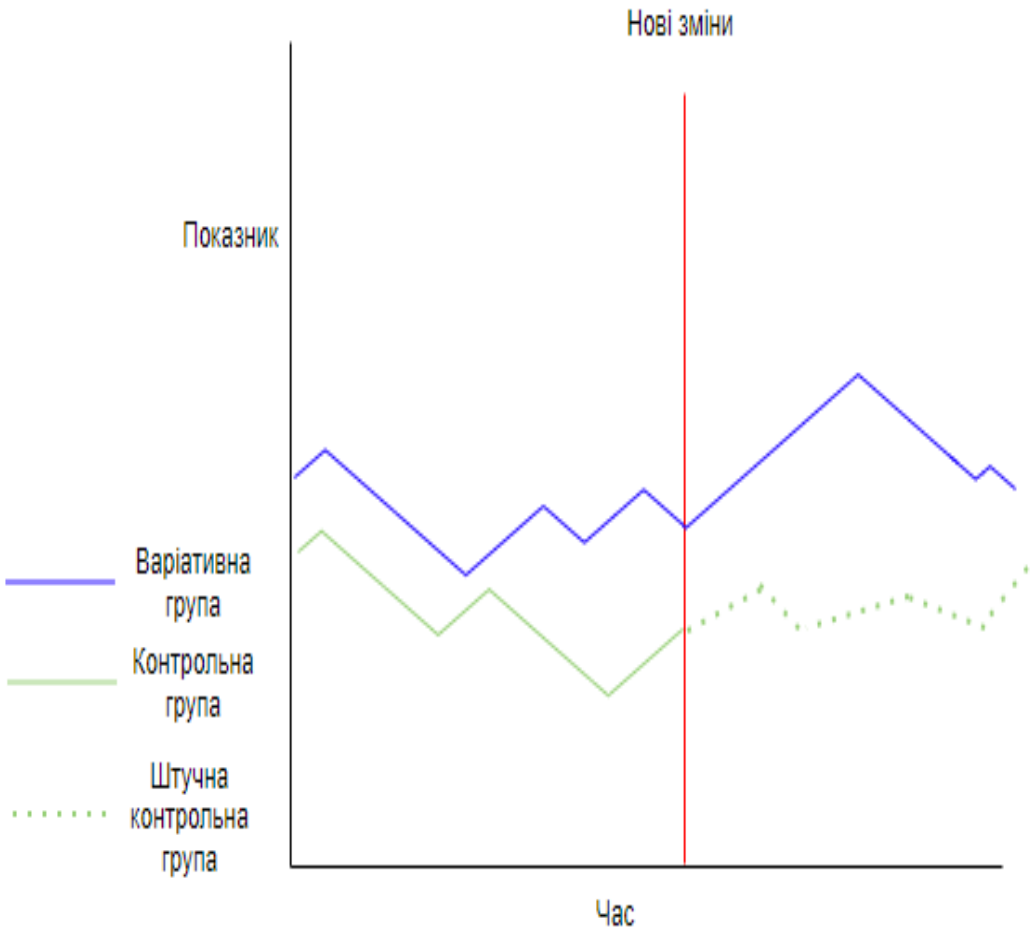


Рисунок 1.6 – Метод синтетичної контрольної групи

Основним недоліком даного методу є важливість забезпечення відсутності впливу змін, що були додані до варіаційної групи на контрольну групу.

1.4 Порівняння з аналогами

На сьогоднішній день існує досить невелика кількість сторонніх сервісів, що дають змогу проводити A/B тестування.

Проаналізувавши ринок, можна виокремити три основні аналоги розроблюваного сервісу «ProTesting».

Optimizely – це американський сервіс, що надає платформу для цифрового дослідження, послуги свої надає на основі моделі software as a service [12]. Платформа підтримує можливість проведення A/B (рис. 1.7) та мультиваріантних тестів, створювати персоналізацію веб-сайтів, feature

toggle функціональність, також можливість керування контентом веб-сторінок. Окрім можливості керування експериментами через веб версію платформи, є можливість встановлення додатку для Windows системи. Інтерфейс у платформи однаковий, як для створення мобільних експериментів під iOS та Android, так само і під веб версії.

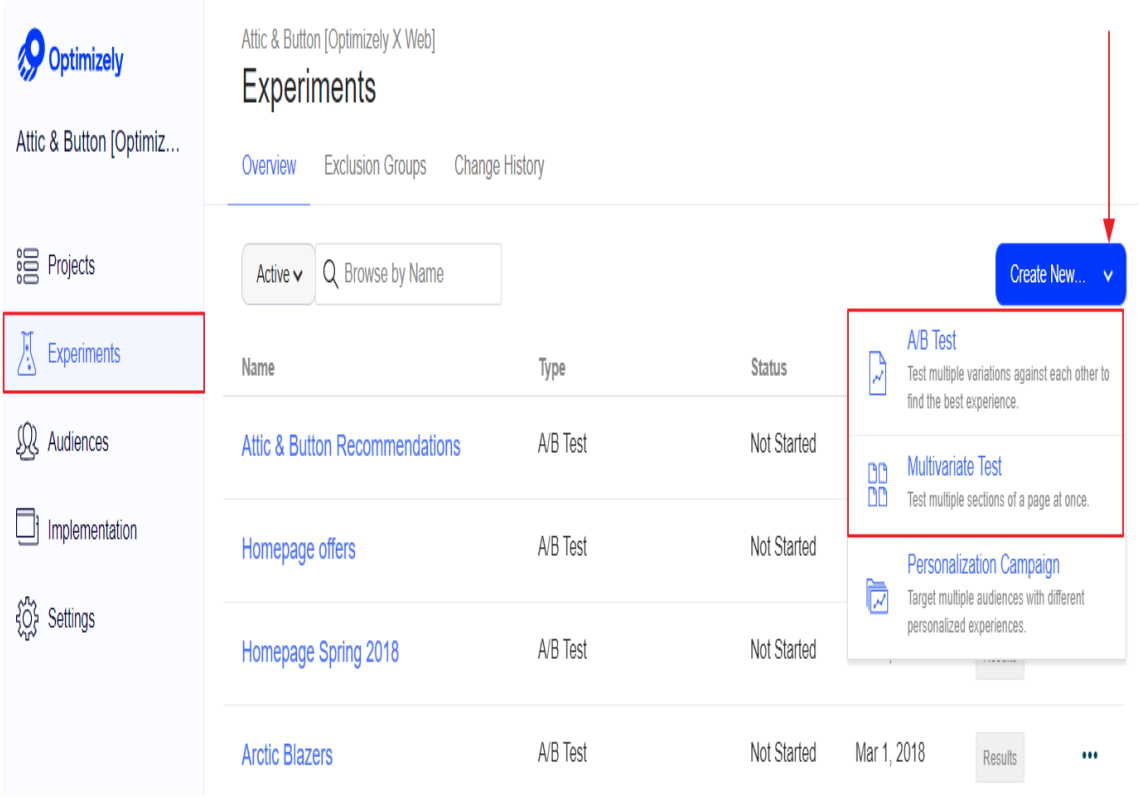


Рисунок 1.7 – Створення A/B тесту на платформі Optimizely

Дана платформа є лідером на ринку A/B тестування, тому кількість різної функціональності для проведення тестів є досить висока: проведення тестів на основі геолокації, формування користувацьких звітів, планувальник експериментів.

Google optimize – це безкоштовний сервіс для оптимізації веб-сайтів, що допомагає онлайн маркетологам збільшити задоволення користувачів за допомогою постійного тестування різних змін [13]. Його функціональність теж включає багато можливостей, що пов’язані з підняттям конверсії сторінок (рис. 1.8). Сервіс належить компанії Google, як результат має якісну

підтримку роботи з Google Analytics. Google optimize може протестувати будь-який елемент що знаходиться на сторінці включаючи виклики до методів, шрифти, заголовки, картинки. Також є підтримка тестування різних комбінацій елементів на сторінках.

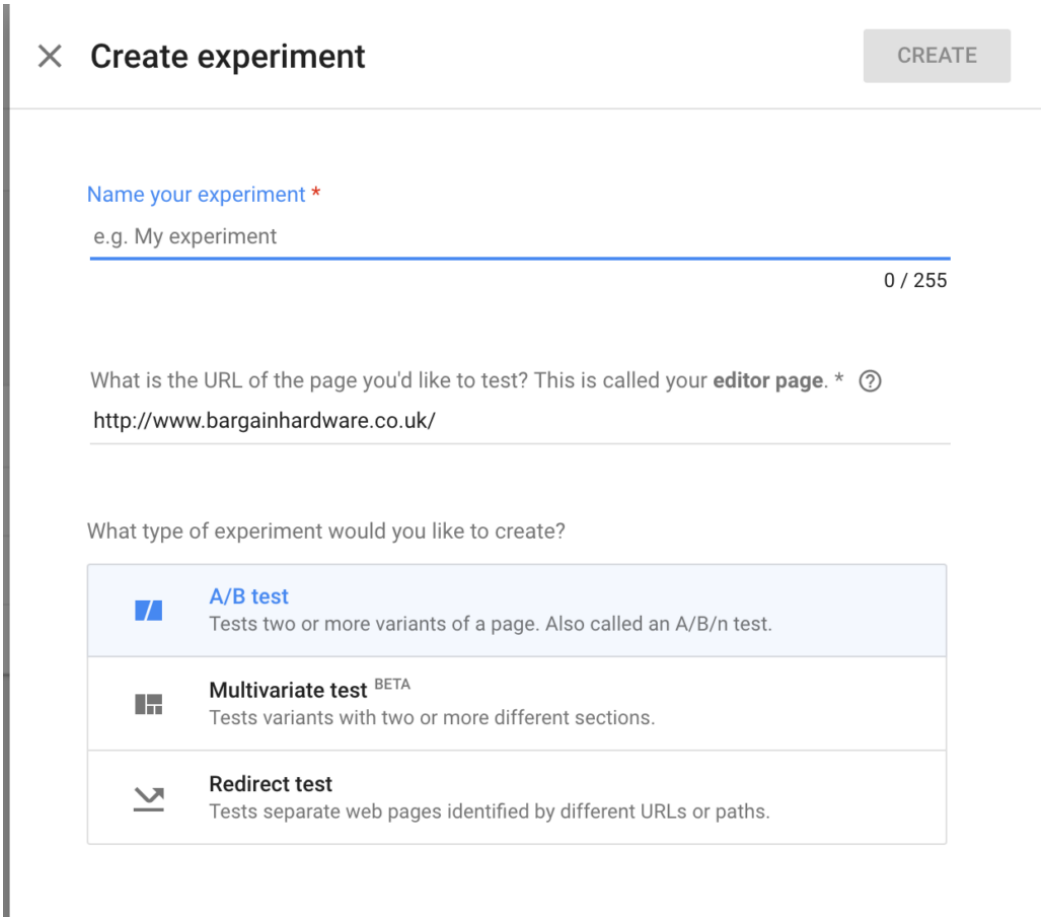


Рисунок 1.8 – Створення A/B тесту на платформі Google optimize

VWO – ще один сервіс для проведення тестувань який з'явився досить недавно, але вже користується великою популярністю. Сервіс працює швидко та має зрозумілий користувацький інтерфейс. Існує підтримка вбудованих віджетів, які можна використати, щоб процес розробки нового тесту зайняв як найменше часу [14]. Тестування проходить з побудовою зручних графіків та потужними можливостями до розподілу користувачів, що дає змогу приймати розумні бізнес рішення. Також підтримується можливість пошуку падіння метрик, таких як кількість користувачів, що стали клієнтами веб-сайту. Існує підтримка проведення серверних

експериментів та мобільних. Також, є можливість створення експериментів у вигляді опитувальників прямо з платформи.

Аналізуючи вищеописані платформи, можна виокремити для кожної з них ряд недоліків. Порівняння їх з розроблюваною системою описано в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Назва	Можливість окремого розгортання системи	Підтримка поєднання з Бета тестуванням з автоматичним переключенням	Підтримка оновлення метрик в реальному часі по кожній варіації	Підтримка календаря тестування	Підтримка targeting умови для повторного виконання тесту
Optimizely	Ні	Ні	Ні	Так	Ні
Google optimize	Ні	Ні	Так	Ні	Ні
VWO	Ні	Ні	Так	Так	Ні
ProTesting	Так	Так	Так	Так	Так

Таким чином, розроблювана платформа «ProTesting» має деякі переваги перед існуючими аналогами, тому її розробка є актуальною.

1.5 Постановка задач дослідження

Для визначення задач для реалізації варто розглянути основні можливі варіанти створення та проведення A/B тестів.

В першу чергу, платформа для тестування повинна підтримувати певну авторизацію, з однієї сторони це дасть змогу захистити платформу від не

санкціонованого доступу, а з іншої контролювати хто вносив певні зміни до експериментів.

При відкриті системи повинна бути можливість швидкого створення експерименту за допомогою кнопок керування. Для того, щоб створити експеримент потрібно вказати його назву, в якій описати коротко основну мету та сторінки на яких будуть відбуватися зміни в процесі проведення експерименту. Після створення експерименту, в користувача повинна бути можливість створити нову варіацію для експерименту, окрім контрольної групи.

Кожна варіація тесту повинна підтримувати можливість додання JavaScript коду який буде виконуватися під час попадання користувача на конкретну варіацію та в результаті модифікувати сторінку.

Для того, щоб визначити умову при якій користувач повинен попасти на створюваний експеримент потрібно, щоб була реалізована підтримка задання умов призначення експерименту, ці умови краще теж задавати JavaScript кодом, адже це дасть можливість бути гнучким, і завжди мати можливість підлаштовуватися під конкретну сторінку та дані з неї, по яких можна визначати умови призначення експерименту.

Кожен експеримент має містити період часу під час якого він планується проводитися. На основі цих даних можливо побудувати календар тестування, щоб різні користувачі могли зрозуміти, що їхній запланований експеримент не перетинається з іншими. Це є важливим фактором, оскільки де які експерименти, особливо які проводяться на одних й тих самих сторінках можуть ламати досвід користування сайтом для користувачів, також декілька експериментів можуть впливати на одну й ту саму метрику, і після проведення буде не можливо оцінити який саме експеримент вплинув на певну метрику.

Також для того, щоб знати коли достатньо даних зібрано для того, щоб в подальшому мати можливість аналізувати та коли експеримент повинен бути зупинений, повинна бути реалізована можливість задання обсягу

вибірки (sample size), що є кількістю користувачів які були залучені до експерименту [15].

Розподіл трафіку є важливим елементом у кожній подібній системі, оскільки при змінах на досить чутливих сторінках, наприклад сторінка введення кредитної карти, бізнес може відчути сильне падіння прибутку, тому бажано досить низький відсоток трафіку відавати на тестові сторінки, які тестуються в перший раз, тому повинна бути можливість встановлення будь-якого розподілу між різними варіаціями, але при умові що сума буде рівна 100%.

Отже, основні задачі, які необхідно реалізувати у розроблюваній системі:

1. Реєстрація та авторизація.
2. Поділ профілів на два типи.
3. Відображення календарю експериментів.
4. Можливість створення нового експерименту.
5. Запуск та зупинка експерименту.
6. Зберігання інформації ким зроблені були зміни.
7. Створення нових варіацій до експериментів.
8. Встановлення дат проведення експерименту.
9. Задання обсягу вибірки для експерименту.
10. Підтримка різного розподілу трафіку.
11. Можливість видалення варіації.
12. Підтримка поєднання A/B тестування з Бета тестуванням з автоматичним переключенням.

Виконання саме цих задач дозволить розробити якісний та корисний продукт.

1.6 Висновки

У першому розділі проаналізовано важливість удосконалення проведення A/B експериментів для ефективної адаптації веб-сайту до вимог

користувачів та виконано аналіз предметної області, на основі чого було підтверджено доцільність розробки власної системи.

Було вперше запропоновано використання Бета тестування у поєднанні з A/B тестуванням, що дає змогу зменшити негативний ефект у випадку наявних дефектів.

Розглянуто метод синтетичного контролю для проведення A/B експериментів, що дає змогу порівнювати дані між групами, які мають вплив одна на одну.

Було розглянуто існуючі системи для проведення A/B тестування, а саме Optimizely, Google optimize та VWO. Проаналізовано їх основні переваги та недоліки на основі чого було підтверджено необхідність створюваної платформи «ProTesting», яка буде об’єднувати функціональність та усувати основні недоліки аналогів.

Також було детально розглянуто основний функціонал який буде присутній у додатку та визначено основні задачі.

2 УДОСКОНАЛЕННЯ МЕТОДІВ А/В ТЕСТУВАННЯ

2.1 Розробка архітектури системи для А/В тестування

Розроблювальна система складається з двох основних частин, з веб-додатку, який дає змогу повністю керувати експериментами та з посилання на код, який містить всі зміни які модифікують сторінки та умови всіх запущених в даний момент експериментів. Він буде добавлятися на кожну сторінку де має відбуватися підтримка А/В тестування.

Архітектура системи описує її основні компоненти, їх взаємозв'язки (структури) та те, як вони взаємодіють між собою [16]. Архітектура та дизайн програмного забезпечення включає декілька головних факторів, таких як бізнес-стратегія, атрибути якості, дизайн та IT-середовище (рис 2.1). Архітектура служить проектом системи. Вона забезпечує абстракцію для управління складними системами та встановлення механізму зв'язку та координації між компонентами. Вона визначає структуроване рішення, яке відповідає всім технічним та експлуатаційним вимогам, одночасно оптимізуючи загальні атрибути якості, такі як продуктивність та безпека.

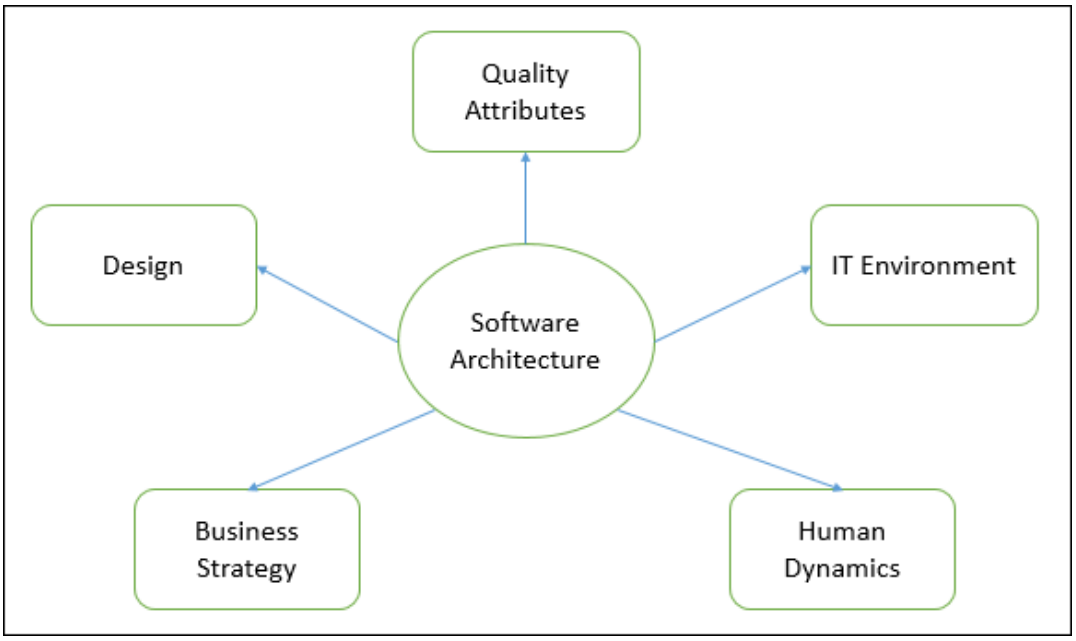


Рисунок 2.1 – Елемент архітектури додатків

Архітектура веб-додатків включає в себе опис взаємодії між системами, базою даних і різними допоміжними системами. Така структура має на меті передбачити, що декілька додатків можуть працювати одночасно (рис 2.2) [17]. При потраплянні на сайт, відбувається запит на сервер, який після обробки повертає сторінку, яку буде вимальовано у браузері, що дасть змогу працювати з сайтом кінцевому користувачу.

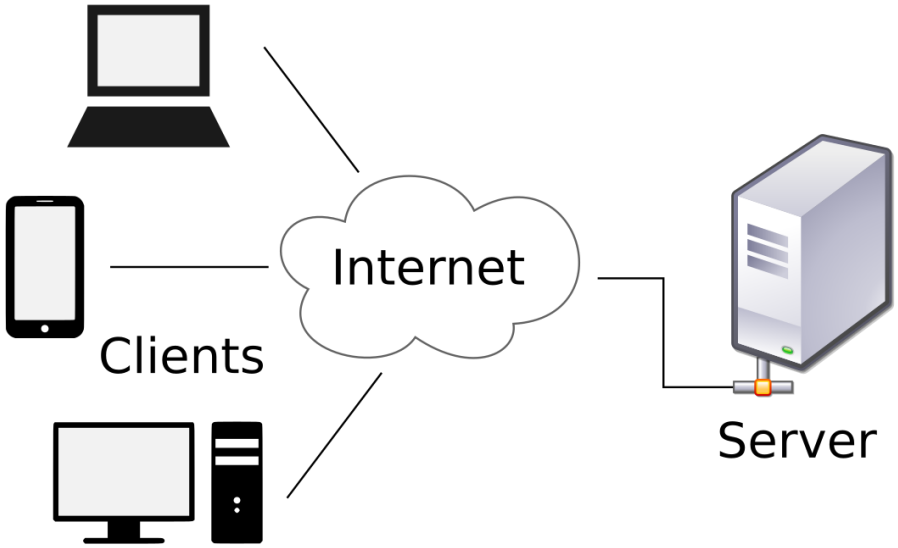


Рисунок 2.2 – Приклад клієнт-серверної архітектури

Платформа для керування експериментами є саме веб-додатком, це не вимагає встановлення додатку на комп’ютер, оскільки сьогодні існує досить багато різних типів комп’ютерної архітектури, це значно ускладнює підтримку додатків під певну платформу. У веб-додатках складність підтримки мінімізована, оскільки основні браузери, якими користується більша частина інтернет користувачів мають подібні вимоги до розроблювальних сайтів. Також архітектура веб-додатків є досить незамінною, так як більшість інтернет трафіку відбувається саме за допомогою веб комунікації.

Отже, для розуміння, як працюють складні веб-додатки їхня архітектура повинна включати всі заплановані модулі та їх детальний опис у вигляді різного роду діаграм.

Узагальнена модель роботи платформи представлена на рисунку 2.3. Вона відображає зв’язок між компонентами серверної частини та клієнтської, та взаємодію компонент з базою даних, що використовується для збереження всієї інформації, що потрібна для повторного використання. В програмі використовується дві основних клієнтських компоненти, перша є веб-платформою, що відповідає за всю конфігураційну функціональність пов’язану з експериментами, інша частина відповідає лише за скрипт, що буде використовуватися на самих веб-сайтах для можливості підтримки A/B тестування.

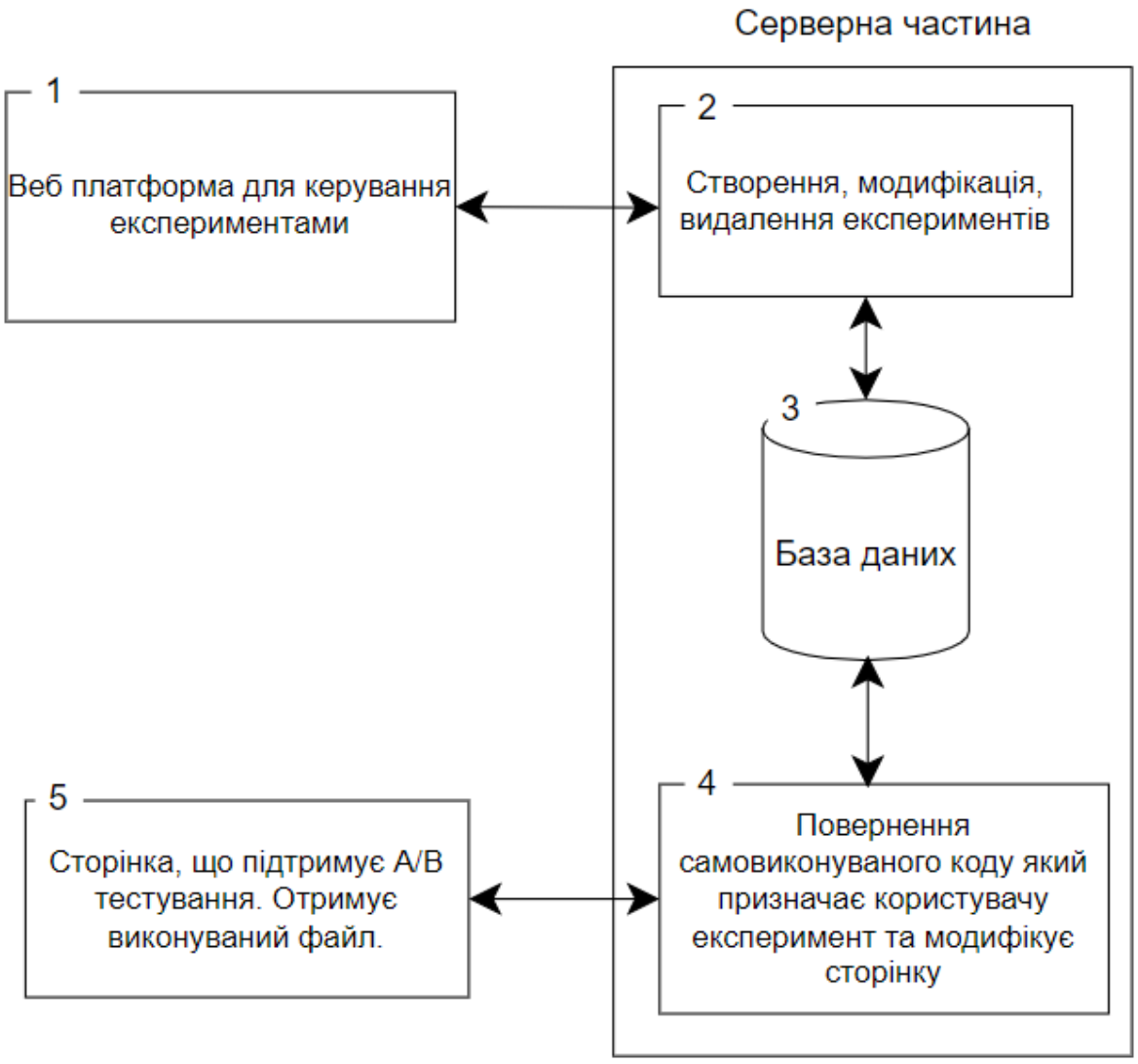


Рисунок 2.3 – Узагальнена модель роботи системи

Архітектуру додатку наведено на рисунку 2.4.

Оскільки платформа працює як веб-сайт, то для коректної роботи необхідно мати доступ до інтернету.

Для стабільної роботи веб-додатку і можливості його масштабування, внутрішня архітектура повинна бути без зайвих ускладнень, у місцях де можливо спростити реалізацію та складатися з простих компонент, які можливо вилучати, при необхідності.

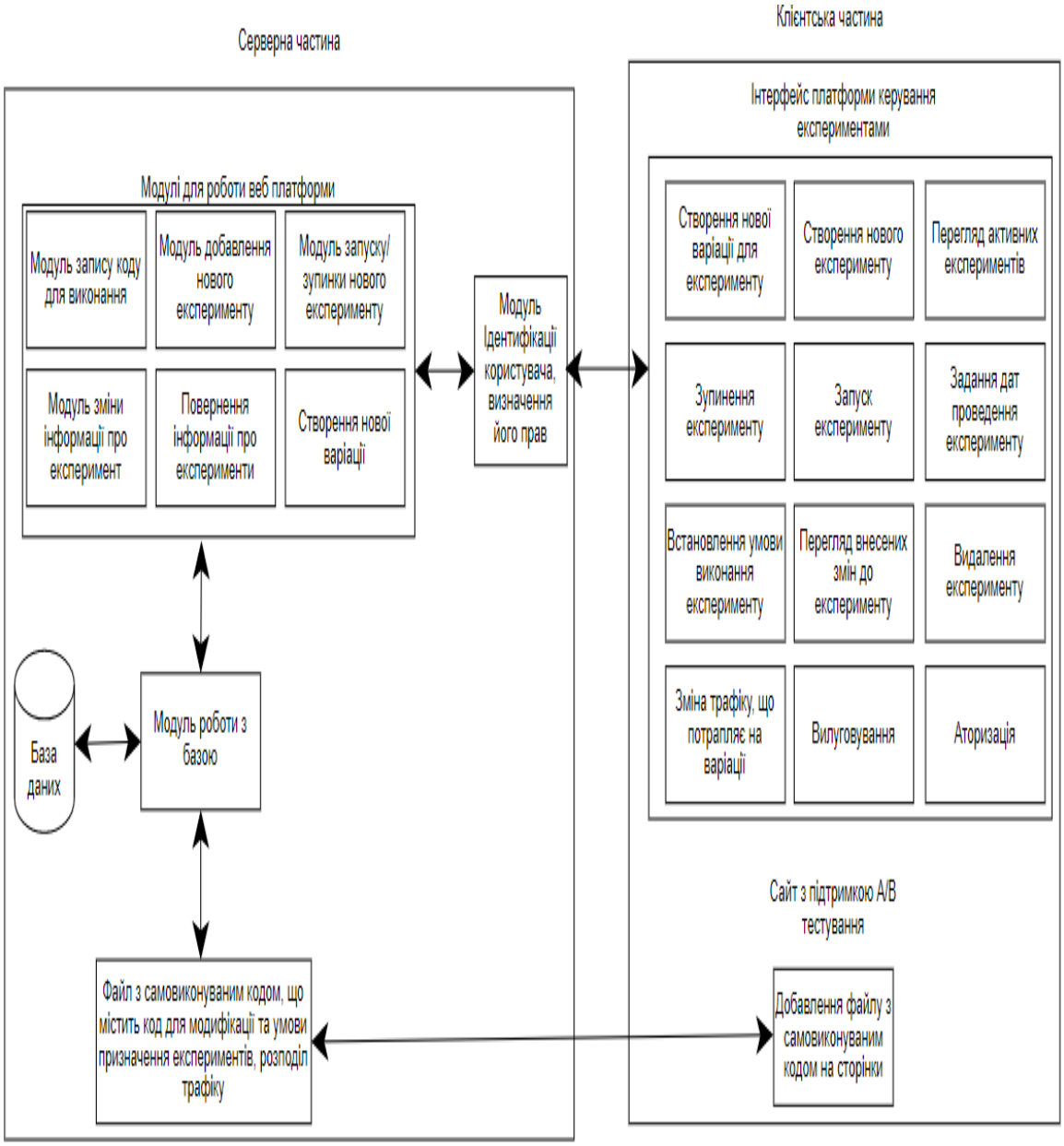


Рисунок 2.4 – Архітектура додатку

Розробка складається з двох незалежних частин, серверної та клієнтської. Серверна частина теж поділяється на складові в залежності від типу запиту він обробляється різними, логічно поділеними, модулями. Всі модулі мають внутрішній поділ на ті що модифікують або вставляють дані та на ті, що лише витягують дані без будь-яких змін, такий підхід надає змогу легко розрізняти методи між собою. Також, окремою частиною серверної частини є модуль, що підготовлює скрипт з активними експериментами та їх варіаціями, який буде виконуватися на сайті, що підтримує тестування.

Запропонована архітектура платформи дозволить створити систему для ефективної обробки запитів, та яка буде безперебійною у роботі з можливістю масштабування.

2.2 Авторизація та автентифікація користувачів

Контрольований доступ до ресурсів у сучасному світі є важливим елементом, оскільки веб-додатки працюють у всесвітній мережі, де кожен може зробити необхідний запит на ресурс. Така відкритість додатків, вимагає різних підходів до захисту додатків від не санкціонованого втручання. Оскільки, у випадку, якщо хакер отримує доступ до системи він може повністю зруйнувати існуючу інфраструктуру.

Авторизація має на меті управління рівнями та засобами доступу до різних об’єктів системи та ресурсів взалежності від ідентифікатора та пароля користувача [18].

Автентифікація – це є процес ідентифікації певного користувача та надання йому прав для доступу до окремих ресурсів.

Кожна платформа для веб-розробки має вбудовані інструменти для авторизації та автентифікації. Вони дають змогу захистити методи та обмежувати ресурси в залежності від того яку роль присвоєно користувачу,

наприклад адміністратор повинен мати більше повноважень ніж звичайний користувач.

Сучасні методи для авторизації і автентифікації використовують токени, як методи передачі даних, які містять в собі достатньо інформації, що авторизувати користувача чи виконати певний запит до ресурсу. Токени це просто набір певної інформації. На сьогодні одним з найпопулярніших видів токенів для авторизації є JSON Web Token (JWT). Це JSON об’єкт, що визначений у відкритому стандарті RFC 7519. Він являється одним з найбільш безпечних методів передачі інформації між двома учасниками. Для його виконання необхідно визначити заголовок з загальною інформацією про токен, корисні дані, такі як ID користувача або його ролі [19].

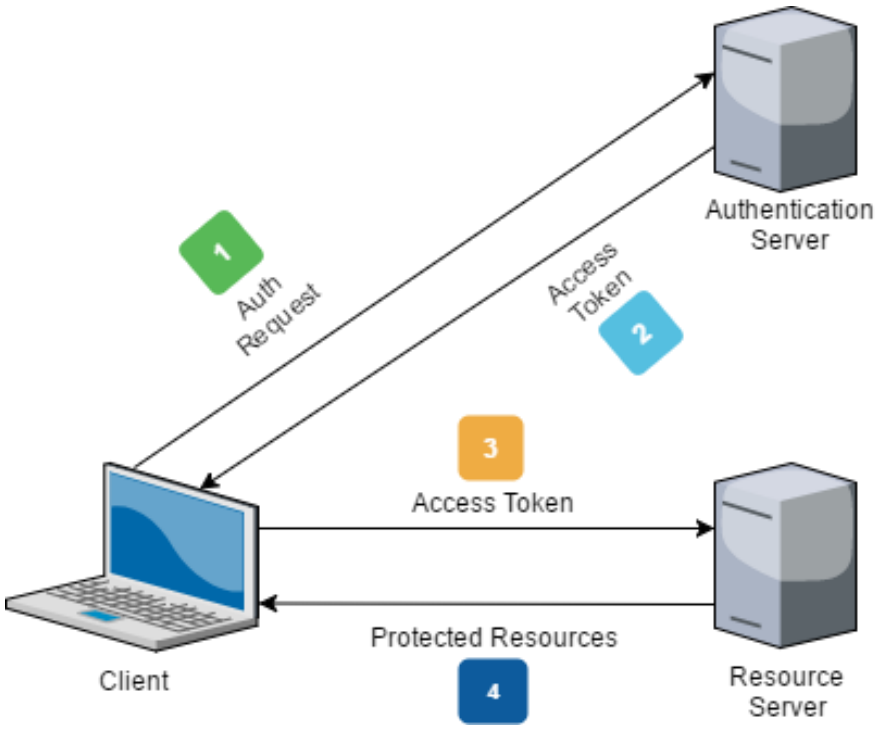


Рисунок 2.5 – Приклад роботи з токенами

На рисунку 2.5 відображено приклад роботи з токенами. Основною умовою є сервер автентифікації. Його відповідальність це повернення користувачу токена, у випадку якщо користувач вказав правильний ідентифікатор та пароль. Токен може бути підписаний за допомогою будь-якого сучасного алгоритму, що немає відомих методів взлому. Потім,

створений токен можливо використати для доступу до інших захищених ресурсів. Ресурси, що обробляють запити мають містити приватний ключ, що буде мати змогу розкодувати вміст токену.

Токени повинні містити в собі час протягом якого вони дійсні. Оскільки, це дає додатковий захист, якщо користувач випадково не вилогувався з веб-сайту, наступний хто отримає доступ до комп’ютера зможе робити дії від імені користувача. Для того, щоб оновити токен необхідно знову ввести власний ідентифікатор та пароль.

Даний підхід з авторизаційними токенами варто використати у розроблюваному додатку, оскільки він має змогу достатньо захистити він несанкціонованого доступу до системи.

При роботі з токенами можливо отримати доступ до системи і діяти від імені користувача у випадку, якщо токен буде перехоплено. Раніше це була поширена проблема, у часи коли існував лише протокол HTTP для передачі даних в інтернеті, особливо в публічних мережах, наприклад в інтернет кафе.

Існують спеціальні програмі, які не вимагають складного налаштування і надають можливість зчитувати весь трафіку, який проходить через роутер. В результаті таких дій хакер може отримати викрасти токен користувача і діяти від його імені. Для того, щоб заборонити таку можливість використовується сучасний протокол HTTPS для роботи з секретною інформацією.

HTTPS – це є протокол поверх HTTP, але використовує додатковий рівень безпеки передачі даних. За допомогою нього шифруються всі дані між користувачем та сервером. Це дає змогу мати повну конфіденційність інформації, яку відправляє користувач та сервер.

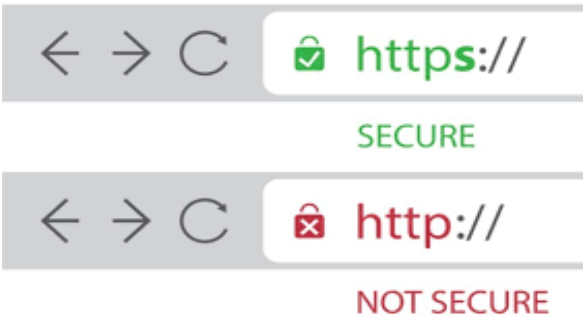


Рисунок 2.6 – Вигляд інформації у браузері про використаний протокол

Під час використання протоколу HTTPS, браузер відповідає за перевірку того, що спілкування відбувається напряму з сервером, до якого відбуваються запити. Також за цільність даних, що передаються мережею. На рисунку 2.6 зображено вигляд браузеру у випадку коли сайт використовує захищений протокол та не використовує.

Для того, щоб платформа працювала з HTTPS необхідно встановити на сайт SSL сертифікат, що виданий авторизованим провайдером [20].

Використання HTTPS протоколу в парі з токенами надасть надійний захист платформі для проведення A/B тестування.

2.3 Проектування бази даних

Створення бази даних слід починати з її проектування.

Проектування бази даних - це процес створення схеми бази даних і визначення необхідних обмежень цілісності.

Основними завданнями проектування бази даних є:

1. Забезпечення зберігання в БД всієї необхідної інформації.

2. Забезпечення можливості отримання даних по всім необхідним запитам.

3. Скорочення надмірності і дублювання даних.

4. Забезпечення цілісності бази даних.

Платформа повинна зберігати інформацію про всі створені експерименти, їхній стан на даний момент, а саме зупинені чи виконуються, варіації експерименту, код який буде модифікувати сторінки. Інформація про експерименти має зберігатися у базі даних, оскільки не є тимчасовою.

Розробка універсального відношення є першою частиною проектування бази. Воно складається з інформації про всі атрибути даних в одній таблиці, яка, при потребі, може бути декомпозована [21]. У невеликих проектах, універсальне відношення може стати відправною точкою при проектуванні. База даних додатку повинна містити атрибути представлені в таблиці 2.1.

На основі створеної таблиці можна виділити сутності та їх атрибути:

1. User (<Id>, Name, Password, Login, UserType).

2. Experiment (<Id>, Name, StartDate, EndDate, IsActive, AssignmentCondition, AuthorId).

3. Variation (<Id>, Name, ExperimentId, CreateDate, ExecutionCode, TrafficPercentage, AuthorId).

Таблиця 2.1 – Перелік атрибутів універсального відношення

№	Назва атрибута	Ім’я поля	Коментар
1	ID користувача	UserId	Ідентифікатор користувача
2	Ім’я користувача	Name	Ім’я користувача
3	Логін користувача	Login	Логін користувача
4	Тип користувача	UserType	Тип користувача
5	Пароль користувача	Password	Пароль користувача
6	ID експерименту	ExperimentId	ID експерименту
7	Назва експерименту	Name	Назва експерименту
8	Дата початку	StartDate	Дата початку
9	Дата закінчення	EndDate	Дата закінчення
10	Умова призначення експерименту	AssignmentCondition	Умова призначення експерименту
11	Автор експерименту	AuthorId	Ідентифікатор автора, що створив експеримент
12	Ідентифікатор	VariationId	Ідентифікатор варіації

	варіації		
13	Назва варіації	Name	Назва варіації
14	Дата створення	CreateDate	Дата створення варіації
15	Код для виконання	ExecutionCode	Код для виконання на сторінці

Продовження табл. 2.1

№	Назва атрибута	Ім’я поля	Коментар
16	Відсоток трафіку на варіації	TrafficPercentage	Відсоток користувачів які будуть потрапляти на варіацію
17	Користувач, що створив варіацію	AuthorId	Користувач, що створив варіацію
18	Ідентифікатор експерименту	ExperimentId	Ідентифікатор експерименту для варіації

Відповідно до вище описаних сутностей було створено базу даних, структура якої зображено на рисунку 2.4

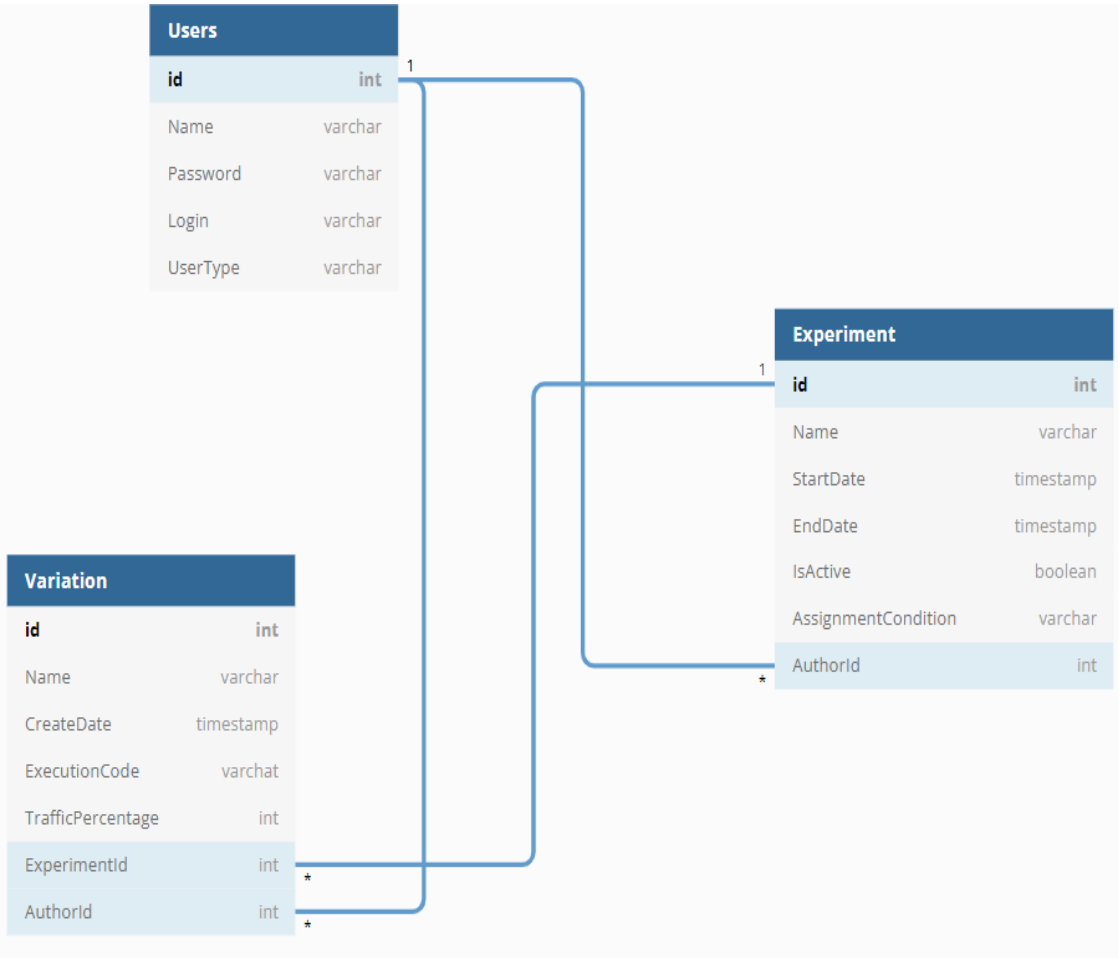


Рисунок 2.4 – Структура бази даних

Сутність Users знаходиться у відношенні 1:N до сутності Experiment та Variation, Experiment до сутності Variation знаходиться також у відношенні 1:N, так як користувач може мати декілька створених експериментів та варіацій, а кожен експеримент може мати у свою чергу декілька варіацій. Як результат, сутність Experiment має одне додаткове поле «AuthorId», а Variation два додаткових поля «AuthorId» і «ExperimentId»

2.4 Удосконалення методу синтетичної контрольної групи для проведення A/B тестування

У випадку використання методу синтетичного контролю, при проведенні A/B тестування, є важливим забезпечити відсутність впливу варіаційної групи на контрольну, що була синтетично створена.

Для того, щоб зробити перевірку, що дійсно зміни впливу не мали, вперше пропонується, додатково обрати ще одну або декілька груп, для яких

не буде здійснюватися будь-яких змін. Це дасть змогу перевірити відсутність впливу варіаційної групи на контрольну. Після обрання групи, необхідно визначити коефіцієнти для даних груп по відношенню до контрольної групи. З наявністю коефіцієнту можливо будувати синтетичні групи для додаткових груп. Очікуваний результат є відсутність будь-яких змін між різними контрольними групами після внесення основних змін до варіаційної групи. Така перевірка дає можливість підтвердити, що зміни до варіаційної групи, не мають будь-якого впливу на основну контрольну групу також.

Для реалізації даного методу необхідно розробити підтримку можливості додання додаткових груп до експерименту, що створений для проведення з використанням синтетично створеної групи. Це можливо зробити за допомогою запуску нових експериментів для таких груп, які лише будуть збирати необхідні дані.

На основі зібраних даних повинна бути реалізована підтримка відображення всіх експериментів, що були запущені, як одна група. Також має бути можливість встановлення для обраних груп необхідних коефіцієнтів, для можливості порівняння. На основі зображених даних можливо переконатися, що дійсно варіаційна група не мала впливу на контрольну групу, оскільки контрольна група у такому випадку, буде мати поведінку схожу до додатково створених груп.

Алгоритм роботи додатку з використанням удосконаленого методу на стороні клієнта, а саме браузерна частина керування експериментами, наведено на рисунку 2.5.

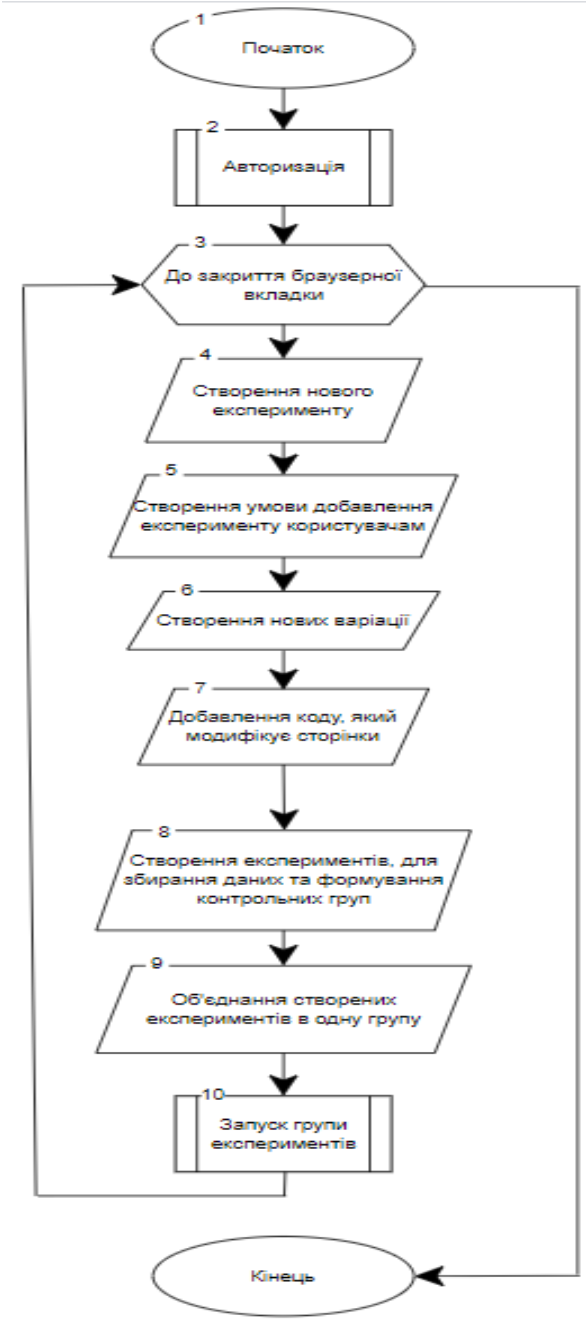


Рисунок 2.5 – Алгоритм роботи програми

Взаємодія користувача з платформою починається з авторизації. Після успішної авторизації у користувача є можливість створити новий експеримент, добавити нові варіації до експерименту та змінити код який модифікує сторінки, також встановити відсоток користувачів які будуть потрапляти на конкретні варіації. В налаштуваннях є можливість вказувати період під час якого буде відбуватися проведення експерименту. Після налаштування експерименту користувач має можливість запустити експеримент. Також є можливість переглянути календар всіх запланованих та

в даний момент запущених експериментів. Платформа працює до її закриття вкладки браузера користувачем, всі дані зберігаються в базі.

2.5 Розробка графічного інтерфейсу

Веб-інтерфейс – це інтерфейс реалізований за допомогою можливостей веб-браузера та відображений на сторінці і безпосередня взаємодія з яким відбувається у вікні веб-браузера.

Інтерфейс поділяється на користувацький досвід (UX) та користувацький інтерфейс (UI).

Рисунок 2.6 – Елементи ефективного інтерфейсу

UX – це скелет майбутнього сайту, який має на меті комплексний підхід до взаємодії користувача з інтерфейсом, виник у результатів покращення користувацького досвіду, чи отримав клієнт позитивний досвід від користування певним продуктом, а UI – це його візуальне втілення та детальне опрацювання [22]. На рисунку 2.6 зображено, які пункти включає в себе ефективний інтерфейс.

Дизайн розроблювальної платформи має бути достатньо простим, адже не правильно створений експеримент може вплинути на поведінку веб-сайту,

який підтримує тестування, що може вплинути на різні показники успішності бізнесу в цілому. Всі дії користувача мають бути для нього інтуїтивно зрозумілі.

Структурну схему розроблюваної платформи зображено на рисунку 2.7.

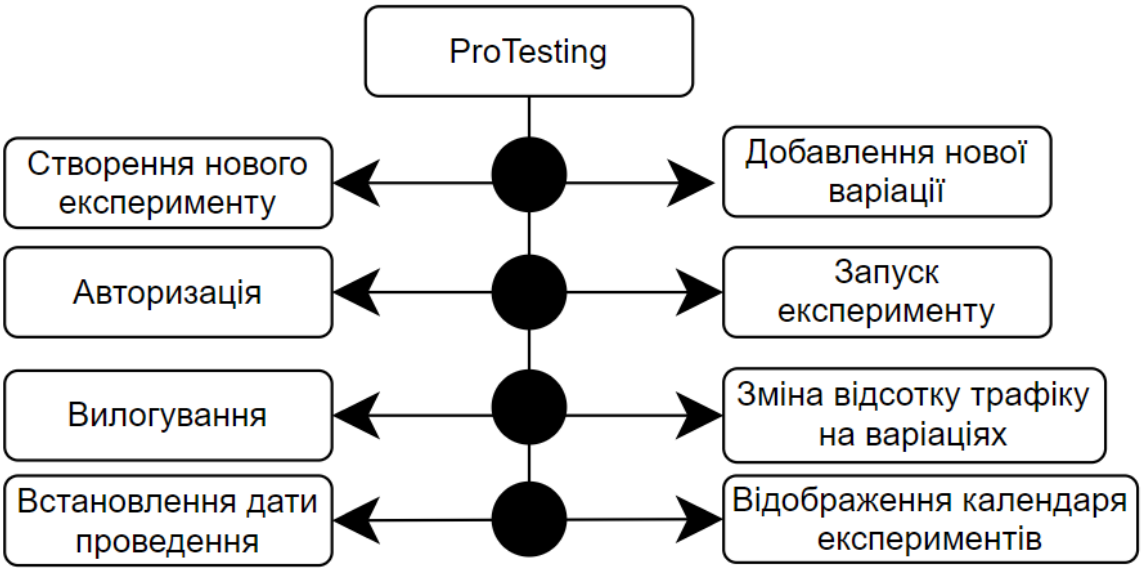


Рисунок 2.7– Структурна схема додатку

Додаток складається з 8 основних вікон керування. Вікно створення нового експерименту надає можливість додавати нові експерименти, вказувати умови при яких користувач може на них потрапляти. Після створення експерименту можливо перейти до вікна встановлення дати проведення експерименту, що може бути в подальшому використано в календарі тестування. Календар теж виступає окремим вікном, та надає змогу у зручний спосіб переглянути заплановані та в даний момент запущені експерименти. Добавлення нової варіації це вікно на яке можливо потрапити з вікна керування експериментами, воно надає змогу створити нову варіацію зі вказанням назви та виконуваного коду. Зміна відсотку трафіку надає можливість гнучко керувати відношенням кількості користувачів, що будуть потрапляти на певні варіації. Також, одним з основних вікон додатку є вікно

налаштування нового експерименту, на рисунку продемонстровано доступ до панелі керування основними функціями експерименту (рис. 2.8).

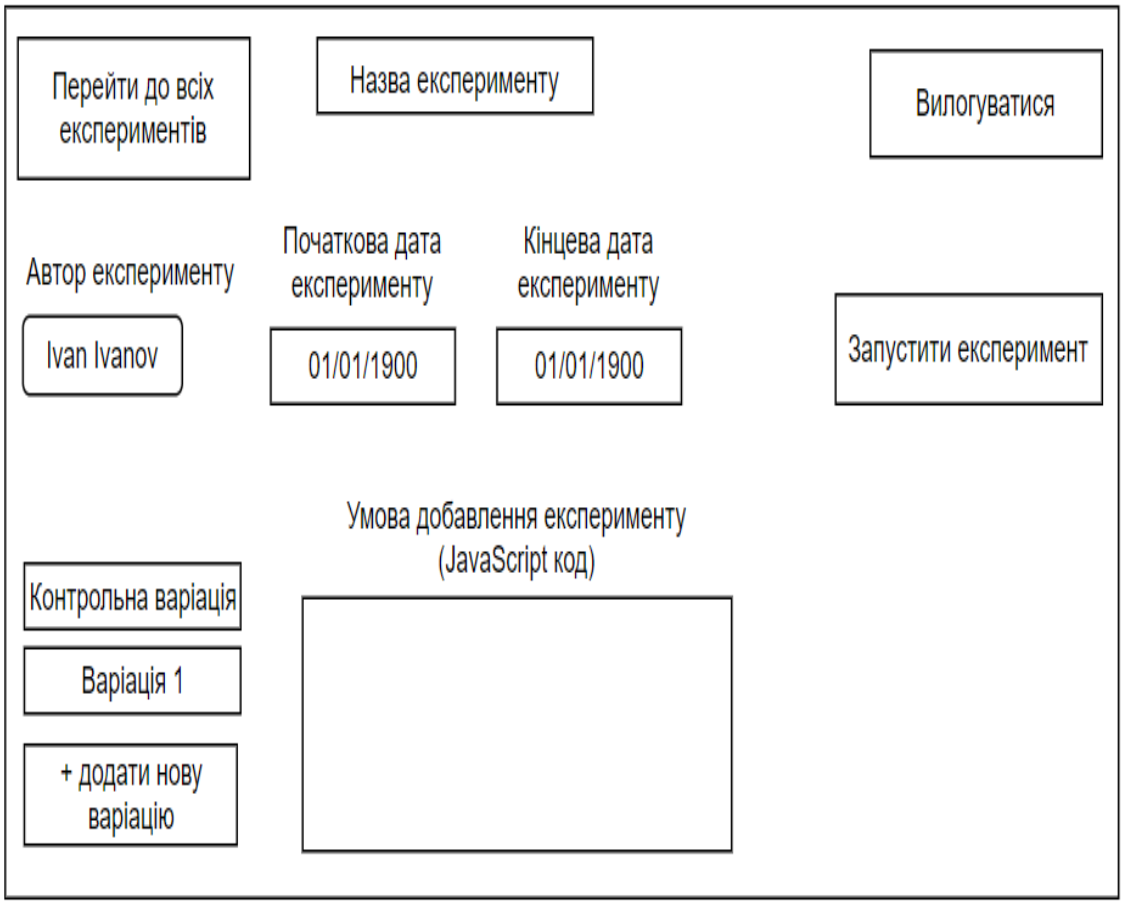


Рисунок 2.8 – Вікно налаштування експерименту

Варіації в A/B експерименті відповідають за групи користувачів, які створюються у результаті проведення певного експерименту. Крім безпосередньо назви, варіації також містять в собі інформацію про відсоток трафіку який повинен потрапляти на неї та самовиконуваний код. Такий код, модифікує сторінку у випадку, якщо було визначено, що користувачу присвоюється дана варіація.

На сторінці налаштування є можливість добавити нову варіацію, додати умову при якій експеримент повинен буде добавлений користувачу, також вказати кількість користувачів які очікується, що будуть добавлені на експеримент під час його проведення. Присутня кнопка запуску експерименту, при натисненні на яку відбувається перевірка наявності хоча б

однієї варіації та присутній умови добавлення експерименту. Також наявні кнопки для швидкого переходу до списку всіх експериментів, де є можливість відфільтрувати лише по активних експериментах, ті які виконуються в даний момент.

Створений користувацький інтерфейс є зрозумілим на інтуїтивному рівні, користування додатком буде зручним та ефективним, адже можливо буде створити новий експеримент у досить короткий проміжок часу.

2.6 Висновки

У другому розділі було представлено узагальнену модель роботи системи та розроблено загальну архітектуру платформи для проведення A/B, що поділяється на клієнтську та серверну частину. Складено універсальне відношення, що складається з 18 атрибутів, на основі якого було спроектовано реляційну базу даних та відображено її структуру. База складається з 3 основних сутностей, а саме Users, Experiments та Variations. Удосконалено метод синтетичного контролю для можливості проведення тестування з подальшою перевіркою відсутності впливу варіаційної групи на контрольну. Розроблено загальний алгоритм роботи платформи. Також, було описано складові ефективного інтерфейсу, зображено структурну схему платформи, та розглянуто інтерфейс одного з вікон, що відповідає за створення та налаштування нового експерименту.

3 РОЗРОБКА МОДУЛІВ ПЛАТФОРМИ ДЛЯ A/B ТЕСТУВАННЯ

3.1 Аналіз засобів розробки модуля управління експериментами

Веб-додатки у сучасному світі можливо розробляти майже за допомогою будь-якої мови програмування. Це пов’язано з значною популярністю веб програмування. Оскільки кожна мова має певну свою специфіку, що робить її унікальною, варто визначити яку мову буде найбільш актуально використати для розробки платформи для A/B тестування.

Для детального порівняння, було обрано три з найбільш популярних мов для веб-розробки, а саме PHP, JavaScript та C#.

Мова PHP – одна з найпростіших серверних мов для вивчення, яка містить багато вбудованих функцій, також розроблено багато фреймворків під цю мову, оскільки існує велика підтримка зі сторони розробників, це значно пришвидшує написання коду. Розроблена для створення динамічних та інтерактивних веб-сторінок. PHP дозволяє веб-сайтам проводити обробку даних в реальному часі [23]. Приклад коду на мові PHP наведено на рисунку 3.1.

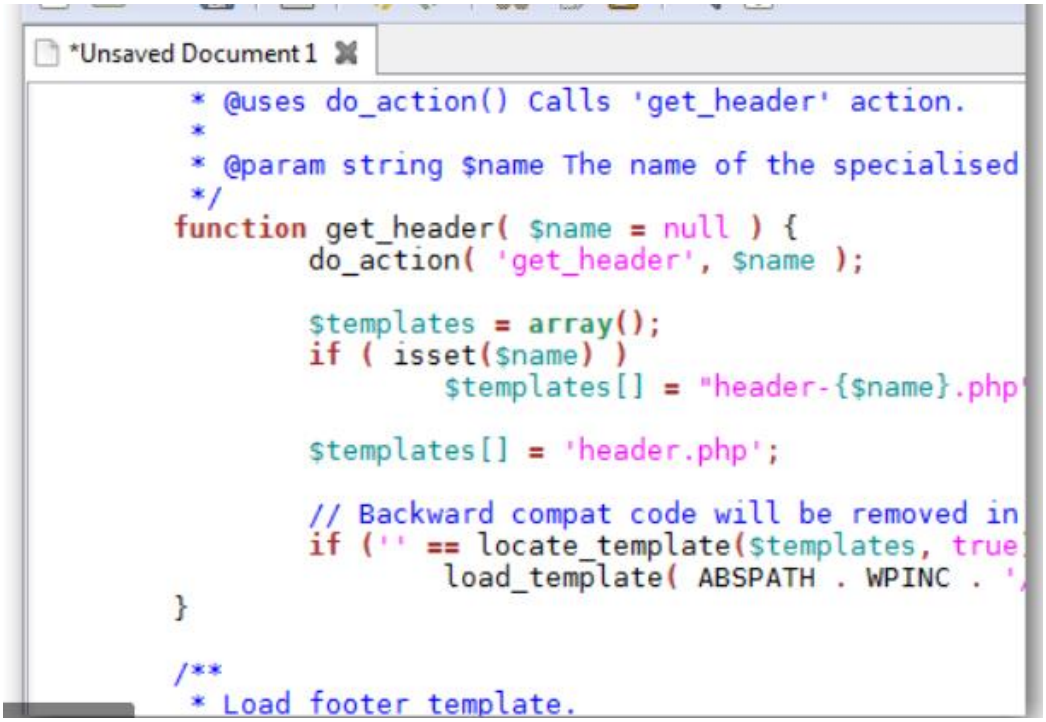


Рисунок 3.1 – Приклад коду на мові PHP

JavaScript – динамічна мова програмування, яка сьогодні використовується для веб-розробки, в веб-додатках, для розробки ігор та багато інших цілей. Вона дає змогу добавити динамічну функціональність для веб-сторінки, яку не можливо зробити за допомогою простого HTML та CSS [24]. Зовсім до недавна, мову не сприймали всерйоз, і вважали, що вона лише потрібна для простих змін на сторінках, але з часом вона потрапила у кожну нішу сучасного програмування і завдяки своїй простоті завжди знаходить своїх прихильників. Мова підтримує парадигму об’єктно-орієнтованого програмування. Одним з небагатьох мінусів мови є відсутність підтримка багатопоточності. В інтернеті є досить багато інформації про дану мову, тому при виникненні питань є можливість завжди швидко знайти потрібну відповідь, завдяки сильно розвинутій спільноті. Приклад коду на мові JavaScript наведено на рисунку 3.2.

```
fs = require 'fs'
express = require 'express'
app = express()

app.use express.bodyParser()
app.use app.router
app.use express.static(__dirname + '/public')
app.set 'views', __dirname + '/views'
app.set 'view engine', 'jade'

nouns = fs.readFileSync('nouns.txt', 'utf8').split('\n')

nurble = (text) ->
  text = text.toUpperCase()
  words = text.toLowerCase().replace(/[^a-z ]/g, '').split(' ')

  for word in words
    if word not in nouns
      re = new RegExp "(\\b)#{word}(\\b)","i"
      replacement = '$1<span class="nurble">nurble</span>$2'
      text = text.replace re, replacement

  text.replace  /\n/g, '<br>'

app.get '/', (req, res) ->
  res.render 'index'

app.post '/nurble', (req, res) ->
  res.render 'nurble', {nurble: nurble req.body.text}

app.listen 3005
```

Рисунок 3.2 – Приклад коду на мові JavaScript

C# - сучасна мова, що запозичила багато своїх характеристик з попередників таких як C та C++. Вона може виконувати багато різних задач та бути задіяною в різних нішах програмування. Її ключові концепти допомагають будувати інтерактивні середовища і володіють функціональністю, які сучасні веб-платформи вимагають. Велику популярність C# мова здобула завдяки підтримці компонент, які можливо перевикористати, що пришвидшує розробку, її синтаксис дуже схожий на C++ та Java, типи даних всередині мови є більш гнучкими, містять статичну типізацію, тому відсутня можливість помилок [25]. Також, ця перевага дає мати строгу кодову базу, тому підтримка великих і складних проблем не буде проблемою. Це є мова високого рівня, тому для того, щоб зрозуміти основну ідею коду інколи достатньо знати лише англійську мову, також завдяки цьому розробник може концентруватися на самому програмування, а не на деталях низького рівня. Мова розроблена компанією Microsoft, завдяки

цьому з кожним роком виходять певні покращення мови та з’являються нові можливості. Приклад коду на мові C# наведено на рисунку 3.3.

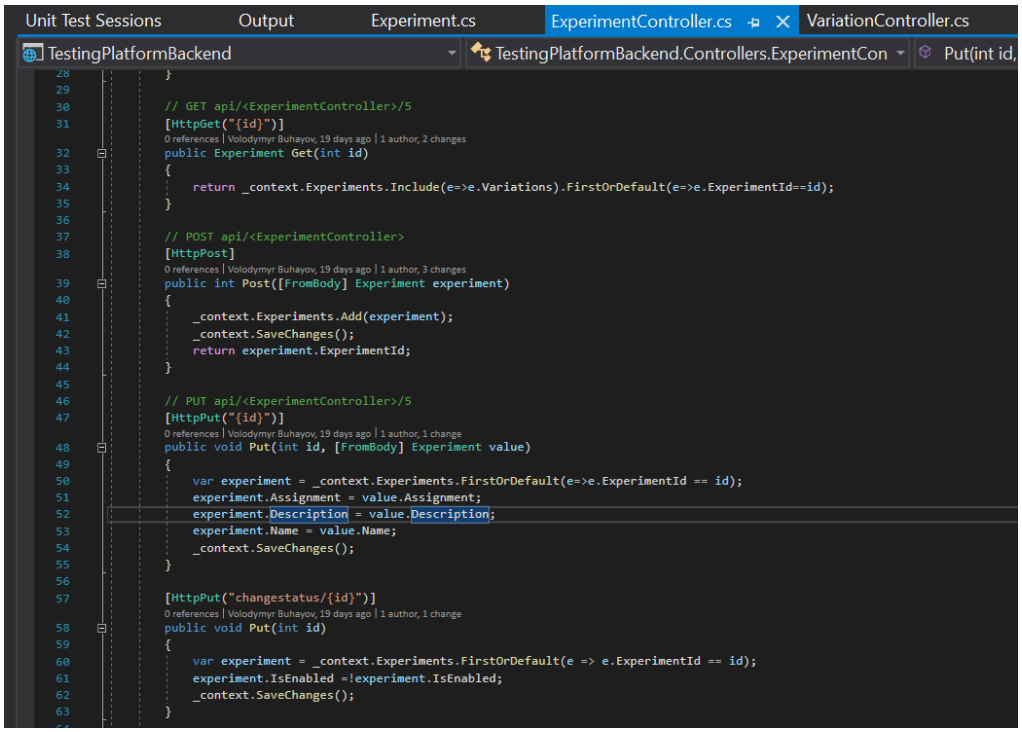


Рисунок 3.3 – Приклад коду на мові C#

Для того, щоб обрати мову для розробки платформи для A/B тестування доцільно створити таблицю порівняння (таблиця 3.1).

Таблиця 3.1 – Порівняння мов програмування

Мова програмування	PHP	JavaScript	C#
Асинхронність	-	+	+
Багатопоточність	+	-	+
Статична типізація	-	-	+
Клієнтська та серверна розробка	-	+	+
Загалом	1	2	4

Проаналізувавши результати порівняння видно, що C# підтримує додатковий функціонал, який є відсутній в інших порівнювальних мовах програмування, для створення серверної частини платформи буде доцільним обрати мову C#.

Для створення веб-додатку потрібно також обрати платформу на основі якої буде відбуватися розробка. Сучасні платформ для веб-розробки мають свої переваги та недоліки, для того, щоб обрати найбільш оптимальну для платформи потрібно здійснити порівняння. Оберемо для порівняння чотири найбільш популярних платформи для розробки, а саме Node.js, Laravel, Flask та ASP.NET MVC.

Node.js – одна з наймолодших платформ. Основна популярність даної платформи повязана з мовою програмування JavaScript, оскільки платформа платформа саме для цієї мови розроблена, а мова набула великої популярності за останні роки. Дана платформа дає можливість писати серверну частину додатків розробникам які раніше займалися лише розробкою клієнтських додатків, оскільки мова є спільна. Це є досить вигідно для компаній, оскільки всі розробники можуть бути універсальними і розуміти в цілому, як працюють додатки і в разі чого вносити зміни в будь-якому місці, де це необхідно. Також існує багато додаткових бібліотек, які можливо використати під час написання коду, це значно пришвидшує процес розробки [26]. Дана платформа є оптимальним рішення для невеликих розробок.

Laravel – платформа для розробки веб-додатків з використанням мови PHP. Вона з’явилась у якості заміни своєму попереднику CodeIgner, але включила в себе багато нових переваг. Платформа підтримує локалізацію, власну систему для аунтефикації та авторизації користувачів, роутінг між різними сторінками. Даний фреймворк має відкритий код, тому кожен може переглянути існуючий код, та можливо навіть запропонувати власні зміни. Структуру проекту з використанням платформи Laravel наведено на рисунку 3.4.

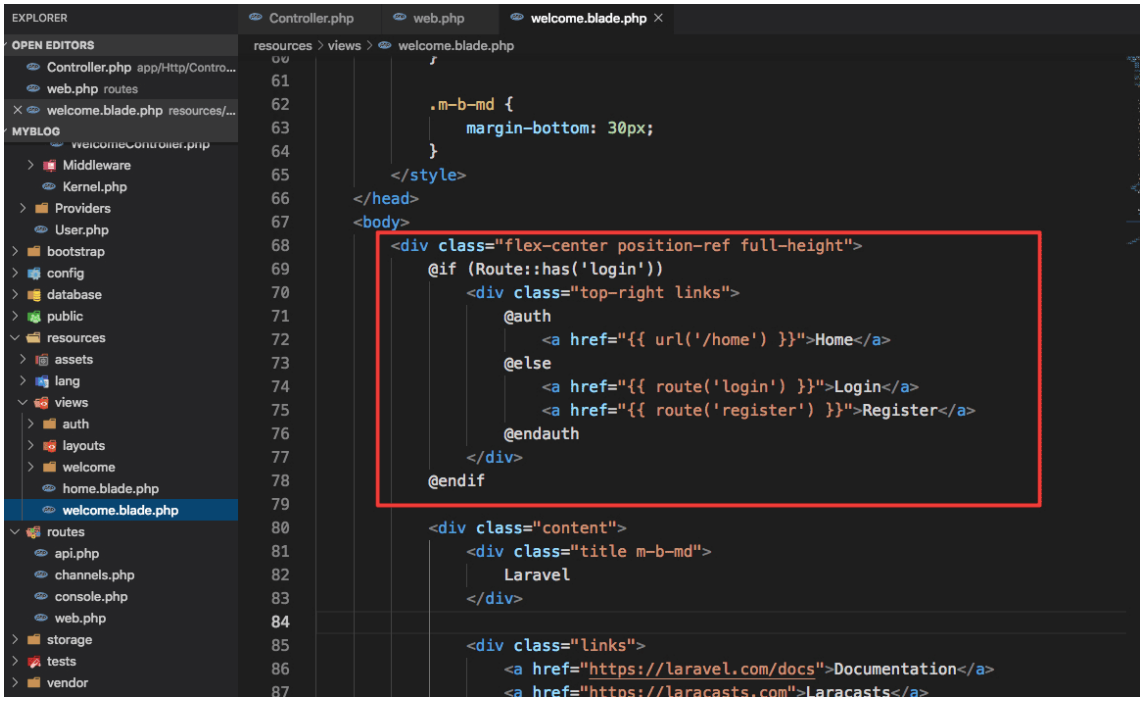


Рисунок 3.4 – Структура проекту на платформі Laravel

Flask – це мікро фреймворк написаний на мові Python. Він називається мікро, тому що не вимагає ніяких додаткових бібліотек чи програм, оскільки відсутній рівень абстракції бази даних, валідіація форм чи будь-які інші компоненти, які вимагають наявність сторонніх бібліотек, що пропонують спільні функції. Хоча, Flask підтримує розширення, що можуть бути додані як функції додатку, і виглядати як вбудовану прямо в Flask. Існують розширення для варіації форм, перетворення об’єктів, відкриті технології для автентифікації. Основною ідеєю Flask є надання початкової функціональності розробникам, на основі якої, розробник може створити необхідний йому функціонал. Приклад структури проекту зображено на рисунку 3.5.

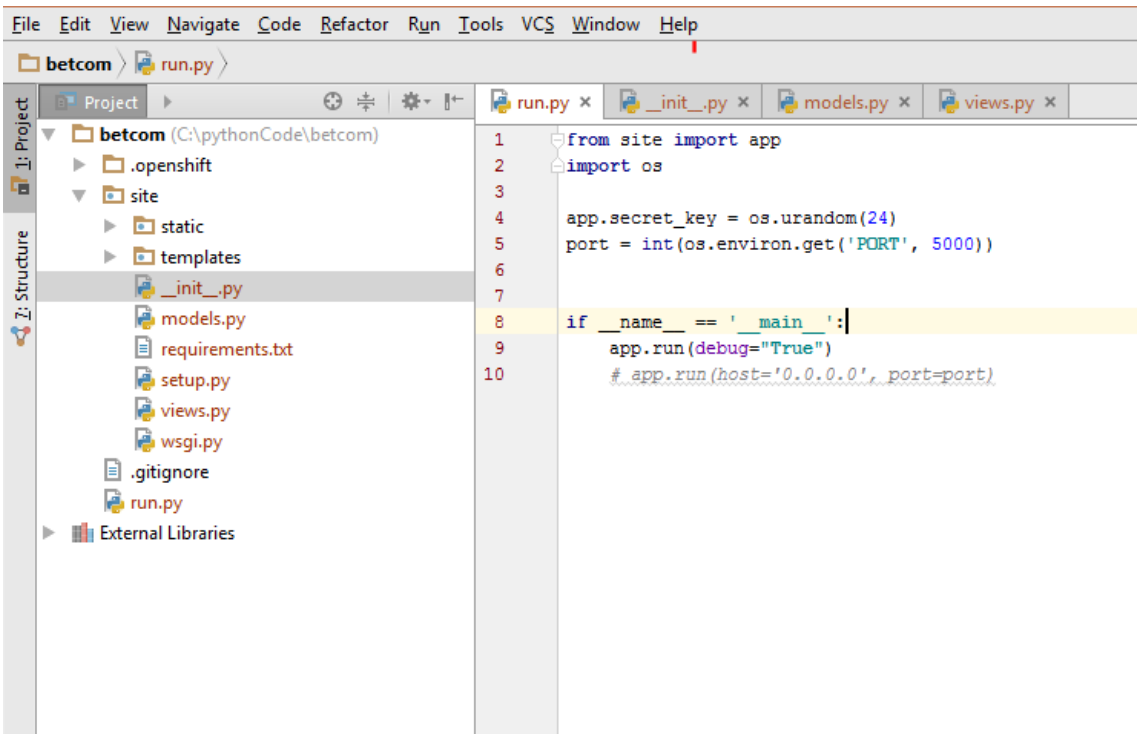


Рисунок 3.5 – Структура проекту на платформі Flask

ASP.NET Core – це безкоштовна веб платформа для побудови веб-сайтів і веб-додатків від компанії Microsoft. Розробники можуть будувати динамічні веб-додатки з використанням даного фреймоворку, які будуть слідувати принципам розділення відповідальності, швидкої розробки та TDD. Для використання платформи потрібно володіти мовою програмування C#. Платформа була побудована на основі попередника ASP.NET MVC. Дану платформу рекомендують використовувати для великих проектів, над якими працює одночасно багато розробників, і потрібний контроль над усіма змінами в проекті. Платформа постійно отримує оновлення. Структури проекту зображено на рисунку 3.6. Платформа може працювати поверх багатоплатформеного середовища .NET Core, який може бути розгорнутий на основних популярних операційних системах: Windows, Mac OS, Linux. Як результат, таким чином можливо створювати багатоплатформені додатки і за допомогою ASP.NET Core. Завдяки модульності фреймворку всі необхідні компоненти веб-додатка можуть завантажуватися, як окремі модулі через пакетний менеджер Nuget. Також, даний фреймворк характеризується

розширюваністю, оскільки він побудований з набору відносно незалежних компонент.

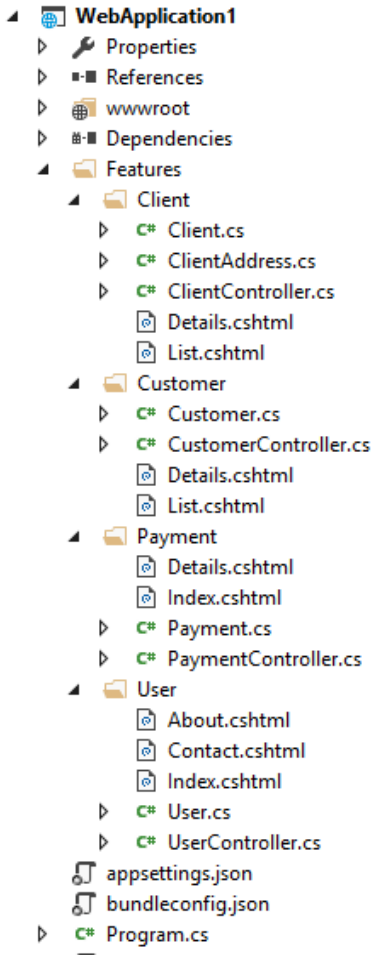


Рисунок 3.6 – Структура проекту на платформі ASP.NET Core

Для вибору платформи яку варто обрати для розроблюваної системи складено таблицю 3.2.

Таблиця 3.2 – Порівняння платформ для розробки веб-додатків

Назва	Node.js	Laravel	Flask	ASP.NET Core
Розробка компонентами	+	-	+	+
Підтримка високого	+	-	-	+

навантаження				
Підходить для великих проектів	-	-	-	+
Загалом	2	0	1	3

Для розроблювальної платформи буде доцільно обрати платформу ASP.NET Core.

Поєднання мови C# та платформи ASP.NET Core є оптимальною комбінацією для створення обраного продукту.

3.2 Аналіз вибору СУБД

Оскільки при проектуванні архітектури було прийнято рішення про необхідність використання бази даних, потрібно обрати систему управління базами даних, що буде використовуватися при роботі програми.

Оскільки, база даних є невід'ємною частиною сучасного проекту, на ринку з'явилось багато різних СУБД, варто обрати три найпоширеніших системи, а саме MySQL, PostgreSQL та Microsoft SQL Server і зробити детальне порівняння.

MySql – це система управління базами даних, що дозволяє керувати реляційними базами. Система є безкоштовною та open-source, і початково розроблена у компанії Oracle [26]. Хоча система є безкоштовно, але існує можливість купити комерційну ліцензії, що включає в себе підтримку зі сторони компанії Oracle. MySQL є досить простою в управлінні, якщо порівнювати з іншими СУБД. Також, є можливість запуску на різних платформах, таких як Windows, Unix та Linux. Крім цього, система є надійною, легко розширюваною та швидкою. Синтаксис є дещо складнішим у порівнянні з SQL Server. Для роботи з базою популярним серед розробників є середовище dbForge Studio for MySQL (рис 3.7).

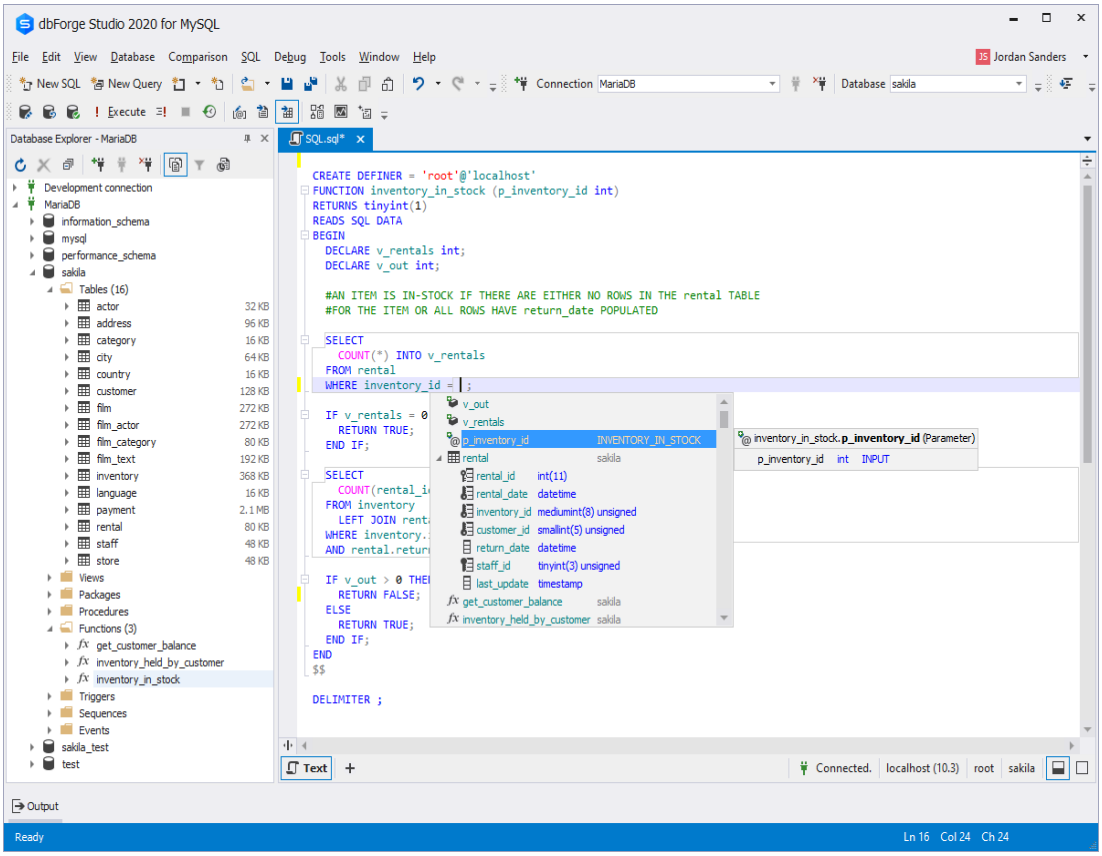


Рисунок 3.7 – Середовище dbForge Studio for MySQL

База даних підтримує багато моделей кластерних серверів, що надають їй швидкодії. Також, пропонується оптимальна швидкість, як для обробки великої аналітики, так і для роботи з електронною комерцією. Підтримується система доступу до користувацьких записів, що дає можливість гарантувати збереженість бази і високий клас її безпеки.

PostgreSQL – проект розроблений волонтерами зі всього світу, він не контролюється ніякою організацією і його вихідний код є доступний кожному, без будь-якої плати. Система підтримує сучасну функціональність, таку як ACID принципи, індекси, роботу з типами даних, що описані в SQL. Для роботи з базою використовують dbForge Studio for PostgreSQL (рис 3.8). PostgreSQL підтримується на всіх сучасних Unix системах, включаючи найбільш поширені такі як Linux, FreeBSD, NetBSD. Також, вона визнавалась базою року декілька разів. База повністю підтримує принципи ACID,

багатоверсійності, реплікації та цілісності даних. Доступ із додатків до даних бази PostgreSQL здійснюється за допомогою спеціального процеса бази даних. Тому, клієнтські програми не можуть отримувати самостійний доступ до даних навіть в тому випадку, якщо вони функціонують на одному й тому ж комп'ютері, на якому відбувається серверний процес.

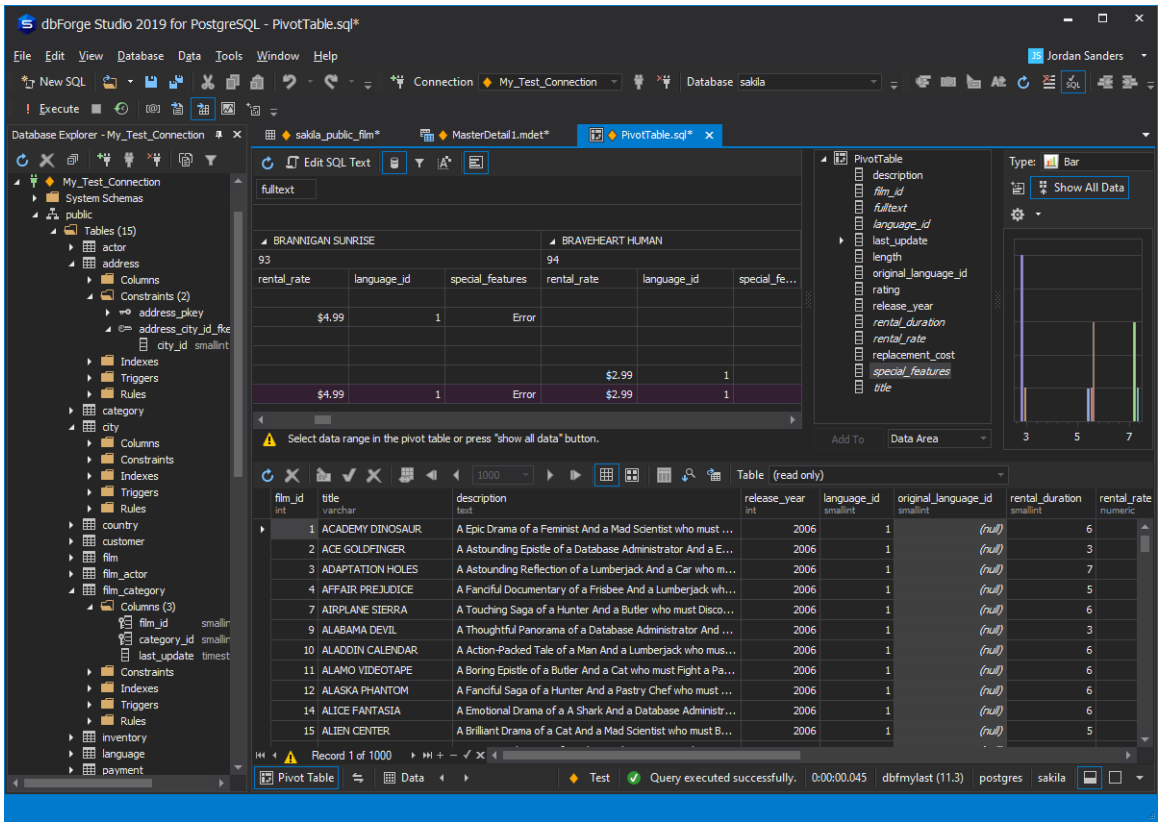


Рисунок 3.8 – Середовище dbForge Studio for PostgreSQL

Microsoft SQL Server – СУБД, розроблена по специфікації для реляційних систем керування базами даних, компанією Microsoft. Вона є залежна під платформу, а саме під запуск на Windows машинах. Працює, як клієнт-серверна архітектура, тому підтримує два типи компонент, сервер та робочі станції, які виступають інтерфейсом для керування з сервером.

SQL Server Management Studio є середовищем для роботою з даним типом баз (рис. 3.9). Це є потужне середовище для управління базами даних. Є можливість налаштування, спостереження та адміністрування різних SQL серверів та баз даних [28]. За допомогою даного середовища можливо

розгортати, спостерігати та обновляти компоненти рівня даних, що використовуються додатки. Також є можливість створювати запити та скрипти. Воно також підтримує роботу з хмарними базами даними, такими як SQL Azure. Є можливість створення нових серверів з базами даних, їх запуск та зупинка.

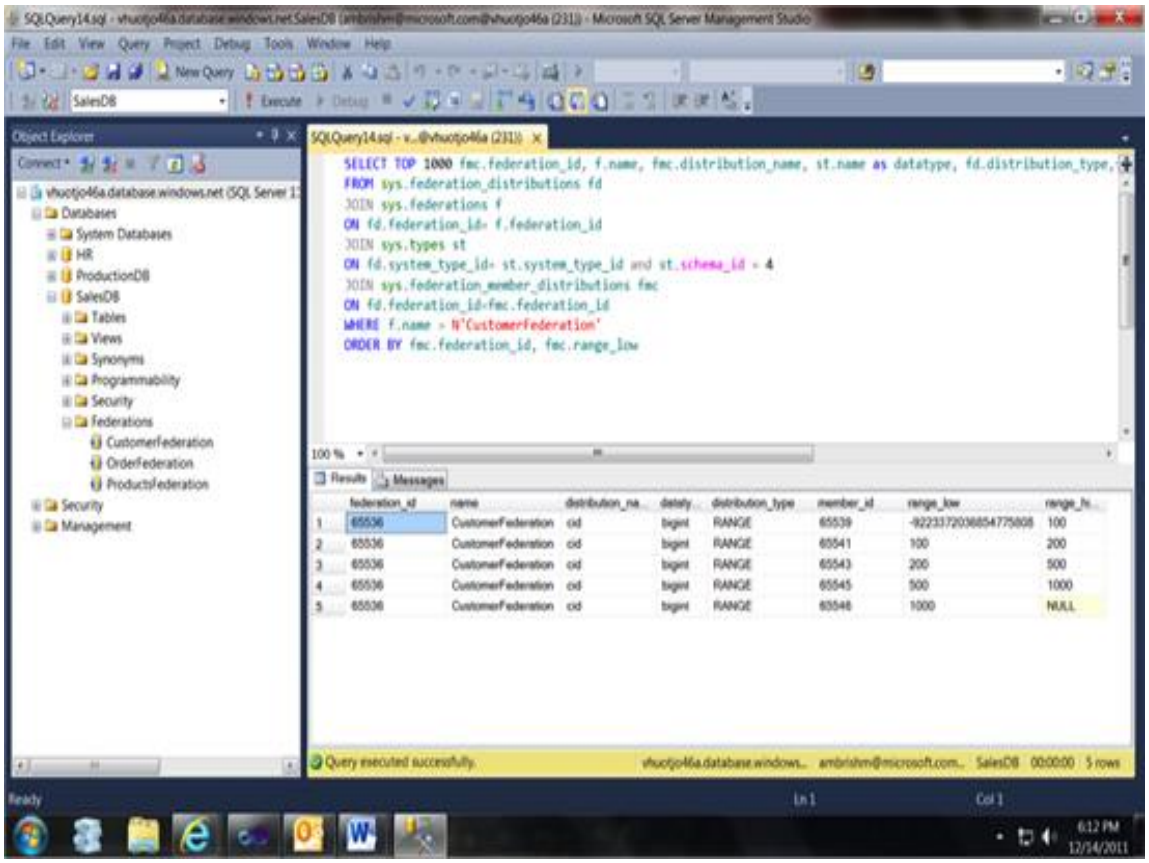


Рисунок 3.9 – Середовище SQL Server Management Studio

Детальне порівняння систем управління базами даних наведено у таблиці 3.3.

Таблиця 3.3 – Порівняння СУБД

Назва	PostgreSQL	MySQL	SQL Server
Відсутня чутливість до регістру	-	+	+

Підтримка тимчасових таблиць	+	-	+
Можливість створення функціональних обмежень	+	-	+
Загалом	2	1	3

На основі складеної таблиці, оптимальною СУБД для розроблюваного продукту буде Microsoft SQL Server.

3.3 Програмна реалізація модулів системи для проведення A/B тестування

У роботі реалізовується система для проведення A/B тестування. Під час виконання було розроблено модулі для авторизації та реєстрації, створення нового експерименту, керування умовами при яких повинен виконуватися експеримент, розподілу трафіку між варіаціями, створення та керування варіаціями експерименту, формування виконуваного коду, який буде добавлятися на сайт на якому проводяться експерименти. Варто розглянуто основні модулі платформи.

Модуль створення нового експерименту є головним модулем при початку роботи з платформою, він відповідає за внесення початкових даних про експеримент у базу даних. Користувач може добавити інформацію про назву експерименту, опис та умови його виконання на сторінці (рис 3.10).

```
export const CreateExperiment: React.FC = () => {
  let history = useHistory();

  async function createExperiment() {
    try {
      const data = { name: name, description: description, assignment: assignment};
      const response = await axios.post(BACK_END_HOST_LOCATION + '/api/experiment', data, axiosConfig);
      history.push(`/experiment/${response.data}`);
      history.go(0)
    } catch (error) {
      alert(error);
    }
  }

  const [name, setName] = React.useState('');

  const handleNameChange = (event) => {
    setName(event.target.value);
  };

  const [description, setDescription] = React.useState('');

  const handleDescriptionChange = (event) => {
    setDescription(event.target.value);
  };

  const [assignment, setAssignment] = React.useState('');

  const handleAssignmentChange = (event) => {
    setAssignment(event.target.value);
  };

  const onCreateButtonClick = () => {
    createExperiment();
  }

 return (
  <Fragment>
    <div className="createExperiment_container">
    <TextField id="outlined-basic" label="Назва" variant="outlined"  sx={{marginBottom: "20px", width: "65%"}}
               value={name}
               onChange={handleNameChange} />
    <TextField
        id="outlined-multiline-static"
        label="Опис"
        multiline
        rows={4}
        sx={{marginBottom: "20px", width: "65%"}}
        value={description}
        onChange={handleDescriptionChange}
      />
          <TextField
        id="outlined-multiline-static"
        label="Умова попадання на експеримент (JavaScript)"
        multiline
```

Рисунок 3.10 –Модуль створення нового експерименту

В результаті створення експерименту, відбувається внесення в базу всіх необхідних даних і користувачу повертається на зовні ID експерименту, який було створено. Після успішного створення користувач потрапляє на сторінку керування щойно створеним експериментом. Модуль управління експериментом підтримує можливість змінення даних які були внесені при створенні експерименти, а також зміна розподілу трафіку, запуск та зупинення експерименту (рис 3.11).

Рисунок 3.11 –Модуль управління експериментом

Модуль управління експериментами також взаємодіє з модулем управління варіацією, адже всі варіації відносяться до певних експериментів. Даний модуль підтримує повне керування варіаціями обраного експерименту. Є можливість добавлення нової варіації для експеримента, видалення, зміна назви варіації та самого коду який виконується на сторінці при потраплянні користувача на дану варіацію (рис 3.12). Керування модулем доступне лише для зареєстрованих користувачів. Інформація про користувача, що робив зміни за допомогою цього модуля теж зберігається, для можливості подальшого перегляду

```
async function deleteVariation(event, variationId1){
 try {
  const response = await axios.delete(BACK_END_HOST_LOCATION + `/api/variation/${variationId1}`, axiosConfig);
  history.go(0)
 } catch (error) {
  alert(error);
 }
}

const [name, setName] = React.useState('');

const handleNameChange = (event) => {
  setName(event.target.value);
};
const handleVariationCodeChange = (event) => {
 setVariationCode(event.target.value);
};
const handleVariationNameChange = (event) => {
 setVariationName(event.target.value);
};

const [description, setDescription] = React.useState('');

const handleDescriptionChange = (event) => {
  setDescription(event.target.value);
};

const [assignment, setAssignment] = React.useState('');

const handleAssignmentChange = (event) => {
  setAssignment(event.target.value);
};
```

Рисунок 3.12 –Модуль управління варіацією

За перегляд всіх експериментів відповідає модуль перегляду експериментів (рис 3.13).

За допомогою модуля можливо:

- переглянути список всіх створених експериментів, їх назву та унікальний номер;

- відфільтрувати лише ті експерименти, які в даний момент запущені;

- можливість перейти на панель управління обраного експерименту;

- здійснити пошук по назві експеримента або по його унікальному ідентифікатору.

Даний модуль виступає також головною сторінкою, коли користувач входить у систему він попадає спершу на результат виконання даного модуля.

Модуль також підтримує пагінації, оскільки в випадку, коли створено багато експериментів, правильним рішенням є відображення їх окремими

сторінками, для зручності користування та швидкості завантаження сторінки, оскільки сервер повертає дані лише для обраної сторінки.

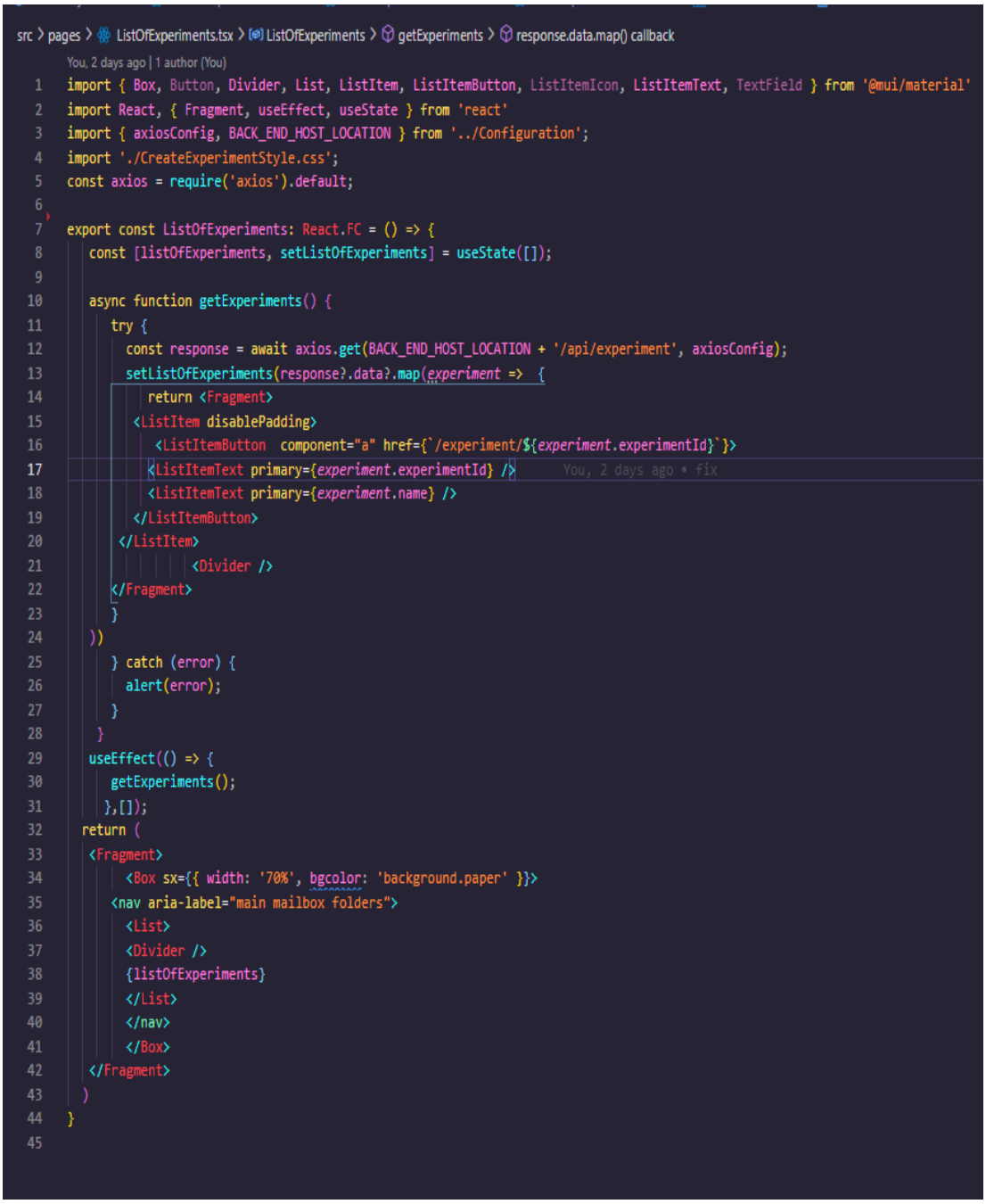


Рисунок 3.13 – Модуль перегляду експериментів

Для розробки платформи, було створено також додаткові модулі та компоненти, що є необхідні для коректної роботи додатку. Лістинги модулів наведено в додатку В.

3.4 Висновки

У третьому розділі було проведено аналіз основних мов програмування для розробки платформи для проведення A/B тестування, в результаті чого було обрано мову програмування C# для розробки серверної частини додатку. Також було обрано використовувати платформу для розробки ASP.NET Core, це є сучасна платформа для розробки веб-додатків. Оскільки є необхідність зберігати дані у базі даних, було прийнято рішення зробити порівняння сучасних СУБД, а саме MySQL, PostgreSQL та SQL Server, в результаті чого було обрано останню, як найоптимальнішу систему для системи. Також було розглянуто 4 основних модулі платформи, модуль для перегляду експериментів, управління експериментами та варіаціями, створення нового експерименту.

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Аналіз методів і засобів тестування

Тестування програмних продуктів – це метод перевірки функціональності додатку, з метою переконатися що розробка відповідає всім поставленим задачам і переконатися, що у відсутності дефектів [29]. Це включає в себе виконання певних програмних або системних компонент з використанням автоматизованого або ручного тестування для перевірки однієї або декількох функцій системи. Основною цілю тестування є ідентифікація помилок, не відповідності поставленим вимогам, пошук потенційних проблем у системі.

Одним з видів тестування є автоматичне, воно включає в себе автоматизовану перевірку системи за допомогою розроблених систем для тестування.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;

namespace seleniumDemo
{
    [TestClass]
    public class UnitTest1
    {
        static IWebDriver driverFF;
        static IWebDriver driverGC;

        [AssemblyInitialize]
        public static void SetUp(TestContext context)
        {
            driverFF = new FirefoxDriver();
            driverGC = new ChromeDriver(@"C:\chromedriver_win32");
        }

        [TestMethod]
        public void TestFirefoxDriver()
        {
            driverFF.Navigate().GoToUrl("http://www.google.com");
            driverFF.FindElement(By.Id("lst-ib")).SendKeys("Selenium");
            driverFF.FindElement(By.Id("lst-ib")).SendKeys(Keys.Enter);
        }

        [TestMethod]
        public void TestChromeDriver()
        {
            driverGC.Navigate().GoToUrl("http://www.google.com");
            driverGC.FindElement(By.Id("lst-ib")).SendKeys("Selenium");
            driverGC.FindElement(By.Id("lst-ib")).SendKeys(Keys.Enter);
        }
```

Рисунок 4.1 – Приклад Selenium тесту

Одним з прикладів системи для автоматизованого тестування є Selenium Framework, що підтримує розробку мовою C# (рис 4.1). Така зручність, дозволяє розробникам займатися написанням автоматизованих тестів без будь-яких додаткових навичок.

Автоматичне тестування включає в себе всі рівні тестів, а саме unit, компонентне, інтеграційне, API та функціональне тестування. Ці рівні тестування можна відобразити у вигляді піраміди (рис 4.2).

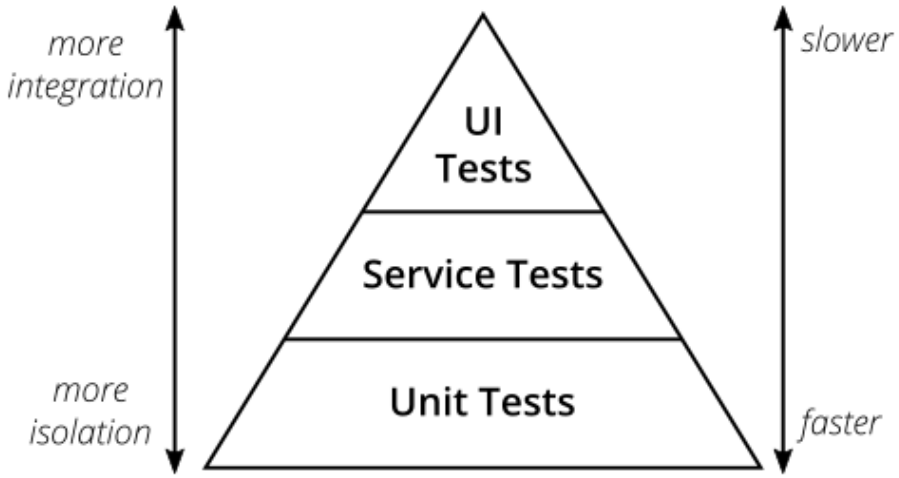


Рисунок 4.2 – Піраміда тестування

У піраміді тестування чим нижче знаходиться певний рівень тестування тим скоріше тести виконуються і ціна виконання є дешевшою. Швидкість виконання є важливою, на найнижчому рівні знаходяться unit тести, у такому виді тестів розробник може отримувати результати безпосередньо під час розробки певної функціональності на власному комп’ютері. Також, дана перевага дає можливість покрити велику кількість функціональності, а її перевірка даним видом тестів буде займати незначний період часу.

Виконання тестів на найвищих рівнях піраміди вимагають багато часу, та розгортання додатку, що також займає певний проміжок часу. Дані тести важливі тим, що дають змогу перевірити всю систему в цілому. Досить часто складні додатки складаються з багатьох окремих сервісів, перевірка їх

взаємодії між собою можлива лише на даному рівні тестування. Оскільки даний вид тестування займає багато часу, рекомендовано мати досить обмежену кількість саме таких тестів та покривати ними лише взаємодії між всіма модулями програми.

Методи тестування також поділяються на статичні та динамічні [30]. У випадку статичного методу відбувається перевірка продукту до відповідності поставленим вимогам. Динамічне тестування має на меті перевірку продукту, а саме огляд чи розроблений додаток слідує всім сучасним стандартам, наприклад підтримка захищеного протоколу HTTPS для з'єднання з серверною частиною.

Існує три підходи до тестування програмного забезпечення: сіра скринька, біла скринька та чорна скринька.

Метод тестування чорної скриньки – це потужна техніка перевірки додатку зі сторони користувача. Він використовується для перевірки стійкості системи на вплив зовнішніх факторів, що зазвичай викликають поломки програмного забезпечення. Також цей метод називають поведінковим тестуванням.

Метод білої скриньки вимагає від тестувальника знання, як працює система. Людина, що проводить тестування повина також володіти знаннями безпечного програмування, для того, щоб мати змогу ідентифікувати потенційні проблеми, якими можуть скористатися хакери, наприклад вставити певний кусок коду, що буде повертати дані з бази даних на зовні.

Тестування методом сірою скриньки поєднує в собі метод білої та чорної скриньки. Тестувальник повинен мати досить обмежені знання про внутрішню будову системи. Цей метод є досить корисним при тестуванні веб-додатків, де будь-який користувач може ззовні передати вхідні дані і система має перевірити їх коректність та тільки потім опрацювати.

Оскільки було розроблено вимоги до функціональності платформи для проведення A/B тестування, тестування методом чорної скриньки буде найефективнішим.

4.2 Автоматизоване тестування розробленого додатку

Для проведення автоматизованого тестування будуть використовуватися unit тести, оскільки вони є стабільні та не вимагають багато часу для виконання. Такі тести можливо створювати у середовищі у якому безпосередньо відбувається основна розробка. Також за допомогою додаткових розширень можливо налаштувати автоматичне виконання тестів під час модифікації коду [31]. Така можливість надає можливість розробнику одразу розуміти, що він зробив помилку в випадку, якщо тест впав. Також зручно користуватися такою функціональністю, якщо виконується розробка з використанням підходу TDD, що має на меті спочатку написання тестів, а потім коду.

На сьогоднішній день існує велика кількість різних бібілотек для проведення unit тестування з використанням мови C#. Для вибору найоптимальнішого варто обрати три найпопулярніших, що існують на ринку та здійснити їх. Порівняння фреймворків для unit тестування описано в таблиці 4.1.

Таблиця 4.1 – Порівняльна характеристика платформ для unit тестування

Назва	NUnit	XUnit	MSTest
Індикація про наявність групи тестів у класі	1	1	1
Індикація, що метод повинен бути викликаний після перед виконанням будь-якого тесту	1	1	0
Можливість отримати доступ до тестового контексту	1	0	1
Вбудована функція Fluent Assertiation	1	0	0
Загалом	4	2	2

Проаналізувавши результати порівняння платформ для тестування, можна зробити висновки, що NUnit є лідером, оскільки має підтримку додаткової функціональності, що є відсутня в інших платформа, тому для проведення тестування оптимальним вибором буде саме дана бібліотека.

Даним типом тестування можливо покрити кожен метод. Приклад unit тесту, що перевіряє метод, який знаходить в списку експериментів по ID експерименту всю його інформацію, наведено на рисунку 4.3.

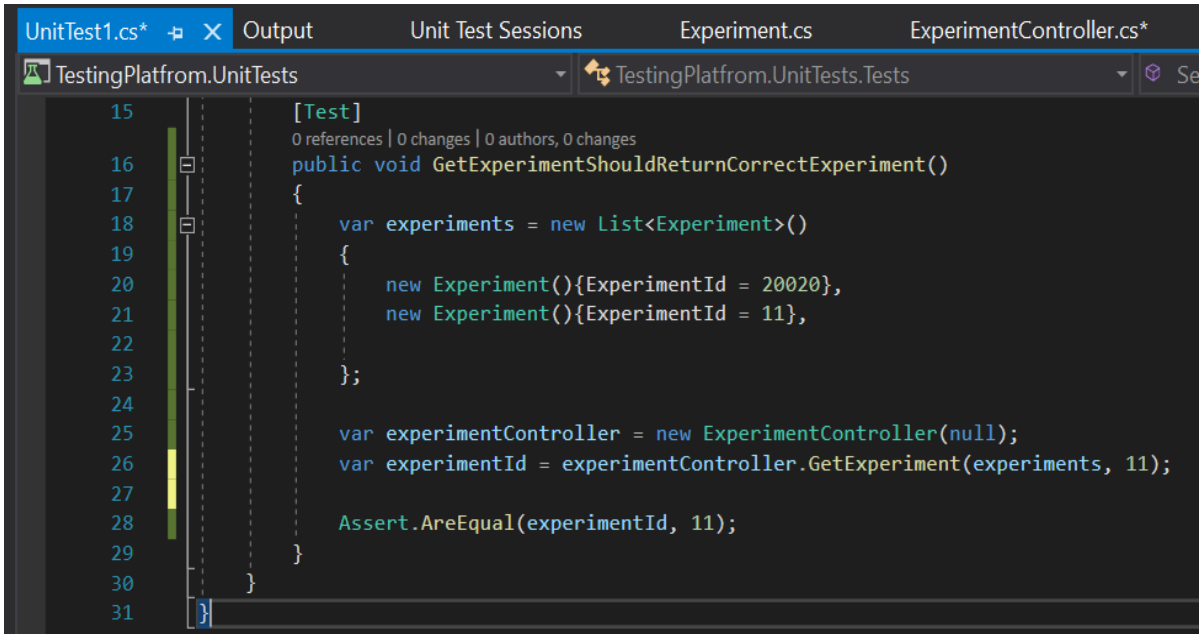


Рисунок 4.3 – Unit тест для перевірки методу знаходження експерименту

Якщо певний метод має якісь залежності на інші стороні класи, є можливість створити mock класу залежності, така функціональність надає можливість самостійно перевіряти з якими параметрами була викликана залежність, та вказати, що вона повинна повернути. Також, це піднімає рівень покриття коду самим тестами, оскільки є можливість перевірити методи, що можуть робити виклики до сторонніх ресурсів. Прикладом тесту з використанням mock об'єкту може бути тест, що створює новий експеримент у базі даних, та повертає назовні ID новоствореного

експерименту (рис 4.4). У даному випадку запис в базу та повернення ID буде відбуватися за допомогою mock об’єкта.

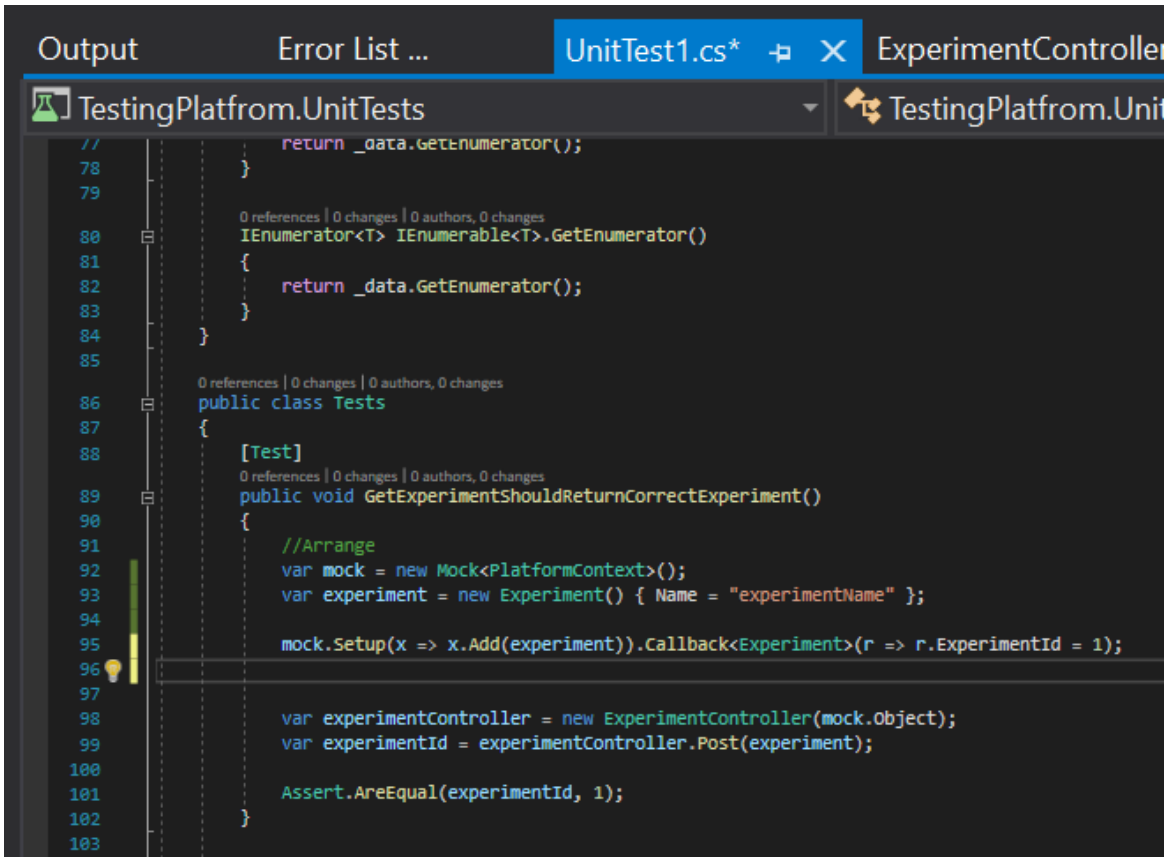


Рисунок 4.4 – Unit тест для перевірки методу створення нового експерименту

4.3 Ручне тестування платформи для A/B експериментів

Тестування методом чорної скриньки вимагає розробки сценаріїв, які будуть перевіряти функціональність додатку. Тестування буде виконуватися у браузері Google Chrome версії 95.

Тест-кейс №1 – Перевірка можливості створити новий експеримент:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Натиснути кнопку «Створити новий».
4. Ввести назву, опис, та умови потрапляння.
5. Натиснути кнопку «Створити».

Очікуваний результат: переадресація на сторінку вікна зі створеним експериментом та внесеними полями.

Результат виконання відповідає очікуваному результату (рис. 4.5).

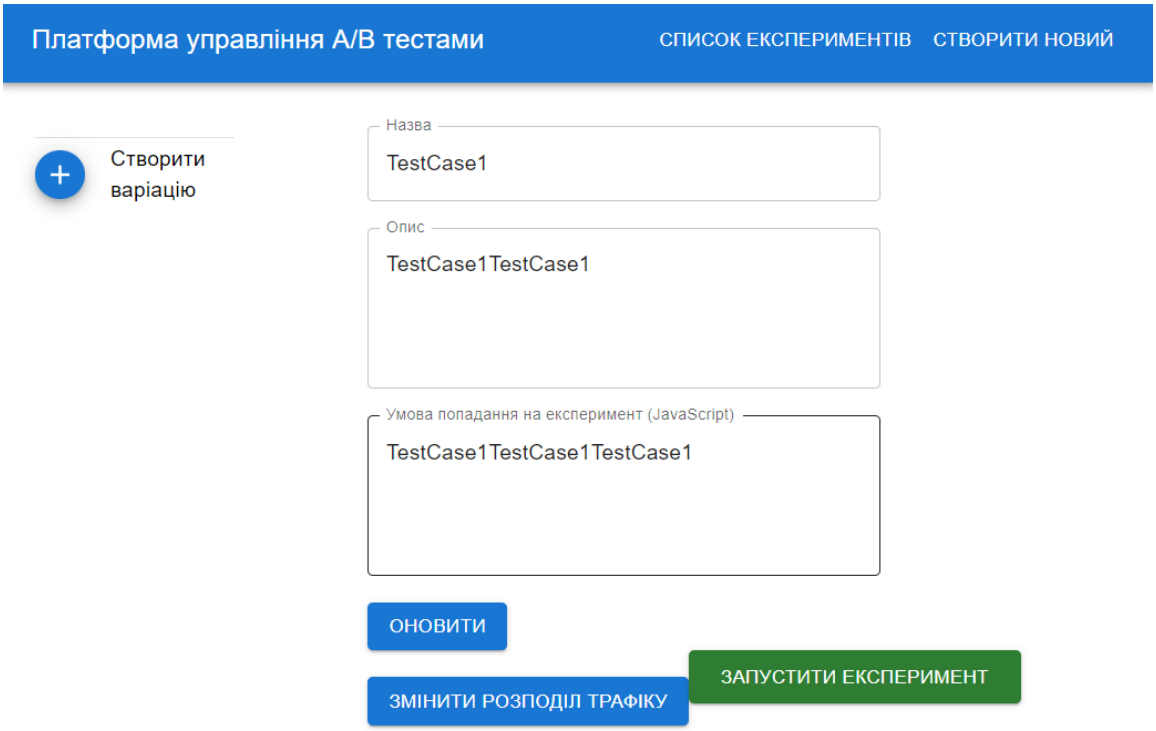


Рисунок 4.5 – Результат виконання тест-кейсу №1

Тест-кейс №2 – Перевірка можливості редагування експерименту:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Відкрити новостворений експеримент.
4. Натиснути змінити назву експерименту та опис.
5. Натиснути кнопку «Оновити».
6. Зробити перезавантаження сторінки.

Очікуваний результат: відкриття сторінки з щойно оновленими даними експеримента.

Результат виконання відповідає очікуваному результату (рис. 4.6).

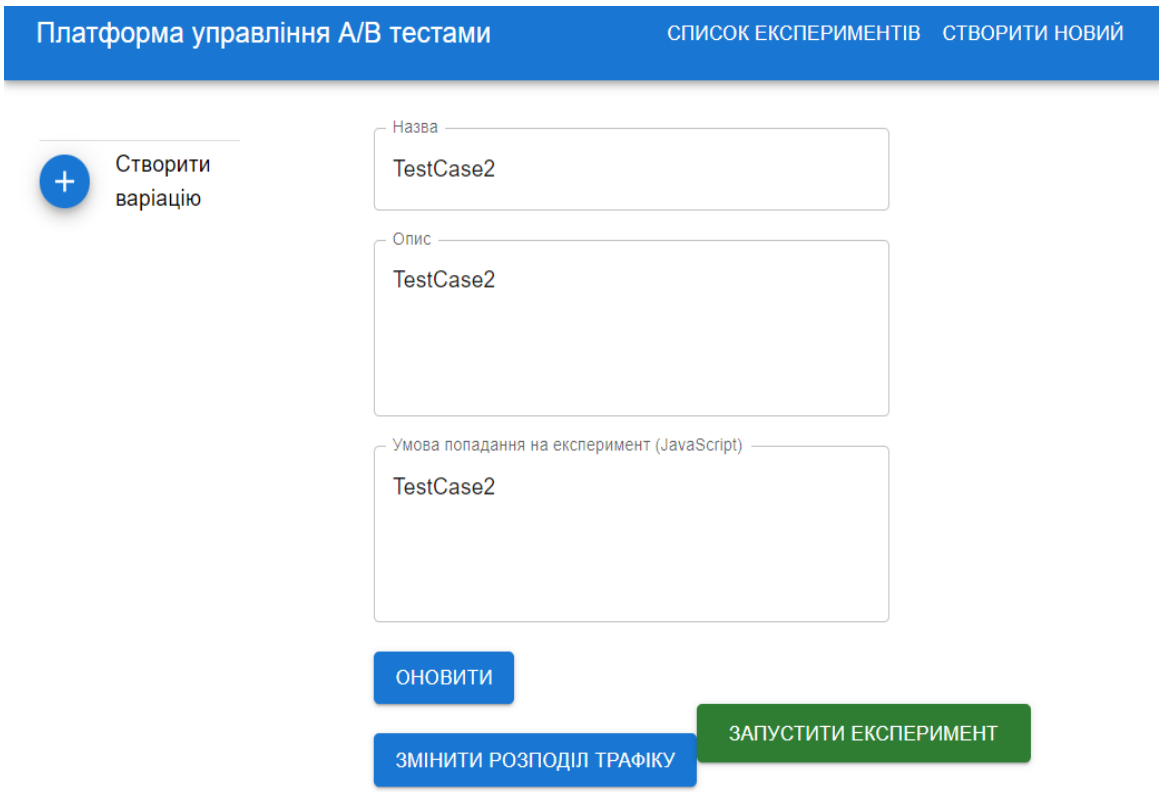


Рисунок 4.6 – Результат виконання тест-кейсу №2

Тест-кейс №3 – Перевірка можливості створення нової варіації з внесенням інформації:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Відкрити список всіх експериментів.
4. Відкрити новостворений експеримент.
5. Натиснути кнопку «Створити варіацію».
6. У відкритому діалоговому вікні ввести назву експерименту та код.
7. Натиснути кнопку «Оновити».
8. У списку варіацій натиснути по новоствореній варіації.

Очікуваний результат: відкриття діалогового вікна з інформацією про варіацію, що була попередньо створена, та назва і код виконання співпадає з попередньо введеними даними.

Результат виконання відповідає очікуваному результату (рис. 4.7).

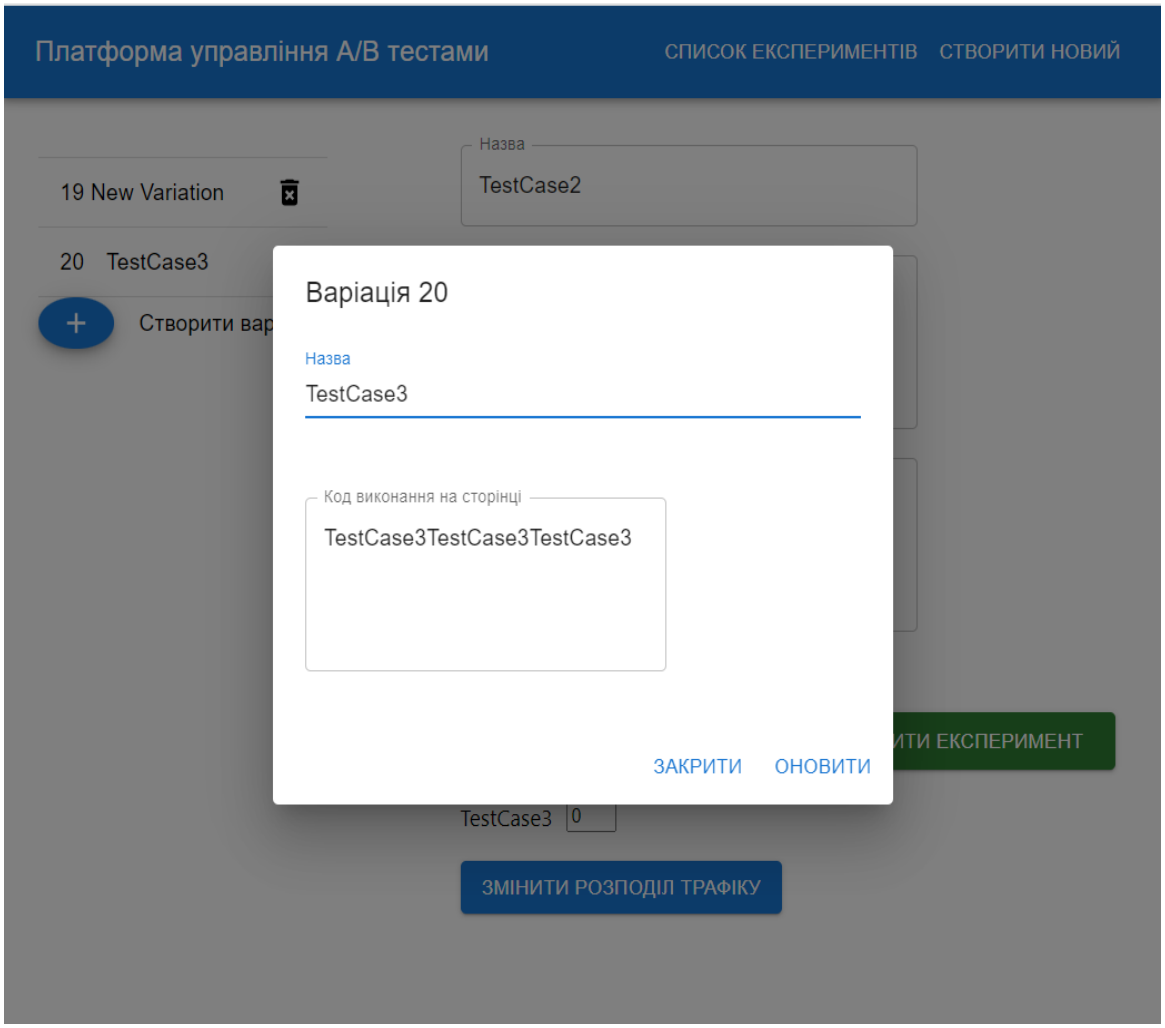


Рисунок 4.7 – Результат виконання тест-кейсу №3

Тест-кейс №4 – Перевірка можливості зміни розподілу трафіку:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Відкрити новостворений експеримент з варіаціям.
4. У полях з розподілом трафіку ввести цифри.
5. Натиснути кнопку «Змінити розподіл трафіку».
6. Перезавантажити сторінку.

Очікуваний результат: відкриття сторінки експерименту з щойно оновленим розподілом трафіку.

Результат виконання відповідає очікуваному результату (рис. 4.8).

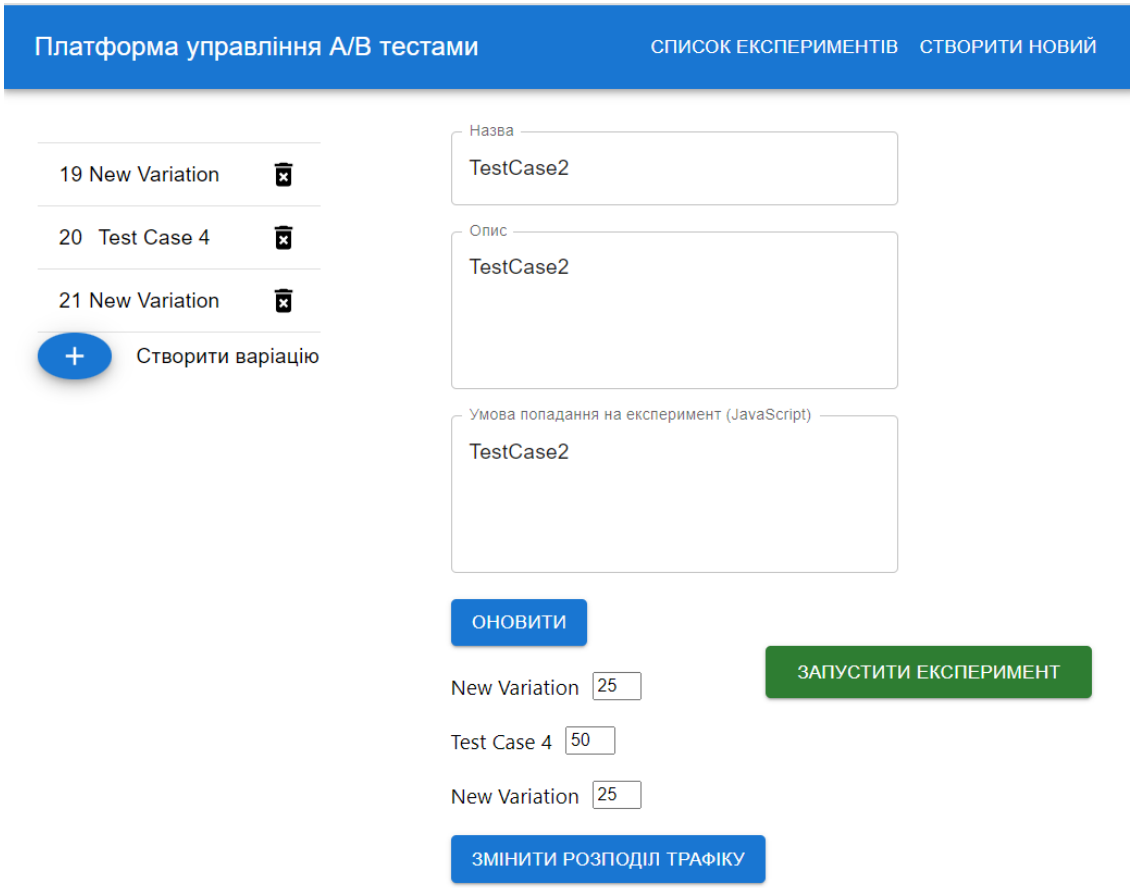


Рисунок 4.8 – Результат виконання тест-кейсу №8

Тест-кейс №5 – Перевірка валідації розподілу трафіку:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Відкрити новостворений експеримент з варіаціям.
4. Перевірити, що список варіацій залишився незмінним.
5. У полях з розподілом трафіку ввести цифри, сума яких не рівна 100.
6. Натиснути кнопку «Змінити розподіл трафіку».

Очікуваний результат: відкриття діалогового вікна з інформацією, що повідомляє, що сума трафіку на всіх створених варіаціях не є рівною 100% відсоткам.

Результат виконання відповідає очікуваному результату (рис. 4.9).

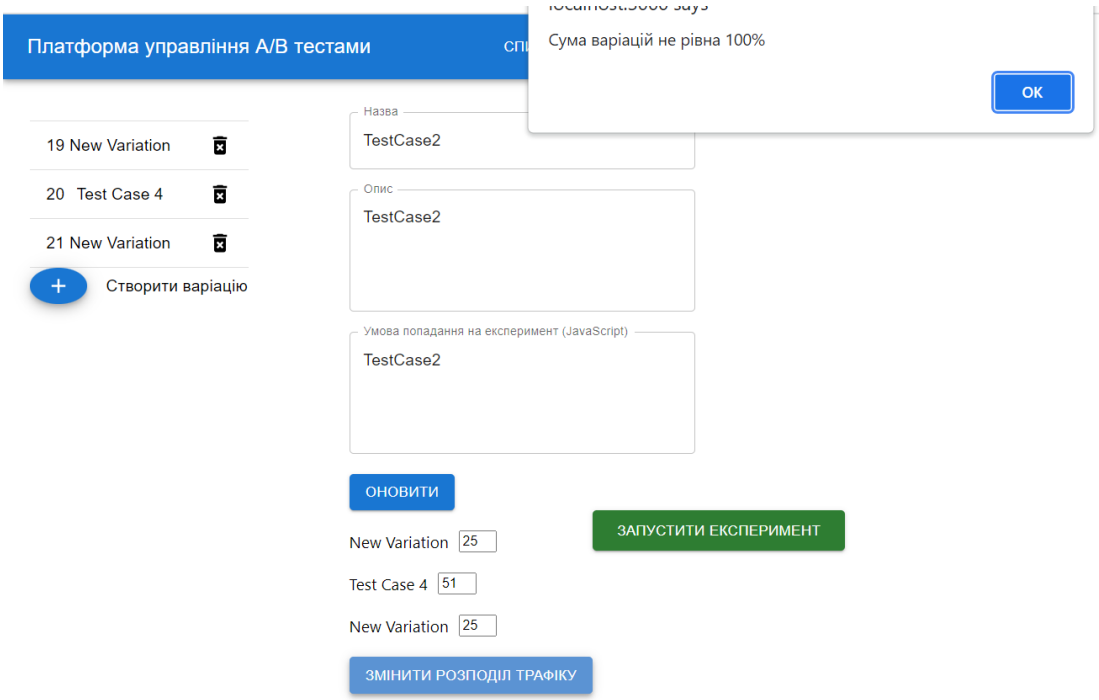


Рисунок 4.9 – Результат виконання тест-кейсу №5

Тест-кейс №6 – Перевірка можливості запуску та зупинки експерименту:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Натиснути кнопку «Створити новий».
4. Ввести назву, опис, та умови потрапляння на експеримент.
5. Натиснути кнопку «Створити».
6. Відкрити новостворений експеримент.
7. У відкритому діалоговому вікні ввести назву експерименту та код.
8. Натиснути кнопку «Оновити».
9. Натиснути кнопку «Запустити експеримент» або «Зупинити експеримент».
10. Перезавантажити сторінку.

Очікуваний результат: експеримент запущено або зупинено, кнопка змінює колір на червоний, якщо експеримент запущений, та зелений у випадку, якщо експеримент зупинений та готовий до запуску, та напис на кнопці відповідає очікуваному стану.

Результат виконання відповідає очікуваному результату (рис. 4.10).

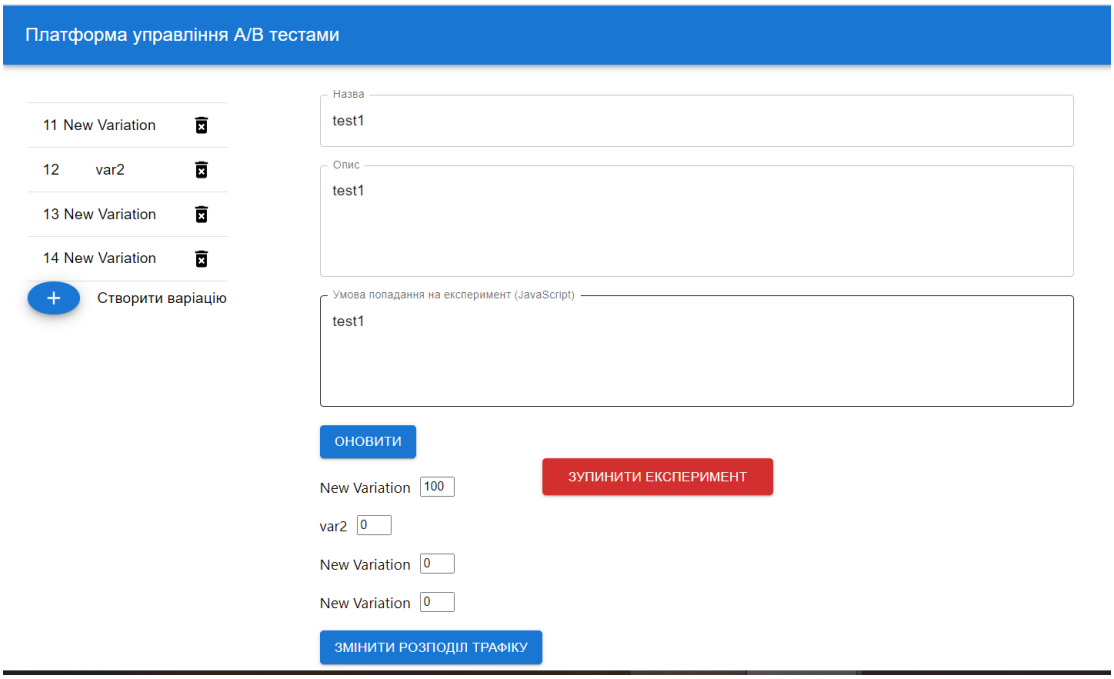


Рисунок 4.10 – Результат виконання тест-кейсу №6

Тест-кейс №7 – Перевірка можливості зміни розподілу трафіку:

1. Відкрити нову вкладку браузера.
2. Відкрити веб-сайт з платформою для керування експериментами.
3. Натиснути кнопку «Створити новий».
4. Ввести назву, опис, та умови потрапляння на експеримент.
5. Натиснути кнопку «Створити».
6. Відкрити новостворений експеримент.
7. Створити дві нових варіації.
8. Змінити розподіл трафіку для створених варіацій, щоб сума була рівна 100 відсотків.
9. Натиснути кнопку «Змінити розподіл трафіку».
10. Перезавантажити сторінку.

Очікуваний результат: відображений розподіл трафіку відповідає, щойно оновленим цифрам та у списку варіації знаходяться всі варіації, що були створені під час виконання тесту.

Результат виконання відповідає очікуваному результату (рис. 4.11).

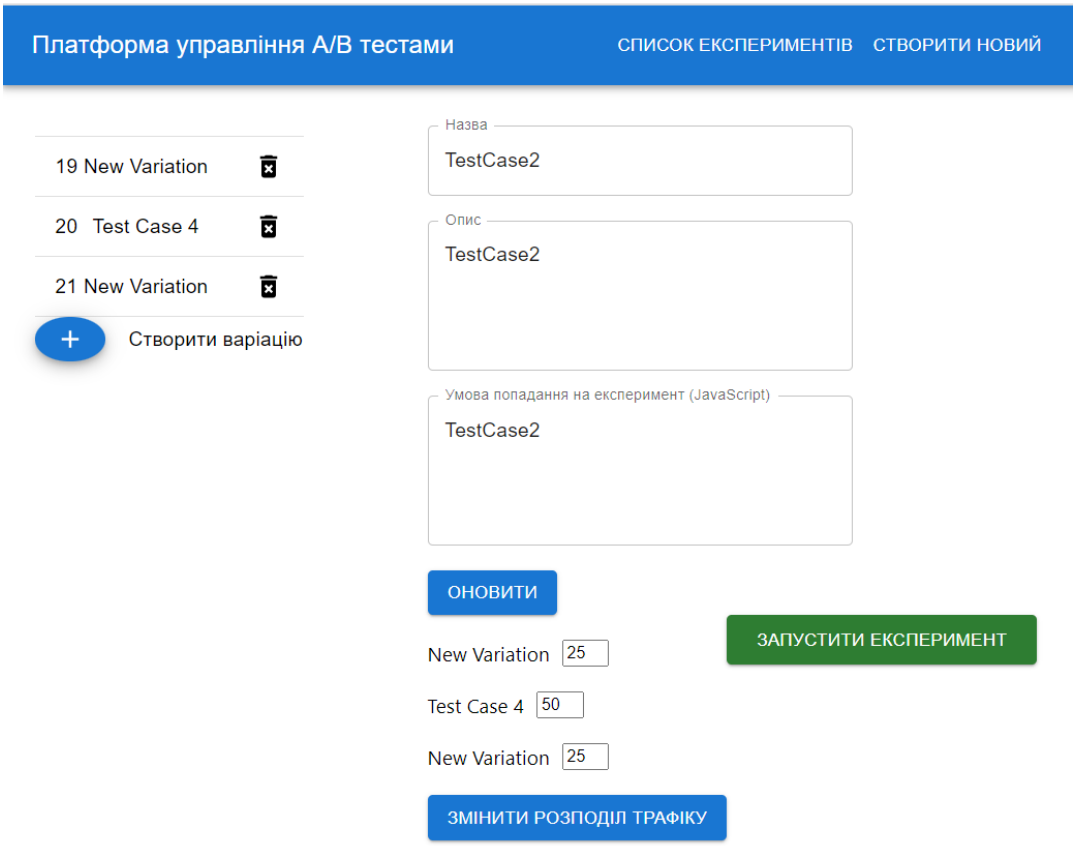


Рисунок 4.11 – Результат виконання тест-кейсу №7

Отже, отримані результати свідчать про виконання всіх визначених функцій в програмному продукті.

4.4 Розгортання та впровадження системи

Розгортання програмного забезпечення – це процес, що має на меті зробити систему готовою до використання кінцевими користувачами. Розгортання повинне відбуватися лише після повного тестування, для того, щоб переконатися, що всі варіанти використання і проблеми були ідентифіковані та виправлені. Зазвичай процес розгортання залежить від типу додатку, який планується до розгортання, оскільки необхідні дії можуть відрізнятися.

Розроблений додаток містить три окремих компоненти, базу даних, серверну та клієнтську частини. Для розгортання таких компонент можливо

обрати будь-яку хмарну платформу, для даної розробки було обрано Azure. Дана платформа належить компанії Microsoft. Вона надає можливість використовувати хмарні сервіси, включаючи сервіси для обрахування, аналітики, зберігання та управління мережею (рис. 4.12).

Рисунок 4.12 – Azure сервіси

Для зберігання коду було обрано платформу GitHub, що підтримує систему контролю версій Git. Дана платформа підтримує можливість створення безкоштовно публічних репозиторіїв та має добру взаємодії з Azure.

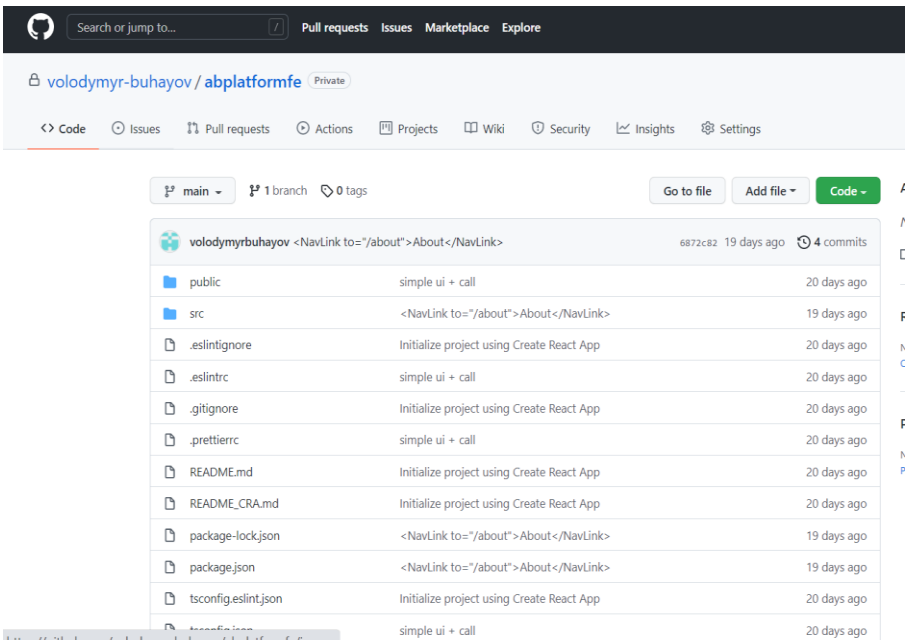


Рисунок 4.13 – GitHub репозиторій для клієнтської частини

Під час зберігання коду в публічних репозиторіях важливо не допустити завантаження секретної інформації, що міститься в конфігураційних файлах. Такі дані повинні замінятися в даних файлах у момент розгортання додатку.

Було створено два окремих GitHub репозиторії, для серверної та клієнтської частини (рис 4.13). Вони дають змогу мати доступ до написаного коду з будь-якого комп’ютера, це захищає код від можливої втрати.

Після того, як код додатку був завантажений на сервери, потрібно розгорнути базу даних. Оскільки у програмі використовувалася реляційна база даних SQL, необхідно створити на Azure порталі також реляційну базу даних, а саме SQL database. Для створення було обрано мінімальні потужності, оскільки поки відсутнє навантаження на систему в цілому (рис 4.14). Базу даних для додатку було названо testingplatform.

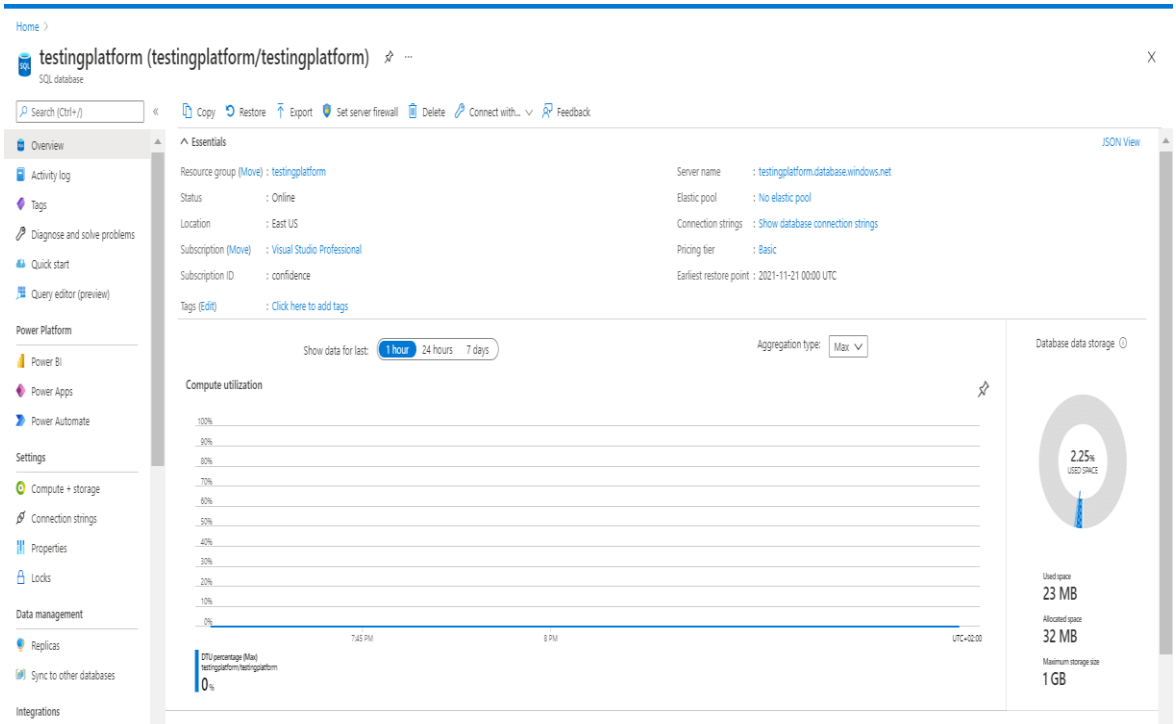


Рисунок 4.14 – Створена база даних в Azure

Наступним кроком після створення бази даних є розгортання серверної

частини та підключення її до новоствореної бази даних. Для створення серверної частини було обрано App Service на Azure порталі. Під час налаштування вказано необхідну систему для запуску додатку, а саме .NET Core 3.1 (рис 4.15). Після створення необхідно з'єднати систему з GitHub репозиторієм та зробити перше автоматичне розгортання додатку. Після успішного розгортання можливо отримати дані, що збережені у базі через API розробленого додатку.

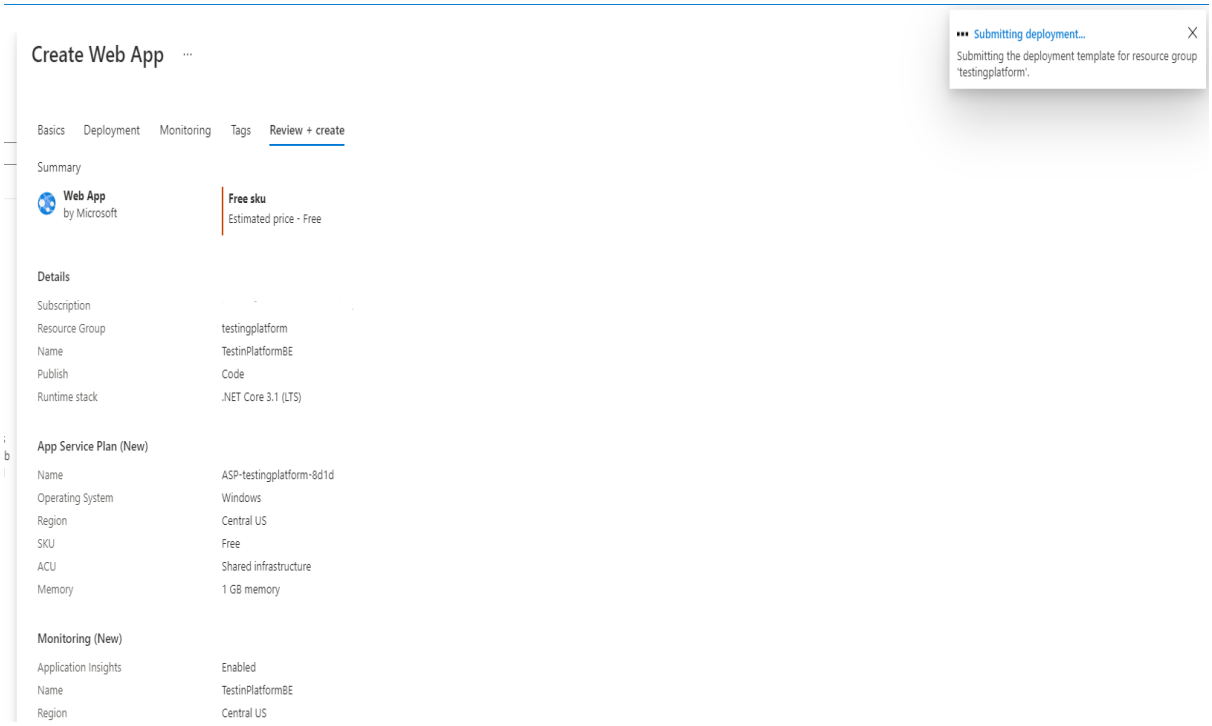


Рисунок 4.15– Створення серверної частини на Azure порталі

Після розгортання серверної частини, потрібно розгорнути клієнтську. Для цього можливо використати редактор коду VS Code та розширення Azure App Service, що дає змогу робити розгортання безпосередньо з редактора (рис 4.16). Для розгортання необхідно додати Azure аккаунт у розширенні, для того щоб отримати доступ до функціональності. Додатковий конфігураційний етап під час розгортання є створення web.config файл, в якому необхідно налаштувати зв'язок з щойно розгорнутою серверною частиною. Це файл може містити інформацію про запити, що повинні робитися на серверну частину з клієнтської. Оскільки фактично запити до

створеної серверної частини будуть виконуватися з окремого сервера для клієнтської частини, так як браузери забороняють викликати стороні ресурси з деякими типами запитів, що можуть оновлювати інформацію про користувача, оскільки фішингові сайти можуть цим користуватися і отримувати інформацію з інших ресурсів.

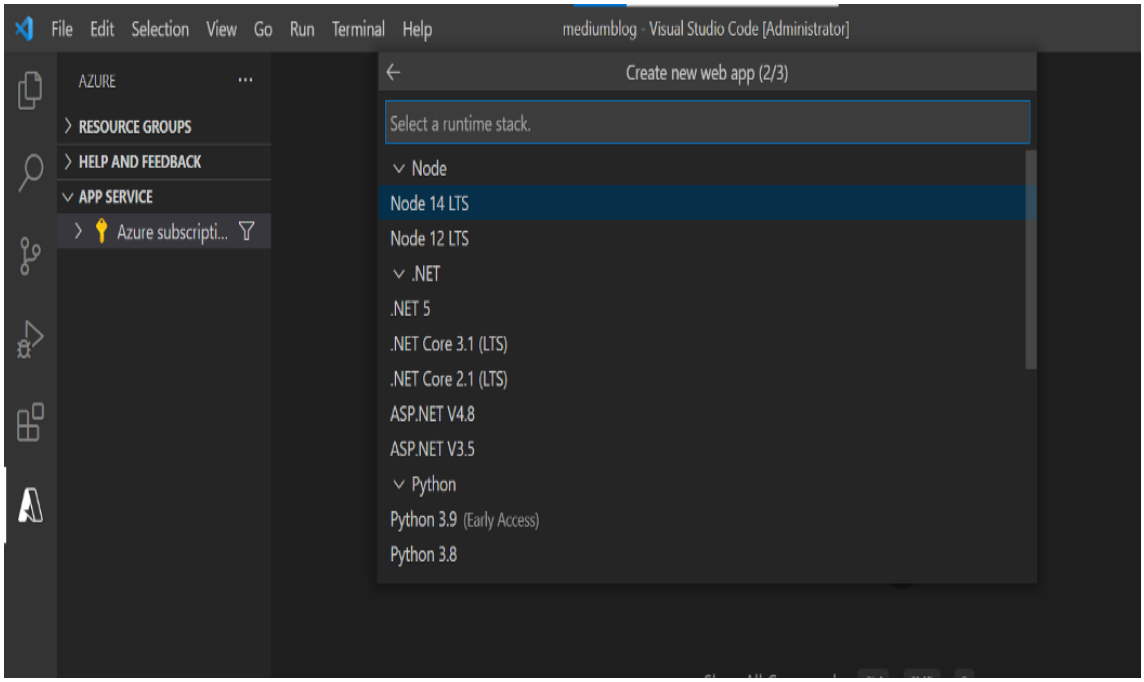


Рисунок 4.16 – Процес розгортання клієнтської частини

Після успішного розгортання клієнтської частини, з використанням хмарної технології Azure, платформа повністю готова до користування, оскільки всі компоненти були успішно об’єднані між собою.

Після розгортання необхідно виконати sanity тестування в реальних умовах, що має включати в себе мінімальний набір тестів, але які покривають основну функціональність додатку. Це дасть змогу перевірити, що всі компоненти були розгорнуті відповідно до поставлених задач.

Для додання підтримки проведення експериментів за допомогою розробленої системи на певний сайт, потрібно налаштувати автоматичне добавлення до всіх сторінок веб-сайту посилання на experiments.js файл, що знаходиться на новоствореній серверній частині. Файл буде містити

інформацію про всі увімкнуті на даний момент експерименти та інформацію про умови за яких вони повинні бути добавлені кінцевому користувачу.

Важливою умовою після успішного розгортання платформи є налаштування моніторингу та нотифікацій. Це надасть можливість дізнаватися про певні проблеми безпосередньо в момент коли вони стаються, за допомогою поштових повідомлень.

Моніторинг надає можливість контролювати в реальному часі стан розгорнутих серверів, їх продуктивність, наповненість пам'яті та функціональність в цілому. Основна ідея – це дізнатися про проблему до того, як її відчують реальні користувачі. Якщо проблему не виявити вчасно та не вирішити, це може вплинути на показники бізнесу.

Azure платформа підтримує можливість налаштування моніторингів та нотифікацій для різних типів сервісів. Ця функціональність може використовувати як і дані зібрані безпосередньо порталом, так і базуючись на логуванні, що було створено під час розробки додатків.

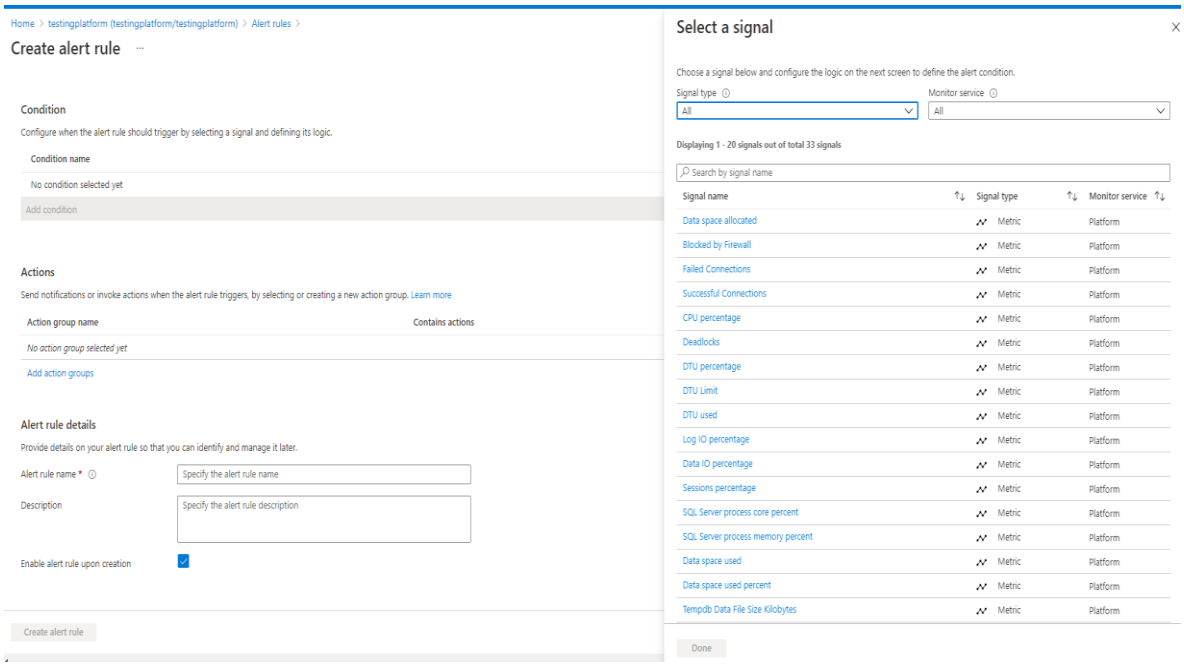


Рисунок 4.16– Створення нотифікацій на Azure порталі

На рисунку 4.16 зображено процес налаштування нотифікацій для створеної бази даних testingplatform. Було налаштовано нотифікації на перевищення розміру бази встановленого ліміту, відсоток IO операцій перевищує 70% та використання CPU, менше ніж 60% від виділеного об’єму. Також, було вказано 1 хвилину, як частоту для перевірки цих даних. Цих показників достатньо для того, щоб бути впевненим, що база даних знаходиться у справному стані. Інформація при настанні цих подій буде відправлятися на поштовий адрес.

4.5 Висновки

У даному розділі було розглянуто сучасні підходи до автоматизованого та ручного тестування програмних додатків, відображено різницю між динамічним та статичним тестуванням. Розглянуто переваги проведення тестування на різних рівнях піраміди для тестування та було вирішено проводити автоматизоване тестування за допомогою написання unit тестів. Цей вид тестів є найшвидшим та має високу стабільність. Було покрито всі класи серверної частини даним видом тестів. Було розглянуто методи тестування за допомогою чорного, білого та сірого ящиків, в результаті чого було обрано метод чорного ящика для ручного тестування платформи для проведення A/B тестування, оскільки він є найоптимальнішим у випадку коли є чіткі вимоги до функціональності. Складено та виконано тест-кейси для того, щоб перевірити, що платформа працює коректно та відповідно до вимог. Під час тестування, у розроблюваному додатку дефектів виявлено не було. Було розгорнуто компоненти розробленої системи для проведення A/B тестування за допомогою хмарної платформи Microsoft Azure.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу впровадження веб-платформи, що надає можливість керувати A/B експериментами на певному сайті.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету д.т.н., проф. Ліщинська Л.Б., к.т.н., доцента Коваленко О.О. з кафедри програмного забезпечення та кафедри комп'ютерних наук к.т.н., доцента Арсенюк І.Р. Технологічний аудит проведемо з використанням таблиці 5.1 [32] де за п'ятибальною шкалою використовуючи 12 критеріїв оцінемо комерційний потенціал.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження табл. 5.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Коваленко О.О.	2. Арсенюк І.Р.	3. Ліщинська Л.Б.
	Бали, виставлені експертами:		
1	3	4	3
2	2	2	3
3	4	2	4
4	3	2	3
5	3	3	3
6	4	2	4
7	2	2	2
8	3	4	4
9	3	3	3
10	4	4	4
11	4	3	4
12	4	4	4
Сума балів	$СБ_1$=39	$СБ_2$=35	$СБ_3$=41
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{39 + 35 + 41}{3} = 38$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 38 бали, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Дану розробку можливо продавати, як готовий продукт, за допомогою B2B моделі. Основними клієнтами можуть виступати власники сайтів, які мають на меті підвищити ефективність веб-сайту, за допомогою зміни певного контенту та дизайну.

Дана платформа надає можливість створювати експерименти, які дають змогу перевірити як впливають нові зміни на користувачів. Цей результат досягається за допомогою, того, що певний відсоток користувачів буде бачити старий дизайн, а інша частина новий. В результаті можливо зібрати певні дані і проаналізувати, чи нові зміни є кращими за попередні.

Для впровадження системи необхідно розробку розгорнути на серверах компанії та зробити конфігураційні зміни, які будуть пов’язувати основний сайт компанії і розроблену платформу, яка буде відповідати за виконання необхідних експериментів.

В якості аналога для розробки було обрано платформу Optimizely. Основними недоліками аналога є відсутність оновлення метрик у реальному часі, відсутність можливість встановити додаткові умови, для користувачів які вже раніше потрапляли на певний експеримент. Також до недоліків можна віднести відсутність можливості розгортання системи на власних серверах.

У розробці дана проблема вирішується шляхом розширення системи з доданням нової функціональності та підтримки швидкого розгортання. Також система випереджає аналог за такими параметрами як швидкодія та швидкість розгортання, оскільки було використано сучасні технології, які не мають прив’язки до певної платформи.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом. В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Кількість одночасних користувачів	20	100	5	25%
Кількість підтримуваних платформ	1	4	4	25%
Максимальна кількість створених експериментів	40	80	2	10%
Відмовостійкість, %	70	90	1,3	20%
Час завантаження сторінки, мс	500	150	3,3	20%

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \tag{5.1}$$

або

$$q_i = \frac{P_{Бi}}{P_{Hi}} \tag{5.2}$$

де P_{Hi}, $P_{Бi}$ – числові значення i-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{100}{20} = 5;$$

$$q_2 = \frac{4}{1} = 4;$$

$$q_3 = \frac{80}{40} = 2;$$

$$q_4 = \frac{90}{70} = 1{,}3;$$

$$q_5 = \frac{500}{150} = 3{,}3.$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i , \qquad (5.3)$$

$$К_{я.в.} = 5 \cdot 0{,}25 + 4 \cdot 0{,}25 + 2 \cdot 0{,}1 + 1{,}3 \cdot 0{,}2 + 3{,}3 \cdot 0{,}2 = 3{,}37$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніший базового товару-конкурента.

Наступним кроком є визначення конкурентоспроможності товару. Конкурентоспроможність товару є головною умовою конкурентоспроможності підприємства на ринку і важливою основою прибутковості його діяльності.

Однією із умов вибору товару споживачем є збіг основних ринкових характеристик виробу з умовними характеристиками конкретної потреби покупця. Такими характеристиками найчастіше вважають нормативні та технічні параметри, а також ціну придбання та вартість споживання товару.

В табл. 5.5 наведено технічні та економічні показники для розрахунку конкурентоспроможності нової розробки відносно товару-аналога, технічні дані взяті з попередніх розрахунків.

Таблиця 5.5 – Нормативні, технічні та економічні параметри нової розробки і товару-виробника

Показники	Варіанти	
	Базовий (товар-конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно-технічні показники		
Кількість одночасних користувачів	20	100
Кількість підтримуваних платформ	1	4
Максимальна кількість створених експериментів	40	80
Відмовостійкість, %	70	90
Час завантаження сторінки, мс	500	150
2. Економічні показники		
Ціна придбання, грн.	15000	10000

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ – індекс технічних параметрів; $I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}}, \tag{5.5}$$

де $P_{Неі}$, $P_{Беі}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$I_{е.п.} = \frac{10000}{15000} = 0{,}66;$$

$$К = \frac{3{,}37}{0{,}66} = 5{,}1.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможніше, ніж конкурентний товар.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)} \qquad (5.6)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки веб-платформи, що надає можливість керувати A/B експериментами на певному сайті необхідно залучити інженера програміста з посадовим окладом 7000 грн. Кількість робочих днів у місяці складає 22, а

кількість робочих днів програміста складає 20. Зведемо сумарні розрахунки до таблиця 5.6.

Таблиця 5.6 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	13000	590,9	5	2955
Інженер-програміст	7000	318,2	20	6364
Всього				9318

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (5.7)$$

$$З_д = 0{,}11 * 9318 = 1025 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.3):

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (5.8)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (9318 + 1025) * \frac{22}{100} = 2275{,}5 \text{ (грн)}$$

4. Витрати на матеріали М та комлектуючі К, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.,} \qquad (5.9)$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг;

$Ц_i$ – вартість матеріалу *і*-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ – маса відходів матеріалу *і*-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу *і*-го найменування, грн/кг;

n – кількість видів матеріалів.

Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i, \qquad (5.10)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Інформацію про використані матеріали та комлектуючі подамо у вигляді табл. 5.7.

Таблиця 5.7 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір	120	1	95
Ручка	15	1	15
CD-диск	12	1	12
Флешка	150	1	150
Всього			297
З врахуванням коефіцієнта транспортування			326,7

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot С_{пргі} \cdot К_і, \quad (5.11)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_і$ = 1,10…1,12).

k – кількість найменувань програмних засобів.

Отримані результати занесемо в таблицю 5.8.

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість
Ліцензія середовища розробки Visual Studio	1	910	910
Microsoft Windows 10	1	3199	3199
Всього			4109

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Ткор \cdot 12} \quad [грн], \tag{5.12}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 10000 грн.

$$\text{А} = \frac{10000 \cdot 1}{2 \cdot 12} = 416{,}67$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і} \quad (5.13)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

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

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 160 \cdot 4{,}1 \cdot 0{,}5}{0{,}8} = 123$$

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 9318 \cdot \frac{100}{100\%} = 9318 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В = 9318+1025+2275,5+326,7+4109+416,67+123+9318=26911,87 \text{ грн.}$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.15)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{26911,87}{0,9} = 29902,07 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.16)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

Цо – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2021 році – 18%.

Припустимо, що при впроваджені результатів наукової розробки покращується якість, що дозволяє підвищити ціну його реалізації на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 22 шт., протягом другого року – на 18 шт., протягом третього року на 12 шт. Реалізація продукції до впровадження розробки складала 150 шт., а її ціна 10000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [1000 \cdot 150 + (10000 + 1000) \cdot 22] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 66963{,}98 \text{ грн.}$$

$$\Delta П_2 = [1000 \cdot 150 + (10000 + 1000) \cdot (22 + 18)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 225163{,}66 \text{ грн.}$$

$$\Delta П_З = [1000 \cdot 150 + (10000 + 1000) \cdot (22 + 18 + 12)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 247712{,}76 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (5.17)$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 29902{,}07 = 59804{,}95$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \qquad (5.18)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

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

(5.19)

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{66963,99}{(1+0,2)^1} + \frac{225163,66}{(1+0,2)^2} + \frac{247712,76}{(1+0,2)^3} = 356186,02 \text{ грн.}$$

$$Е_{абс} = (356186,02 - 59804,95\ ) = 296381,07 \text{грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.20)$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{296381,07}{59804,95}} - 1 = 1,22 = 122\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.21}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2018 році в Україні d= (0,14…0,2);

f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0,18 + 0,05 = 0,23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.22}$$

$$\text{Т}_{\text{ок}} = \frac{1}{1,22} = 0,8 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Висновки

Було проведено оцінку комерційного потенціалу розробки веб-платформа, що надає можливість керувати A/B експериментами на певному сайті, яка є на вище середньому рівні. При порівнянні нової розробки з аналогом виявлено, що вона є якіснішою і конкурентоспроможнішою відносно аналога, а також краще по технічним і економічним показникам.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 26911,87 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 29902,07 грн.

Вкладені інвестиції в даний проект окупляться через 8 місяців при прогнозованому прибутку 356186,02 грн. за три роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено програмний додаток, який дозволяє інтегрувати підтримку проведення A/B тестування на обраному веб-сайті. Основна ідея, полягає у можливості створювати у простий спосіб експерименти, для їх подальшого аналізу та ефективної адаптації веб-сайту на основі зібраної інформації. У роботі було розглянуто важливість проведення експериментів для ефективної адаптації веб-сайтів до вимог користувачів, проаналізовано різні підходи до проведення користувацьких експериментів. Також, розглянуто існуючі системи для створення A/B експериментів, виявлено їх недоліки та переваги, на основі чого було вирішено розробити власний програмний продукт. Було складено вимоги до основної функціональності платформи.

У ході роботи було розроблено метод поєднання Бета тестування у парі з A/B тестуванням. Ключова ідея, що лежить в даному методі це зменшення впливу не виявлених помилок в експерименті на початкових етапах проведення без впливу на велику кількість користувачів, що підвищує клієнтоорієнтованість компанії. Також, отримав подальшого розвитку метод синтетичного контролю для проведення A/B експериментів, було запропоновано використання додаткових груп для можливості додаткової перевірки відсутності впливу внесених змін на контрольну групу.

У роботі було запроектовано архітектуру розроблюваного додатку, що відображає взаємодію між клієнтською, серверною частинами та базою даних. Було розглянуто важливість підтримки авторизації та автентифікації у веб-додатках, та використання захищених протоколів, для попередження несанкціонованого доступу у систему. При проектуванні бази даних було створено універсальне відношення на основі якого було розроблено

структуру бази даних. Відображено функціонал платформи для експериментів за допомогою загального алгоритму.

Було розглянуто мови програмування для розробки веб-додатків та обрано C#, як найоптимальнішу мову для розроблювальної платформи. В результаті аналізу основних платформ для розробки було обрано платформу ASP.NET Core, оскільки вона підходить для розробки великих проектів та підтримує високі навантаження, що є необхідною умовою для якісною розробки продукту. Було порівняно СУБД, що використовуються для реляційних баз даних, на основі чого було обрано SQL Server для даної розробки. У роботі було розроблено весь необхідний функціонал для повноцінної можливості проведення A/B експериментів, описано та продемонстровано основні модулі, що використовується у додатку.

Розглянуто сучасні методи тестування веб-додатків, проаналізовано піраміду тестування, на основі чого було прийнято рішення про важливість покриття додатку автоматизованими тестами. Для автоматизованого тестування було обрано unit тестування, оскільки дані тести є швидкими у виконанні та мають високу стабільність. Ручне тестування було здійснено за допомогою методу чорного ящика, було розроблено тест-кейси та перевірено працездатність. Було описано процес розгортання та впровадження системи з використанням хмарної платформи Azure.

Виконано економічний аналіз та розрахунки, результати яких підтвердили економічну доцільність даної розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Importance Of Improving User Experience. URL: https://www.forbes.com/sites/forbesagencycouncil/2017/06/15/the-importance-of-improving-user-experience (дата звернення: 20.10.2021).

2. A/B Testing Guide. URL: https://vwo.com/ab-testing (дата звернення: 22.10.2021).

3. Бугайов В. Ю., Коваленко О. О. Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів. Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної наук.-практ. Інтернет конференції, 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021. С. 34-37.

4. Бугайов В. Ю., Коваленко О. О. Використання процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів. Theoretical and empirical scientific research: concept and trends. Збірник матеріалів III Міжнародної наук.-практ. конференції, 10 грудня 2021 р. – Оксфорд/Вінниця: LLC OXFORD SCIENCES LTD/European Scientific Platform, 2021. Т. 2. С. 48-49. DOI: https://doi.org/10.36074/logos-10.12.2021.

5. What Is the Software Development Life Cycle? URL: https://phoenixnap.com/blog/software-development-life-cycle (дата звернення: 27.10.2021).

6. The power of A/B testing in product development. URL: https://medium.com/agileinsider/the-power-of-a-b-testing-in-product-development-2a783777b66 (дата звернення: 24.10.2021).

7. A Refresher on A/B Testing. URL: https://hbr.org/2017/06/a-refresher-on-ab-testing (дата звернення: 24.10.2021).

8. Multivariate testing. URL: https://www.optimizely.com/optimization-glossary/multivariate-testing (дата звернення: 01.11.2021).

9. When to Run Bandit Tests Instead of A/B/n Tests. URL: https://cxl.com/blog/bandit-tests (дата звернення: 03.11.2021).

10. Everything You Need to Know About Beta Testing. URL: https://xd.adobe.com/ideas/process/user-testing/everything-you-need-to-know-about-beta-testing (дата звернення: 04.11.2021).

11. Synthetic Control Method. URL: https://medium.com/analytics-vidhya/synthetic-control-method-5c01f72da4e (дата звернення: 07.11.2021).

12. Optimizely. URL: https://en.wikipedia.org/wiki/Optimizely (дата звернення: 10.11.2021).

13. Website, A/B Testing & Optimization Tools. URL: https://marketingplatform.google.com/about/optimize (дата звернення: 12.11.2021).

14. VWO Testing Reviews & Product Details. URL: https://www.g2.com/products/wingify-vwo-testing/reviews (дата звернення: 14.11.2021).

15. Determining sample size: how to make sure you get the correct sample size. URL: https://www.qualtrics.com/uk/experience-management/research/determine-sample-size (дата звернення: 15.11.2021).

16. George H. Fairbanks. Just Enough Software Architecture: A Risk-Driven Approach. Boulder : Marshall & Brainerd, 2010. 376 p.

17. Alex Berson. Client/Server Architecture. New York : McGraw-Hill, 1996. 569 p.

18. Security Authentication vs. Authorization. A Quick Guide. URL: https://swoopnow.com/security-authentication-vs-authorization (дата звернення: 19.11.2021).

19. JWT Authentication Flow with Refresh Tokens in ASP.NET Core Web API. URL: https://fullstackmark.com/post/19/jwt-authentication-flow-with-refresh-tokens-in-aspnet-core-web-api (дата звернення: 21.11.2021).

20. How to enable HTTPS on your server. URL: https://www.godaddy.com/garage/enable-https-server (дата звернення: 25.11.2021).

21. Relational Decomposition. URL: Режим доступу: https://javatpoint.com/bms-relational-decomposition (дата звернення: 25.11.2021).

22. The Definition of User Experience (UX). URL: https://nngroup .com/articles/definition-user-experience (дата звернення: 25.11.2021).

23. What is PHP? Write your first PHP Program. URL: https://www.guru99.com/what-is-php-first-php-program.html (дата звернення: 26.11.2021).

24. JavaScript: Here’s What You Need To “Learn JavaScript Deeply”. URL: https://html.com/javascript (дата звернення: 26.11.2021).

25. Mark J. Price. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development. Birgmingham : Packt Publishing, 2017. 641 p.

26. A brief history of Node.js. URL: https://nodejs.dev/a-brief-history-of-nodejs (дата звернення: 26.11.2021).

27. MySQL – Introduction. URL: https://tutorialspoint.com/mysql/ mysql-introduction.htm (дата звернення: 27.11.2021).

28. MS SQL Server. URL: https://searchsqlserver.tech.target.com /definition/SQL-Server (дата звернення: 27.11.2021).

29. Software testing – Wikipedia. URL: https://en.wikipedia.org /wiki/Software_testing (дата звернення: 28.11.2021).

30. Srinivasan Desikan. Software Testing. Principles and Practices. Vancuver : Pearson, 2009. 480 p.

31. How to configure and use Live Unit Testing. URL: https://docs.microsoft.com/en-us/visualstudio/test/live-unit-testing (дата звернення: 28.11.2021).

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

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
" 13 " вересня 2021 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доц. О.О. Коваленко
"____"_____2021 р.
Виконав:
_____студент гр.1ПІ-20м В.Ю. Бугайов
"____"_____2021 р.

Вінниця – 2021 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів».

Галузь застосування – веб-сайти.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 277 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів.

Призначення роботи – розробка платформи для A/B тестування, що дає можливість впровадити проведення експериментів на веб-сайті.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Mark J. Price. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development. Birgmingham : Packt Publishing, 2017. 641 p.
2. Alex Berson. Client/Server Architecture. New York : McGraw-Hill, 1996. 569 p.
3. Srinivasan Desikan. Software Testing. Principles and Practices. Vancuver : Pearson, 2009. 480 p.
4. George H. Fairbanks. Just Enough Software Architecture: A Risk-Driven Approach. Boulder : Marshall & Brainerd, 2010. 376 p.

5. Технічні вимоги

Середовище розробки – ASP.NET Core.

Мова програмування – C#, TypeScript, SQL.

Тип додатку – веб-додаток.

6. Конструктивні вимоги.

Конструкція додатку повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз, вибір та обґрунтування актуальності розробки додатку	15.09.2021 – 27.09.2021
2	Розробка структури та алгоритму роботи додатку	28.09.2021 – 15.10.2021
3	Розробка та тестування модулів додатку	16.10.2021 – 01.11.2021
4	Економічна частина	02.11.2021 – 11.11.2021
5	Оформлення матеріалів до захисту	12.11.2021 – 30.11.2021

	магістерської кваліфікаційної роботи	

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Протокол перевірки роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 20м

Науковий керівник: к.т.н. доц. Коваленко О. О.

Unicheck	
Оригінальність	**96,6 %**
Схожість	3,4 %

Аналіз звіту подібності

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

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

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

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Удосконалення процесів A/B тестування для ефективної адаптації веб-сайту до вимог користувачів».

Автор _____ Бугайов Володимир Юрійович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____

(за потреби) (підпис) (прізвище, ініціали, посада)

Додаток В. Лістинг програми

Лістинг серверної частини платформи для проведення A/B експериментів

ExperimentController.cs

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using TestingPlatformBackend.Models;


namespace TestingPlatformBackend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ExperimentController : ControllerBase
    {
        readonly PlatformContext _context;

        public ExperimentController(PlatformContext context)
        {
            _context = context;
        }
        // GET: api/<ExperimentController>
        [HttpGet]
        public IEnumerable<Experiment> Get()
        {
            return _context.Experiments.ToList();
        }

        // GET api/<ExperimentController>/5
        [HttpGet("{id}")]
        public Experiment Get(int id)
        {
            return
_context.Experiments.Include(e=>e.Variations).FirstOrDefault(e=>e.ExperimentId==id);
        }

        // POST api/<ExperimentController>
        [HttpPost]
        public int Post([FromBody] Experiment experiment)
        {
            _context.Experiments.Add(experiment);
            _context.SaveChanges();
            return experiment.ExperimentId;
        }

        // PUT api/<ExperimentController>/5
        [HttpPut("{id}")]
        public void Put(int id, [FromBody] Experiment value)
        {
            var experiment = _context.Experiments.FirstOrDefault(e=>e.ExperimentId ==
id);
            experiment.Assignment = value.Assignment;
            experiment.Description = value.Description;
            experiment.Name = value.Name;
```

```
            _context.SaveChanges();
        }

        [HttpPut("changestatus/{id}")]
        public void Put(int id)
        {
            var experiment = _context.Experiments.FirstOrDefault(e => e.ExperimentId ==
id);
            experiment.IsEnabled =!experiment.IsEnabled;
            _context.SaveChanges();
        }

        public Experiment GetExperiment(List<Experiment> experiments, int experimentId)
        {
            return experiments.FirstOrDefault(experiments => experiments.ExperimentId ==
experimentId);
        }
    }
}
```

VariationController.cs

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using TestingPlatformBackend.Models;


namespace TestingPlatformBackend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class VariationController : ControllerBase
    {
        readonly PlatformContext _context;

        public VariationController(PlatformContext context)
        {
            _context = context;
        }

        // GET api/<ExperimentController>/5
        [HttpGet("{id}")]
        public IEnumerable<Variation> Get(int id)
        {
            return _context.Variations.ToList().Where(e => e.ExperimentId==id);
        }

        // POST api/<ExperimentController>
        [HttpPost]
        public int Post(Variation variation)
        {
            _context.Variations.Add(variation);
            _context.SaveChanges();

            return variation.VariationId;
        }

        // PUT api/<ExperimentController>/5
        [HttpPut("{id}")]
        public void Put(int id, [FromBody] Variation value)
```

```
        {
            var variation = _context.Variations.FirstOrDefault(e => e.VariationId == id);
            variation.ExecutionCode = value.ExecutionCode;
            variation.Name = value.Name;
            _context.SaveChanges();
        }

        // PUT api/<ExperimentController>/5
        [HttpPut("updatetraffic")]
        public void Put([FromBody] List<ExperimentTraffic> experimentTraffic)
        {
            var variations = _context.Variations.ToList();
            foreach (var VARIABLE in experimentTraffic)
            {

variations.FirstOrDefault(f=>f.VariationId==VARIABLE.Id).PercentageOfTraffic=VARIABLE.Per
centageOfTraffic;

            }
            _context.SaveChanges();
        }

        // DELETE api/<ExperimentController>/5
        [HttpDelete("{id}")]
        public void Delete(int id)
        {
            _context.Variations.Remove(_context.Variations.FirstOrDefault(e =>
e.VariationId == id));
            _context.SaveChanges();
        }
    }
       }
```

Experiments.cs

```
using System.Collections.Generic;

namespace TestingPlatformBackend.Models
{
    public class Experiment
    {
        public int ExperimentId { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public string Assignment { get; set; }
        public bool IsEnabled { get; set; }

        public List<Variation> Variations { get; set; }
    }
}
```

ExperimentTraffics.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace TestingPlatformBackend.Models
{
    public class ExperimentTraffic
    {
        public int Id { get; set; }
        public int PercentageOfTraffic { get; set; }
    }
}
```

PlatformContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace TestingPlatformBackend.Models
{
public class PlatformContext : DbContext
{
    public DbSet<Experiment> Experiments { get; set; }
    public DbSet<Variation> Variations { get; set; }


        public PlatformContext(DbContextOptions<PlatformContext> options) : base(options)
    {

    }
}
}
```

Variation.cs

```
namespace TestingPlatformBackend.Models
{
    public class Variation
    {
        public int VariationId { get; set; }
        public string Name { get; set; }
        public int PercentageOfTraffic { get; set; }
        public string ExecutionCode { get; set; }
        public int ExperimentId { get; set; }
    }
}
```

Лістинг клієнтської частини платформи для проведення A/B експериментів

CreateExperimnet.tsx

```
import { Button, TextField } from '@mui/material'
import React, { Fragment } from 'react'
import { useHistory } from 'react-router-dom';
import { axiosConfig, BACK_END_HOST_LOCATION } from '../Configuration';
import './CreateExperimentStyle.css';
```

```
const axios = require('axios').default;

export const CreateExperiment: React.FC = () => {
   let history = useHistory();

   async function createExperiment() {
      try {
         const data = { name: name, description: description, assignment: assignment};
        const response = await axios.post(BACK_END_HOST_LOCATION + '/api/experiment',
data, axiosConfig);
        history.push(`/experiment/${response.data}`);
        history.go(0)
      } catch (error) {
        alert(error);
      }
    }

   const [name, setName] = React.useState('');

   const handleNameChange = (event) => {
      setName(event.target.value);
   };

   const [description, setDescription] = React.useState('');

   const handleDescriptionChange = (event) => {
      setDescription(event.target.value);
   };

   const [assignment, setAssignment] = React.useState('');

   const handleAssignmentChange = (event) => {
      setAssignment(event.target.value);
   };

   const onCreateButtonClick = () => {
      createExperiment();
    }

  return (
   <Fragment>
      <div className="createExperiment_container">
      <TextField id="outlined-basic" label="Назва" variant="outlined"  sx={{marginBottom:
"20px", width: "65%"}}
                value={name}
                onChange={handleNameChange} />
      <TextField
          id="outlined-multiline-static"
          label="Опис"
          multiline
          rows={4}
          sx={{marginBottom: "20px", width: "65%"}}
          value={description}
          onChange={handleDescriptionChange}
        />
             <TextField
          id="outlined-multiline-static"
          label="Умова попадання на експеримент (JavaScript)"
          multiline
          rows={4}
          sx={{marginBottom: "20px", width: "65%"}}
          value={assignment}
          onChange={handleAssignmentChange}
        />
            <div>
```

```
      <Button variant="contained" onClick={onCreateButtonClick}>Створити</Button>
      </div>
      </div>
   </Fragment>
  )
}
```

ListOfExperiments.tsx

```
import { Box, Button, Divider, List, ListItem, ListItemButton, ListItemIcon,
ListItemText, TextField } from '@mui/material'
import React, { Fragment, useEffect, useState } from 'react'
import { axiosConfig, BACK_END_HOST_LOCATION } from '../Configuration';
import './CreateExperimentStyle.css';
const axios = require('axios').default;

export const ListOfExperiments: React.FC = () => {
   const [listOfExperiments, setListOfExperiments] = useState([]);

   async function getExperiments() {
      try {
        const response = await axios.get(BACK_END_HOST_LOCATION + '/api/experiment',
axiosConfig);
        setListOfExperiments(response?.data?.map(experiment =>  {
           return <Fragment>
         <ListItem disablePadding>
            <ListItemButton  component="a"
href={`/experiment/${experiment.experimentId}`}>
           <ListItemText primary={experiment.experimentId} />
           <ListItemText primary={experiment.name} />
         </ListItemButton>
       </ListItem>
                 <Divider />
      </Fragment>
      }
   ))
      } catch (error) {
        alert(error);
      }
    }
   useEffect(() => {
      getExperiments();
     },[]);
  return (
   <Fragment>
        <Box sx={{ width: '70%', bgcolor: 'background.paper' }}>
      <nav aria-label="main mailbox folders">
        <List>
        <Divider />
        {listOfExperiments}
        </List>
        </nav>
        </Box>
   </Fragment>
  )
}
```

ViewExperiment.tsx

```
import { Box, Button, Dialog, DialogActions, DialogContent, DialogContentText,
DialogTitle, Divider, Fab, List, ListItem, ListItemButton, ListItemText, TextField } from
'@mui/material'
import React, { Fragment, useEffect, useState } from 'react'
import { Redirect, useHistory, useParams } from 'react-router-dom';
import { axiosConfig, BACK_END_HOST_LOCATION } from '../Configuration';
import './ViewExperimentStyle.css';
import AddIcon from '@mui/icons-material/Add';
const axios = require('axios').default;
import DeleteForeverIcon from '@mui/icons-material/DeleteForever';

export const ViewExperiment: React.FC = () => {
    let { id } = useParams<{ id?: string }>();
    let { variationId } = useParams<{ variationId?: string }>();
    const [listOfVariations, setListOfVariations] = useState([]);
    const [listOfVariationsData, setListOfVariationsData] = useState([]);
    const [variationName, setVariationName] = useState("");
    const [variationCode, setVariationCode] = useState("");
    const [trafficPercentage, setTrafficPercentage] = useState([]);
    const [open, setOpen] = React.useState(false);
    const [isEnabled, setIsEnabled] = useState(false);

    const handleClickOpen = () => {
        setOpen(true);
    };

    const handleClose = () => {
        history.push(`/experiment/${id}`);
        setOpen(false);
    };

    useEffect(() => {
        loadExperiment();
    }, []);

    async function deleteVariation(event, variationId1) {
        try {
            const response = await axios.delete(BACK_END_HOST_LOCATION +
`/api/variation/${variationId1}`, axiosConfig);
            history.go(0)
        } catch (error) {
            alert(error);
        }
    }

    const [name, setName] = React.useState('');

    const handleNameChange = (event) => {
        setName(event.target.value);
    };
    const handleVariationCodeChange = (event) => {
        setVariationCode(event.target.value);
    };
    const handleVariationNameChange = (event) => {
        setVariationName(event.target.value);
    };

    const [description, setDescription] = React.useState('');

    const handleDescriptionChange = (event) => {
        setDescription(event.target.value);
    };

    const [assignment, setAssignment] = React.useState('');
```

```
    const handleAssignmentChange = (event) => {
        setAssignment(event.target.value);
    };


    async function loadExperiment() {
        try {
            const response = await axios.get(BACK_END_HOST_LOCATION +
`/api/experiment/${id}`, axiosConfig);
            setIsEnabled(response?.data?.isEnabled);
            setName(response.data.name);
            setDescription(response?.data?.description);
            setAssignment(response?.data?.assignment)
            if (variationId) {
                if (response?.data?.variations.find(v => v.variationId ==
variationId)?.name)
                    setVariationName(response?.data?.variations.find(v => v.variationId
== variationId)?.name);
                if (response?.data?.variations.find(v => v.variationId ==
variationId)?.executionCode)
                    setVariationCode(response?.data?.variations.find(v => v.variationId
== variationId)?.executionCode);
                handleClickOpen();
            }
            setListOfVariationsData(response?.data?.variations);
            var variations = response?.data?.variations.map(v => {
                return { variationId: v.variationId, percentageOfTraffic:
v.percentageOfTraffic, name: v.name }
            });
            setTrafficPercentage(variations);
            setListOfVariations(response?.data?.variations.map(variation => {
                return <Fragment>
                    <ListItem disablePadding>
                        <ListItemButton sx={{ width: "160px" }} component="a"
href={`/experiment/${id}/${variation.variationId}`}>
                            <ListItemText primary={variation.variationId} />
                            <ListItemText primary={variation.name} />
                        </ListItemButton>
                        <ListItemButton onClick={e => deleteVariation(e,
variation.variationId)}>
                            <DeleteForeverIcon />
                        </ListItemButton>
                    </ListItem>
                    <Divider />
                </Fragment>
            }
            ))
        } catch (error) {
            alert(error);
        }
    }

    const onCreateButtonClick = () => {
        updateExperiment();
    }

    async function updateExperiment() {
        try {
            const data = { name: name, description: description, assignment: assignment
};
            const response = await axios.put(BACK_END_HOST_LOCATION +
`/api/experiment/${id}`, data, axiosConfig);
        } catch (error) {
            alert(error);
        }
```

```
    }
    let history = useHistory();

    async function addVariation() {
        const data = { name: 'New Variation', executionCode: '', experimentId:
Number.parseInt(id) };
        const response = await axios.post(BACK_END_HOST_LOCATION + `/api/variation`,
data, axiosConfig);

        setVariationName('New Variation');
        setVariationCode('');
        history.push(`/experiment/${id}/${response.data}`);
        handleClickOpen();
    }

    async function handleUpdateVariation() {
        try {
            const data = { name: variationName, executionCode: variationCode };
            await axios.put(BACK_END_HOST_LOCATION + `/api/variation/${variationId}`,
data, axiosConfig);
            handleClose();
            history.push(`/experiment/${id}`);
            history.go(0)
        } catch (error) {
            alert(error);
        }
    }

    function updateTraffic(id, traffic) {
        const prevState = JSON.parse(JSON.stringify(trafficPercentage));
        prevState.find(v => v.variationId == id).percentageOfTraffic = traffic;
        setTrafficPercentage(prevState);
    }

    function getVariationsTraffic() {
        return trafficPercentage.map((v) => {
            return (
                <div className="viewexperiment_variationtraffic">
                    <label className="viewexperiment_label">
                        {v.name}
                    </label>
                    <input type="text" style={{ width: "30PX" }}
defaultValue={v.percentageOfTraffic} onChange={th => updateTraffic(v.variationId,
th.currentTarget.value)} />
                </div>)
        })
    }

    async function onChangeTrafficButtonClick() {
        var sum = trafficPercentage.map(t =>
Number.parseInt(t.percentageOfTraffic)).reduce((partial_sum, a) => partial_sum + a, 0)
        if (sum != 100)
            alert("Сума варіацій не рівна 100%");
        else {
            var diff = [];
            trafficPercentage.forEach(t => {
                if (t.percentageOfTraffic != listOfVariationsData.find(l => l.variationId
== t.variationId).percentageOfTraffic) {
                    diff.push({ id: t.variationId, percentageOfTraffic:
Number.parseInt(t.percentageOfTraffic) });
                }
            })
            if (diff.length != 0) {
                try {
```

```
                    await axios.put(BACK_END_HOST_LOCATION +
`/api/variation/updatetraffic`, diff, axiosConfig);
                    history.go(0)
                } catch (error) {
                    alert(error);
                }
            }
        }
    }

    async function changeStatus() {
        setIsEnabled(!isEnabled);
        try {
            const response = await axios.put(BACK_END_HOST_LOCATION +
`/api/experiment/changestatus/${id}`, axiosConfig);
        } catch (error) {
            alert(error);
        }
    }

    return (
        <Fragment>
            <div className="viewExperiment_root">
                <Dialog open={open} onClose={handleClose}>
                    <DialogTitle>            Варіація {variationId}
                    </DialogTitle>
                    <DialogContent>

                        <TextField
                            autoFocus
                            margin="dense"
                            id="name"
                            label="Назва"
                            type="email"
                            fullWidth
                            variant="standard" sx={{ marginBottom: "55px" }}
                            value={variationName}
                            onChange={handleVariationNameChange}
                        />
                        <TextField
                            id="outlined-multiline-static"
                            label="Код виконання на сторінці"
                            multiline
                            rows={4}
                            sx={{ marginBottom: "20px", width: "65%" }}
                            value={variationCode}
                            onChange={handleVariationCodeChange}
                        />
                    </DialogContent>
                    <DialogActions>
                        <Button onClick={handleClose}>Закрити</Button>
                        <Button onClick={handleUpdateVariation}>Оновити</Button>
                    </DialogActions>
                </Dialog>
                <div className="viewExperiment_variations">
                    <Box sx={{ bgcolor: 'background.paper' }}>
                        <nav aria-label="main mailbox folders">
                            <List>
                                <Divider />
                                {listOfVariations}
                                <ListItem disablePadding>
                                    <Fab color="primary" aria-label="add" sx={{ height:
"26px" }} onClick={addVariation} >
                                        <AddIcon sx={{ height: "26px" }} />
```

```
                                    </Fab>         <ListItemText primary="Створити
варіацію" sx={{ paddingLeft: "19px" }} />
                                </ListItem>

                            </List>
                        </nav>
                    </Box>
                </div>
                <div className="viewExperiment_container">
                    <TextField id="outlined-basic" label="Назва" variant="outlined" sx={{
marginBottom: "20px", width: "65%" }}
                        value={name}
                        onChange={handleNameChange} />
                    <TextField
                        id="outlined-multiline-static"
                        label="Опис"
                        multiline
                        rows={4}
                        sx={{ marginBottom: "20px", width: "65%" }}
                        value={description}
                        onChange={handleDescriptionChange}
                    />
                    <TextField
                        id="outlined-multiline-static"
                        label="Умова попадання на експеримент (JavaScript)"
                        multiline
                        rows={4}
                        sx={{ marginBottom: "20px", width: "65%" }}
                        value={assignment}
                        onChange={handleAssignmentChange}
                    />
                    <div>
                        <Button variant="contained"
onClick={onCreateButtonClick}>Оновити</Button>
                    </div>
                    <div className="trafficPercantage_control">
                        <div >
                            {getVariationsTraffic()}
                            <Button sx={{ marginTop: "20px" }} variant="contained"
onClick={onChangeTrafficButtonClick}>Змінити розподіл трафіку</Button>
                        </div>
                        {!isEnabled ?
                            <Button variant="contained" color="success" sx={{ height:
"40px", width: "250px" }} onClick={changeStatus}>
                                Запустити експеримент
</Button> : <Button variant="contained" color="error" sx={{ height: "40px", width:
"250px" }} onClick={changeStatus}>
                                Зупинити експеримент
</Button>}
                    </div>
                </div>
            </div>
        </Fragment>
    )
}
```

App.tsx

```
import React from 'react'
import { BrowserRouter, Switch, Route } from 'react-router-dom'
```

```
import { Navbar } from './components/Navbar'
import { About } from './pages/About'
import { CreateExperiment } from './pages/CreateExperiment'
import { Home } from './pages/Home'
import { ListOfExperiments } from './pages/ListOfExperiments'
import { ViewExperiment } from './pages/ViewExperiment'

const App: React.FC = () => {
  return (
    <BrowserRouter>
      <Navbar />
      <div className="container">
        <Switch>
          <Route path="/" component={ListOfExperiments} exact />
          <Route path="/createexperiment" component={CreateExperiment} />
          <Route path="/list" component={ListOfExperiments} />
          <Route path="/experiment/:id/:variationId?" component={ViewExperiment} />
        </Switch>
      </div>
    </BrowserRouter>
  )
}

export default App
```

CreateExperimentStyle.css

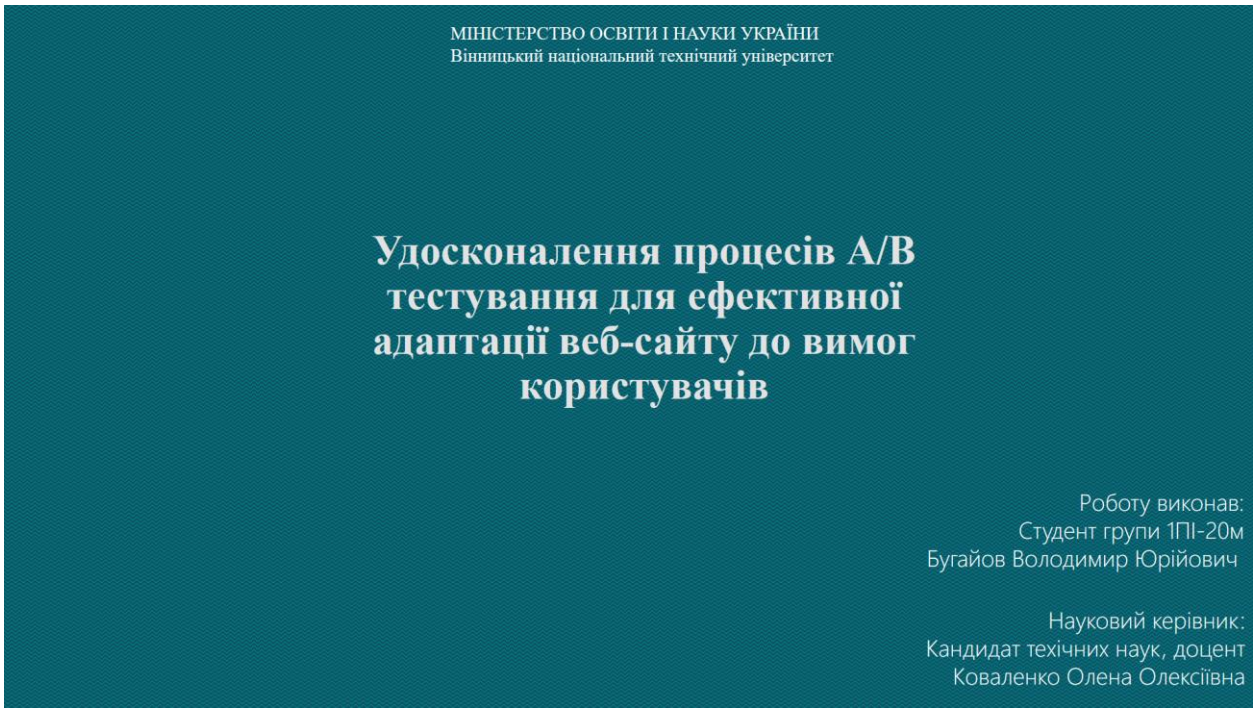
```
.createExperiment_container{
   display: flex;
   padding-top:20px;
   padding-left: 20px;
   flex-direction: column;
}
```

Додаток Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

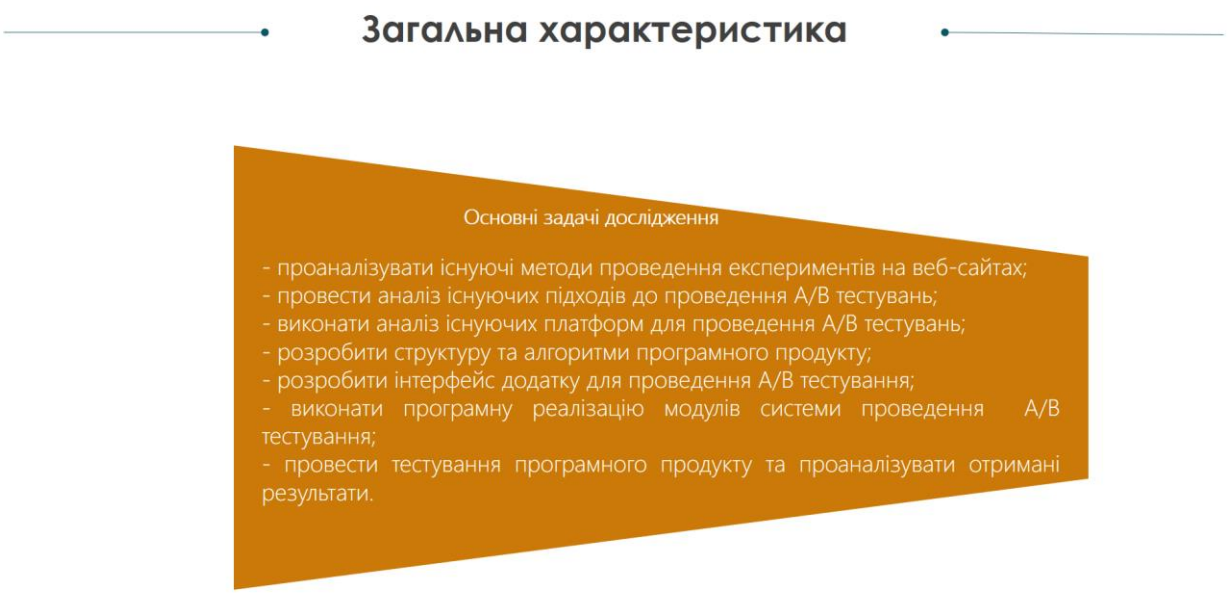
УДОСКОНАЛЕННЯ ПРОЦЕСІВ А/В ТЕСТУВАННЯ ДЛЯ ЕФЕКТИВНОЇ АДАПТАЦІЇ ВЕБ-САЙТУ ДО ВИМОГ КОРИСТУВАЧІВ

Плакат 1 – Тема, автор, науковий керівник магістерської кваліфікаційної роботи

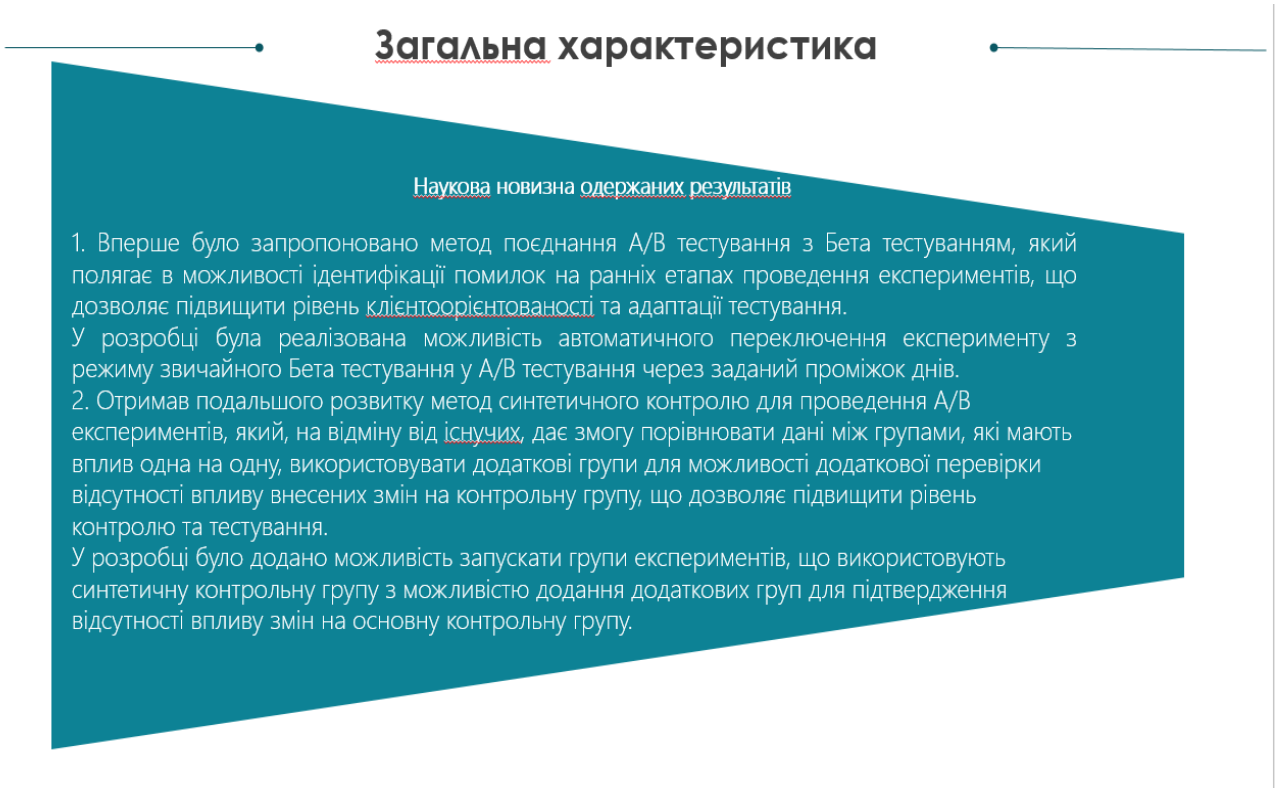


Плакат 2 – Загальна характеристика роботи

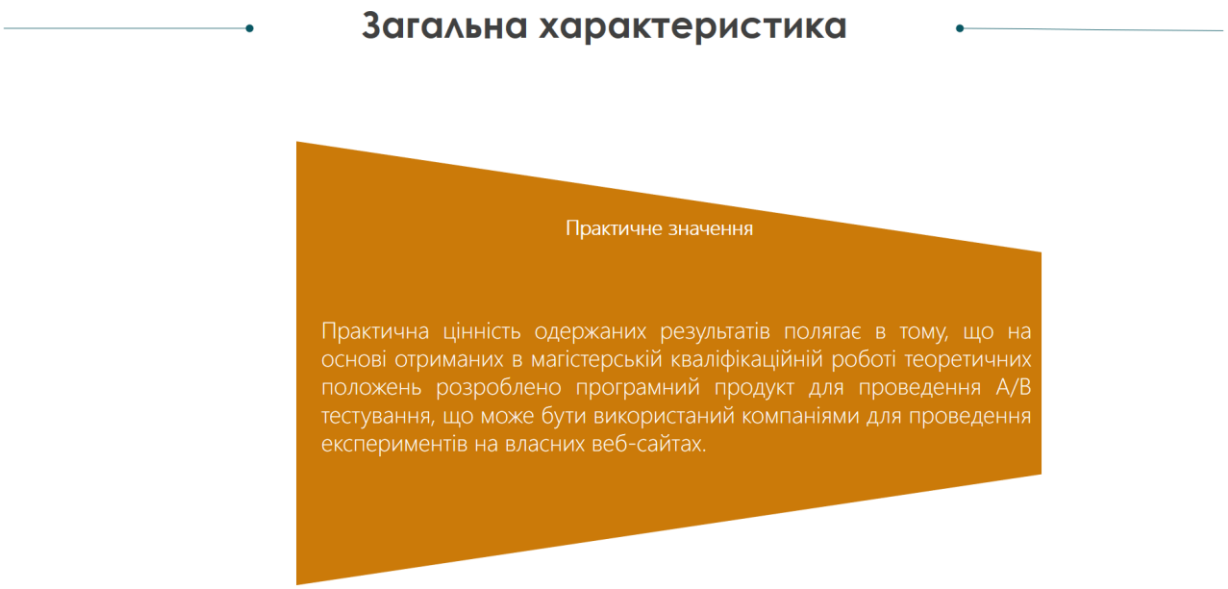
Плакат 3 – Основні задачі дослідження



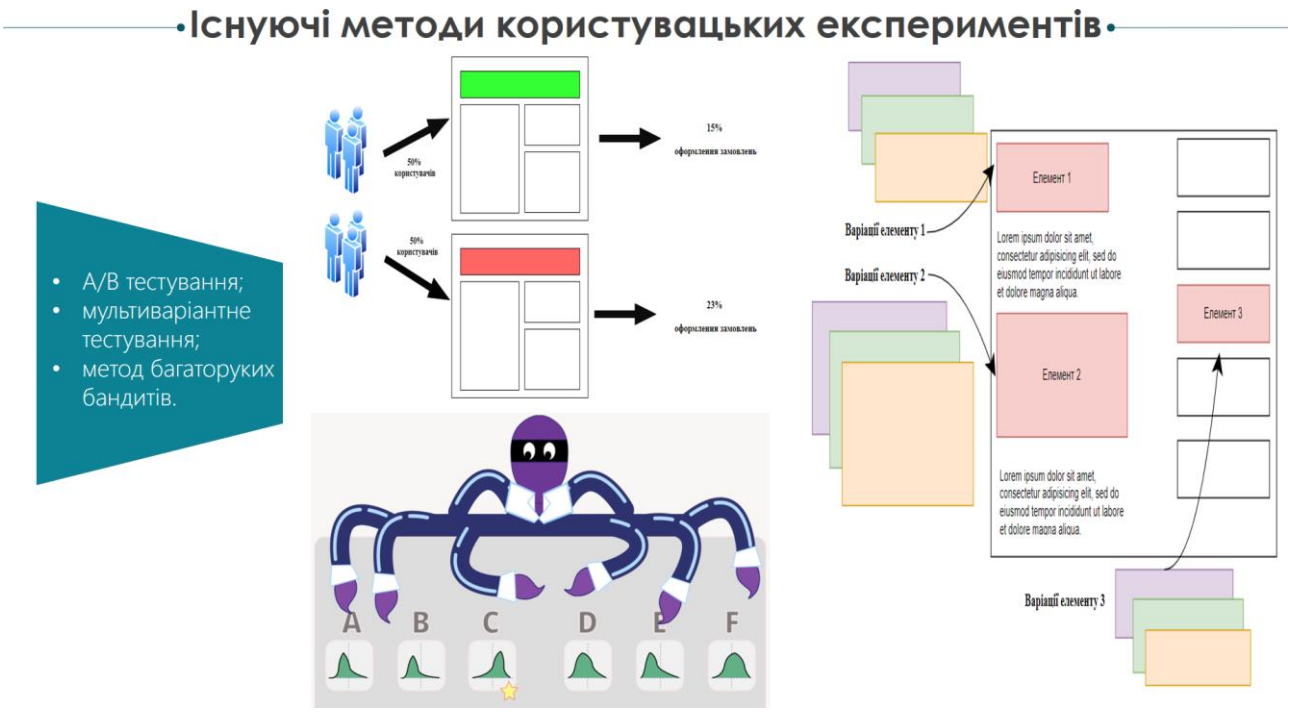
Плакат 4 – Наукова новизна одержаних результатів



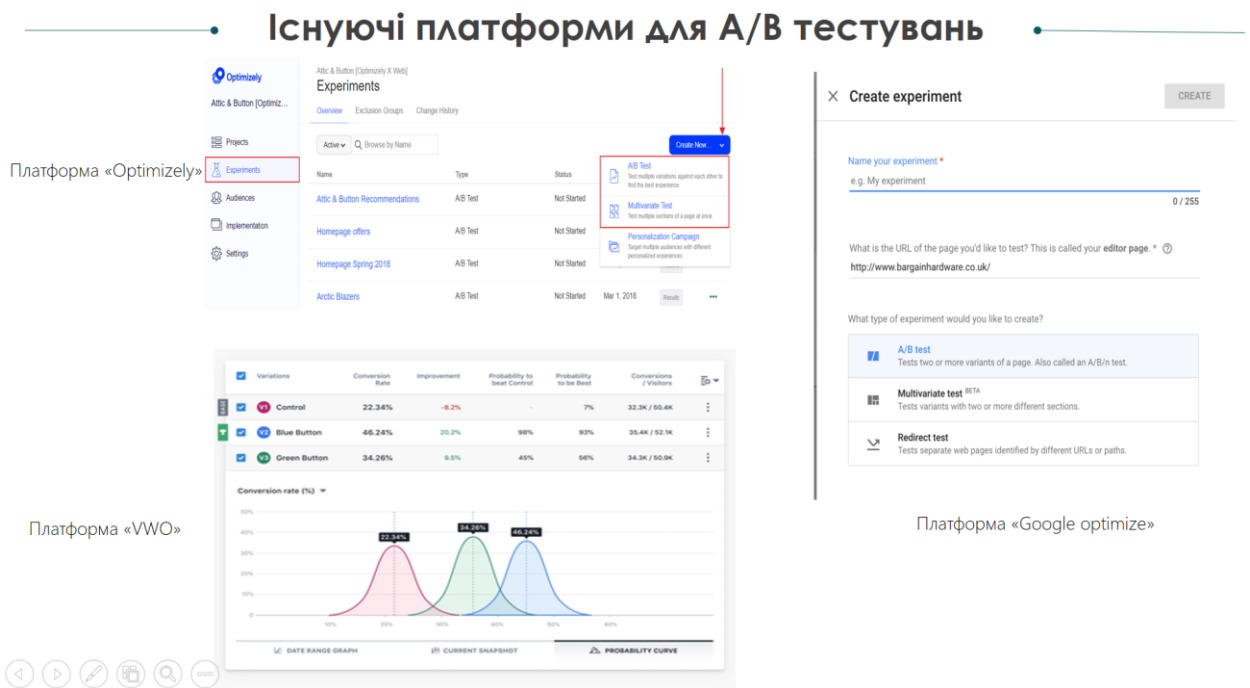
Плакат 5 – Практичне значення



Плакат 6 – Існуючі методи користувацьких експериментів

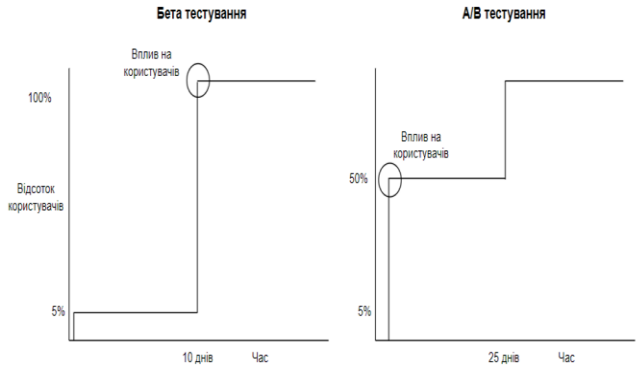


Плакат 7 – Існуючі платформи для A/B тестувань

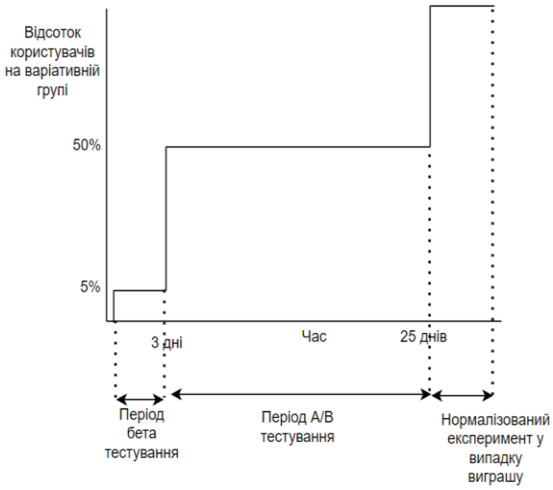


Плакат 8 – Метод поєднання A/B та Бета тестування

Метод поєднання A/B та Бета тестування

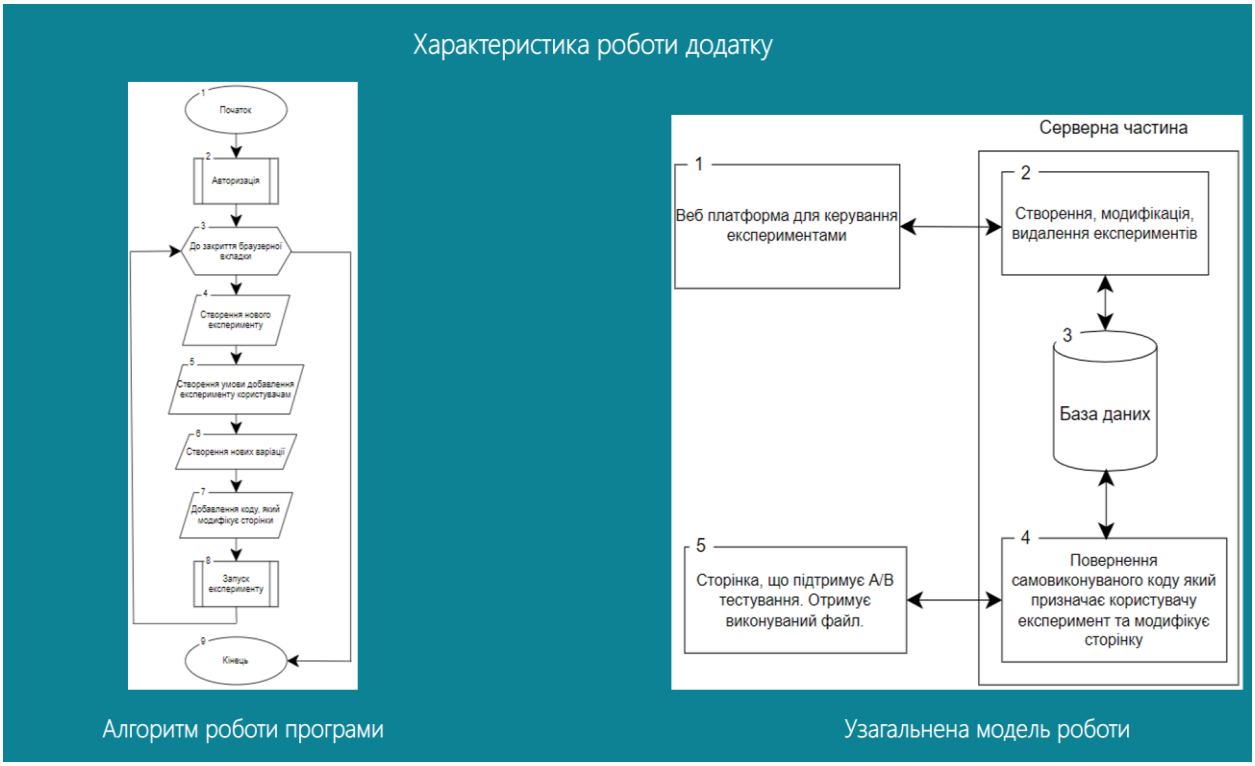


Приклад впливу змін
на користувачів у методах Бета та A/B тестування

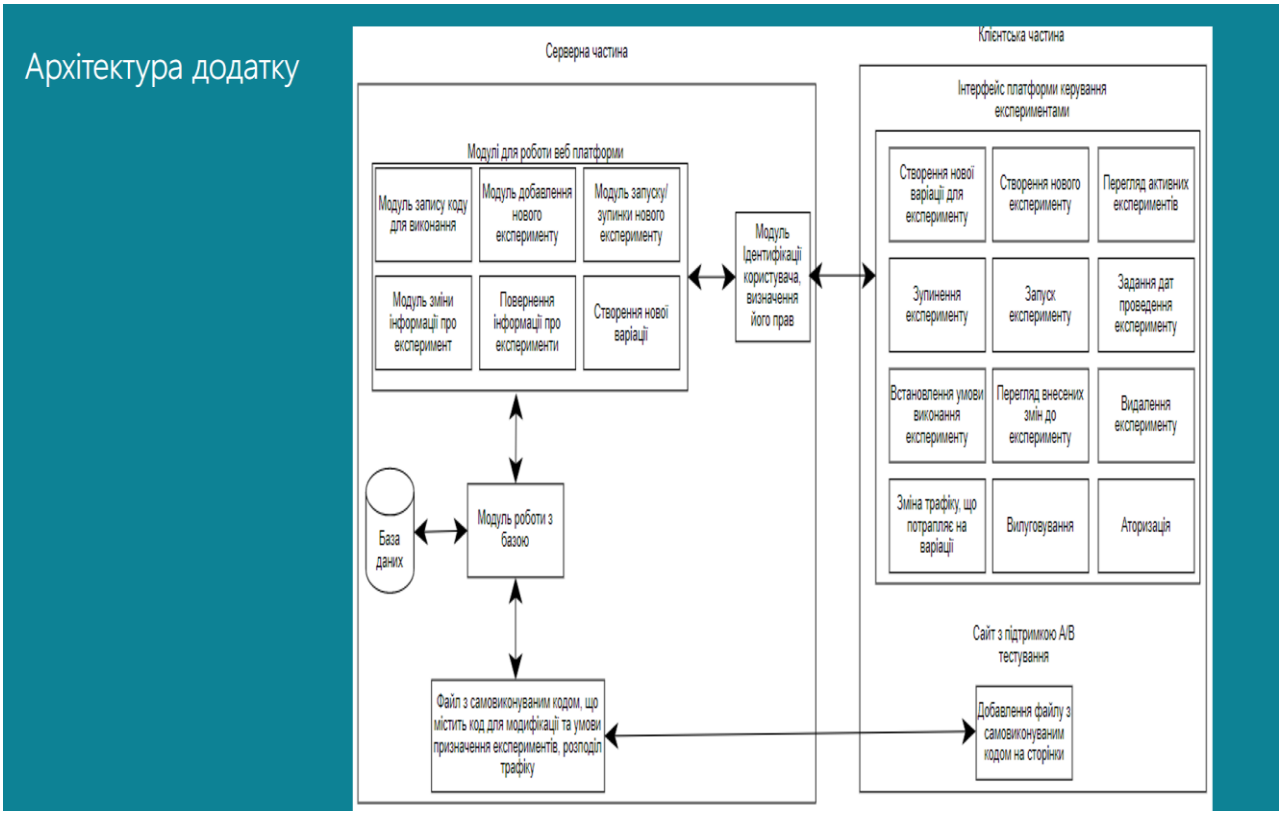


Приклад впливу змін
на користувачів у методі поєднання
A/B з Бета тестуванням

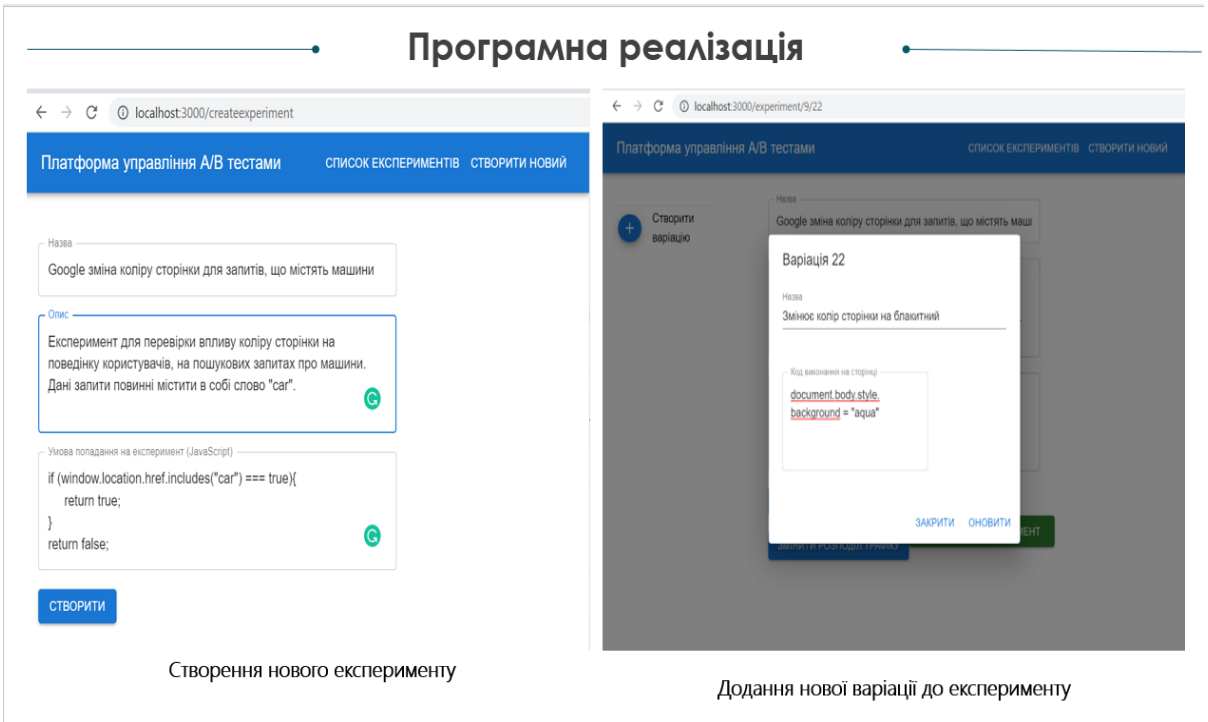
Плакат 9 – Характеристика роботи додатку



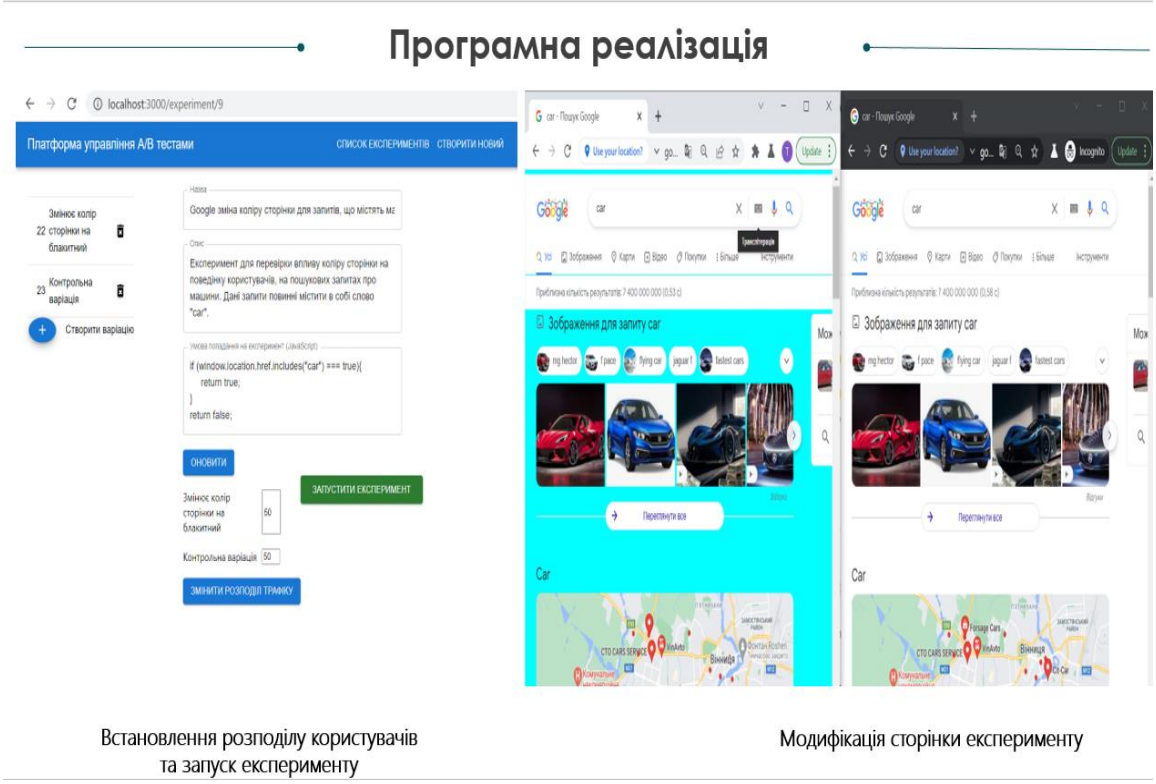
Плакат 10 – Архітектура додатку



Плакат 11 – Програмна реалізація



Плакат 12 – Програмна реалізація



Плакат 13 – Результати роботи