ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»
Навчально-науковий інститут заочно-дистанційного навчання
Форма навчання заочна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)
«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕСТУВАННЯ СТУДЕНТІВ»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Лутченко Олена Євгеніївна
_____«____»_____ 2023р.
(підпис)
Науковий керівник к. ф.-м. н., доцент, Черненко Оксана Олексіївна
_____«____»_____ 2023р.
(підпис)
Рецензент

ПОЛТАВА 2023

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ПЛАН

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ПЛАН

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ПЛАН

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ПЛАН

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ПЛАН

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ПЛАН

РЕФЕРАТ

Записка: 85 с., 4 рис., 7 таблиць, 5 додаток, 43 джерел

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕСТУВАННЯ СТУДЕНТІВ.

Об’єкт розробки – розробка програмного забезпечення для тестування студентів.

Мета роботи – дослідження та вдосконалення рівня знань здобувачів освіти.

Методи дослідження – в процесі виконання кваліфікаційної роботи були використані наступні методи: теоретичний, групування, системний аналіз, аналіз та ситнез, абстрагування, порівняння.

Було проведено огляд схожого відомого програмного забезпечення та виділено основні переваги та недоліки. За результатами дослідження сформульовано та визначено основні завдання для створення програмного продукту.

Здійснено програмну реалізацію що відповідає наступним завданням:

— створення шаблонів тестів з використанням різних тематик тестування;
— інтерактивне тестування якості засвоєння матеріалу;
— динамічне формування звітів з результатами тестування рівня знань здобувачів освіти;
— загальна статистика тестування.

За результатами тестування програмне забезпечення процює корректно та відповідає поставленим завданням.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
ПЗ	Програмне забезпечення
ПП	Програмний продукт

ВСТУП

Актуальність кваліфікаційної роботи: на даний момент широкою увагою користуються різноманітні тести, які слугують для збору різної інформації, статистичних даних, для підведення якихось підсумків чи просто в розважальних цілях. Тому розробка програмного забезпечення для створення тестів буде досить актуальною, адже це полегшить процедуру створення тестів та аналізу результатів тестування.

Метою роботи: є створення веб-ресурсу – місця в інтернеті, де користувач може створити власний тест та проаналізувати результати його проходження.

Об’єктом дослідження в роботі є технології та засоби розробки веб-сторінок, а саме веб-ресурсів.

Предметом дослідження є розробка веб-ресурсу в мережі інтернет.

Методи дослідження: в процесі виконання кваліфікаційної роботи були використані наступні методи:

Теоретичні – порівняльний аналіз підходів до створення ігрових додатків з метою детального аналізу їх структури.

Групування – для визначення залежності одних показників від інших, утворення однорідних груп на основі розподілу сукупності на окремі частини або об’єднання досліджуваних одиниць у часткові сукупності за суттєвими для них ознаками.

Системний аналіз – для деталізації і розчленування на окремі важливі складові частини, що надає можливості з аналізу, прогнозування, проектування прийняття рішень в складних системах різної природи на основі системної методології.

Аналіз і синтез – для визначення особливостей процесу візуального оформлення програмного продукту.

Абстрагування – для формулювання узагальнених висновків на основі системного аналізу та синтезу теорії і практики.

Порівняння – для зіставлення даних у динаміці, використовуються з метою встановлення логічних закономірностей які впливають на досліджувані

об’єкти або явища, і пошуку переваг та вразливостей які можуть виявлятися під впливом факторів.

Обсяг та структура роботи. Кваліфікаційна робота складається з вступу, 5 розділів, висновків, списку використаної літератури та додатків. Вона викладена на 85 сторінках, ілюстрована 7 таблицями та 4 рисунками.

1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ЗАВДАННЯ

1.1. Поняття веб-ресурсу

Веб-ресурс – це сукупність інтегрованих засобів технічного і програмно-апаратного характеру, а також інформації, призначеної для публікації у Всесвітній павутині. [1] Веб-ресурс може містити інформацію в текстовій, графічній та мультимедійній формі. Кожен веб-ресурс повинен мати унікальну адресу, яка дозволяє знайти його в мережі.

Поняття «веб-ресурс» частіше використовується в спеціальній лексиці, оскільки його етимологія належить до спеціальної комп'ютерної термінології. Ресурсом, або системним ресурсом, в інформаційних технологіях прийнято називати фізичний або віртуальний компонент обмеженою доступності в комп'ютерній системі. [1]

Тобто, будь-який пристрій в комп'ютері можна розглядати в якості ресурсу. До ресурсів також прийнято відносити не тільки фізичні компоненти (елементи пам'яті і т.п.), але і віртуальні, наприклад, файли. Саме до віртуальної частини ресурсів входить ідея веб-ресурсів. В епоху «молодого інтернету» в мережі шукали документи, а точніше файли, які мали спеціальну адресу. Згодом під ресурсом в інтернеті стали також розуміти і пов'язану в мережу всю інформаційну систему (наприклад, електронні бібліотеки тощо).

При використанні термінів в інтернеті термін «ресурс» отримав схожі характеристики (ідентифікацію та позначення, адресацію і технічну обробку). Однак була й певна специфіка, яка викликала тривалі дискусії серед фахівців. Найвідомішим сюжетом цієї дискусії була суперечка про класифікацію веб-ресурсів. Однак в дійсності, приводів для суперечок було більше. Крім технічних питань фахівці обговорювали соціальні, мовні та навіть філософські аспекти.

Подальший розвиток інтернету призвів не тільки до збільшення кількості веб-сайтів, але і до різноманітності їх функцій і призначення. [1]

Веб-ресурси зазвичай застосовуються розробниками для розширення програм за допомогою файлів, які використовуються для веб-розробок. Як користувачу програми вам може знадобитися керувати веб-ресурсами, які надає розробник або дизайнер.

Веб-ресурси – це віртуальні файли, що зберігаються в системі. Кожен веб-ресурс має унікальне ім'я, яке можна додати до URL-адреси для отримання доступу до файлу. [2] Це можна використати таким чином. Якщо є доступ до фактичного веб-серверу, на якому виконується веб-програма, файли можна копіювати на цей сайт. Але з більшістю онлайн-служб цього зробити не можна. Натомість для завантаження файлів можна використовувати веб-ресурси, щоб передавати файли до системи і надавати до них посилання, у яких міститься ім'я файлу, – так, ніби скопіювали їх як файли на веб-сервер.

Оскільки веб-ресурси – це дані в системі, які пов'язані з рішеннями, їх можна перемістити до різних організацій, експортуючи їх як частину рішення та імпортуючи рішення в різні організації. [2]

1.2. Поняття веб-додатку

Веб-програма – це будь-яка комп'ютерна програма, яка виконує певну функцію, використовуючи веб-браузер як свого клієнта. Програма може бути настільки ж простою, як дошка оголошень або контактна форма на веб-сайті або як складна, як текстовий процесор або багатофункціональний додаток для мобільних пристроїв, завантажений на ваш телефон.

"Клієнт" використовується в середовищі клієнт-сервер для позначення програми, яку людина використовує для запуску програми. Середовище клієнт-сервер – це той, в якому кілька комп'ютерів обмінюються інформацією, такою як введення інформації в базу даних. "Клієнт" – це програма, яка використовується для введення інформації, а "сервер" – це програма, яка використовується для зберігання інформації.

Веб-програма звільняє розробника від відповідальності за створення клієнта для певного типу комп'ютера або конкретної операційної системи, тому кожен може використовувати додаток разом, оскільки вони мають доступ до інтернету. [3] Оскільки клієнт запускається в веб-браузері, користувач може використовувати IBM-сумісний або Mac. Вони можуть працювати під керуванням Windows XP або Windows Vista. Вони навіть можуть використовувати Internet Explorer або Firefox, хоча для деяких програм потрібен певний веб-браузер.

Веб-користувачі зазвичай використовують комбінацію сценарію на сервері (ASP, PHP та ін.) та сценарій на стороні клієнта (HTML, Javascript та ін.) для розробки програми. Сценарій на стороні клієнта стосується презентації інформації, тоді як серверний скрипт займається всіма важкими справами, такими як зберігання та отримання інформації.

Перші основні веб-додатки були відносно простими, але наприкінці 90-х років спостерігалося поштовх до більш складних веб-програм. В даний час мільйони людей використовують веб-додатки, щоб подати свої податки на прибуток в інтернеті, виконувати онлайн-банківські завдання, підтримувати зв'язок з друзями та близькими та багато іншого.

Більшість веб-програм засновані на архітектурі клієнт-сервер, в якій клієнт вводить інформацію, поки сервер зберігає та витягує інформацію.

Протягом останніх декількох років було великим поштовхом для створення веб-додатків для функцій, які зазвичай не потребують сервера для зберігання інформації. Текстовий редактор, наприклад, зберігає документи на вашому комп'ютері та не потребує сервера.

Веб-застосунки можуть забезпечити таку ж функціональність і використовувати переваги роботи на різних платформах. Наприклад, веб-додаток може працювати як текстовий процесор, зберігати інформацію в хмарі та дозволяти "завантажувати" документ на особистий жорсткий диск.

G Suite (раніше Google Apps), Microsoft Office 365 – це ще кілька прикладів новітнього покоління веб-програм. Мобільні додатки, які підключаються до

інтернету (наприклад, програма Facebook, додаток Dropbox або додаток для онлайн-банкінгу), також є прикладами того, як веб-додатки розробляються для все більш популярного використання мобільної мережі. [3]

1.3. Класифікація та види веб-ресурсів

Мережеві інформаційні ресурси – це веб-ресурси, що зберігаються в «світовій павутині» і доступні через телекомунікаційні мережі. Мережеві ресурси можливо розподілити на: повнотекстові видання, як аналоги традиційних публікацій, що існують тільки в електронному середовищі; бібліографічні, фактографічні та повнотекстові бази даних – електронні ресурси, що мають пошукові засоби; програмні продукти – всі вони можуть бути подані як на переносних носіях, так і в мережі.

Світовими електронними ресурсами являються розподілені комплекси соціально-значущих документів за всіма галузями знань, в яких відображаються істотні відомості про минуле, сучасне та майбутнє країни, світу, соціуму. Вони орієнтовані на задоволення інформаційних потреб різних цільових груп користувачів – від представників міської спільноти до жителів різних регіонів з широким колом інтересів. Це джерела інформації, що створюються в електронному середовищі різними українськими та світовими організаціями, а також, найбільш інформативні джерела іноземного виробництва будь-якою мовою. [4]

Виробниками мережевих ресурсів є центри та банки інформації, інформаційні служби підприємств, інформаційні агенства, відділи аналітики та прогнозування, архіви, музеї, бібліотеки та інші організації. [4]

Сьогодні не існує чітко визначеної та науково-обгрунтованої класифікації веб-ресурсів, яка необхідна для вирішення питань їхньої каталогізації, зберігання, обслуговування користувачів. Тому доцільно використовувати також ознаки класифікації електронних документів, що характеризують особливості доступу до них. Таким чином, можна виділити 2 групи

документних ресурсів, що складаються з мережевих локальних документів і мережевих віддалених документів. Мережеві видалені документи можна в свою чергу розподілити за режимом доступу на мережеві віддалені документи відкритого доступу та використання і мережеві видалені документи обмеженого доступу і використання.

З точки зору ресурсів, що використовуються для надання інформаційного обслуговування користувачів, розрізняють обслуговування з використанням власного фонду та довідково-пошукового апарату та послуги з використанням сукупних інформаційних ресурсів суспільства (мережевих електронних інформаційних ресурсів). [4]

1.4. Веб-розробка

Веб-розробка – процес створення веб-сайтів або веб-додатків. Основними етапами процесу є веб-дизайн, верстка сторінок, програмування для веб на стороні клієнта і сервера, а також конфігурування веб-сервера. [5]

Цей вид діяльності потребує постійного розвитку і пошуку нових рішень. Мови програмування постійно вдосконалюються, оновлення з'являються щомісяця, якщо не щодня. І завжди потрібно тримати руку на пульсі, тому що є ризик безнадійно відстати, з огляду на сучасні темпи розвитку. Розробка для програміста або дизайнера, це як творчість для художника, а кожен створений сайт – це предмет гордості. Хоча просто захопленням веб-розробку теж не назвеш. Працюючи в колективі і спілкуючись з клієнтами, розробник – це перш за все фахівець. Його завдання, не лише писати код або малювати дизайни, але створювати продукт, який буде задовольняти вимоги клієнта і потреби кінцевого користувача. Для того, щоб всі сторони були задоволені, існують загальні стандарти веб-розробки, а процес чітко розділений на етапи, які повинні бути реалізовані в рамках створення проекту з нуля. [6]

1.5. Основні етапи веб-розробки

АНАЛІЗ БРИФУ І НАПИСАННЯ ТЕХНІЧНОГО ЗАВДАННЯ

Бриф – це документ, свого роду анкета, яку заповнює замовник. У брифі клієнт вказує побажання щодо дизайну, функціоналу сайту або програми, а також описує інші деталі проекту зі свого боку. На основі брифу менеджерами ІТ-компаній складається технічне завдання для розробників. [6]

РОЗРОБКА МАРКЕТИНГОВОЇ СТРАТЕГІЇ І СТРУКТУРИ МАЙБУТНЬОГО РЕСУРСУ

Цей етап є дуже важливим. Здійснюється він паралельно з першим. Веб-сайт, як правило, це лише частина маркетингової стратегії і її інструмент. Для того, щоб він повністю виконував свої функції, перш ніж приступати до розробки, потрібно проконсультуватися з маркетологами, зокрема з SEO-спеціалістом.

РОЗРОБКА ДИЗАЙНУ

На цьому етапі дизайнер промальовує сторінки ресурсу, починаючи з головної. Якщо говорити про дизайн сайту, то раніше спочатку створювали макети десктопної версії, але часи змінюються. Більшість користувачів для виходу в інтернет використовують мобільні пристрої, тому дуже важливо, щоб сайт не лише добре виглядав, але і правильно працював на смартфонах. Так недавно з’явилася технологія Mobile First (дослівно перекладається як “спочатку мобільний”), яка передбачає створення мобільної версії сайту в першу чергу. Незмінною вимогою до дизайну є “дружній” інтуїтивно зрозумілий інтерфейс, взаємодіючи з яким користувач ресурсу легко і швидко знаходить всю необхідну інформацію.

ПРОГРАМУВАННЯ

Front-end розробники «оживлюють» дизайн, перетворюючи макети в інтерактивні веб-сторінки. Потім back-end фахівці приступають до написання серверної частини сайту або програми. Як правило, їх роботу не бачиш оком, але відчуваєш під час використання тих чи інших функцій ресурсу, створюючи акаунт або здійснюючи покупку онлайн, адже всі ваші запити обробляються

на стороні бекенд. Серверну частину індивідуально під кожен сайт пишуть не завжди. Існують і готові рішення – різні системи управління контентом (CMS). Залежно від типу сайту підбирають оптимальну CMS, за допомогою якої і працює ресурс. У такому випадку фахівці налаштовують систему під конкретний сайт.

НАПОВНЕННЯ САЙТУ ІНФОРМАЦІЄЮ

Наповнення – окремий від загального процесу розробки етап. Цим займається контент-менеджер, власник сайту або IT-компанія.

РОЗМІЩЕННЯ РЕСУРСУ В ІНТЕРНЕТІ

На цьому етапі компанія-розробник допомагає клієнту вибрати хостинг (місце в інтернеті) і отримати домен (адреса ресурсу) – дві речі, необхідні для розміщення сайту. Після чого переносять його на цей хостинг.

ТЕСТУВАННЯ

Ресурс проходить ручне тестування на відповідність макетів дизайну, швидкість роботи, оптимізацію під мобільні пристрої і відображення в різних браузерах на кожному етапі. Фінальна перевірка якості – це оцінка готового продукту з точки зору, як програміста, так і користувача.

ПРОСУВАННЯ РЕСУРСУ

Цей етап не входить в основний процес розробки, але є дуже важливим в епоху цифрового маркетингу. На цьому етапі SEO-фахівці, SMM-менеджери, таргетологи займаються комплексним просуванням ресурсу: аналізують його роботу, складають стратегію внутрішньої і зовнішньої оптимізації та втілюють її в життя. [6]

1.6. Основні відмінності веб-додатку від веб-сайту

ІНТЕРАКТИВНІСТЬ

Перша відмінність – це різні степені взаємодії із сторінкою. У той час, як веб-сайт містить тексти та візуальний контент, з яким користувач не може

взаємодіяти, веб-додатки надають користувачеві можливість не лише читати, а й змінювати, додавати інформацію на сторінках.

Інтернет-магазин, який дозволяє користувачам купувати товари, шукати їх у каталозі, можна назвати веб-додатком. Ще один цікавий приклад – це соціальні мережі. Вони включають у себе функції блогу, чати, контент на вибір користувача та можливість ділитися цим контентом.

Сьогодні велика кількість веб-сайтів мають інтерактивність. Тому що користувачеві це подобається. Для цього власники веб-сайтів додають на свої веб-сайти невеликі веб-програми.

На сайті деяких ресторанів є доступ до Google Maps, який допомагає користувачеві знайти дорогу до ресторану. У той ж час, більшість сайтів більш інформативні ніж інтерактивні. Таким чином відвідувачі сайтів більш захоплені переглядом, читанням або прослухуванням інформаціїї. А відвідувачі веб-додатку направлені на взаємодію з користувачами. [7]

ІНТЕГРАЦІЯ

Інтеграція – це процес, що містить об'єднання простихкомпонентів у одному складному. [7] Розробники можуть інтегрувати веб-додатки та сайти з програмами, включаючи ERP, CRM. Однак у більшості випадків інтеграція відбувається саме з веб-додатками, тому що їх складним функціям дуже часто потрібна інформація з інших систем. Найпопулярніший вид інтеграції в електронній комерції – це інтеграція веб-додатків із системою управління взаємодією з клієнтами (CRM). Це допомагає зберігати дані про покупців, інформацію про замовлення та покращити продажі в цілому. Завдяки інтеграції, інформація про користувачів, автоматично збирається та зберігається в CRM системі. Така інтеграція може дозволити команді відділу продажів дізнатися більше про поведінку клієнтів і ефективніше працювати з негативними відгуками. Доступна можливість взаємодії з інформацією про клієнта може принести збільшення в продажі та вдосконалити процес роботи в інтернет-магазині.

У деяких випадках власники веб-сайтів використовують інтеграцію з CRM для того, щоб представити відвідувачам більше персоналізованого контенту. Але, на відміну від веб-додатків, подібна інтеграція з веб-сайтами – це скоріше опціональна функція, а не частина основного функціоналу.

АВТОРИЗАЦІЯ

Цей процес включає в себе введення користувальницьких даних для отримання доступу до веб-сайтів або систем. Ця функція важлива для систем, які вимагають особисту інформацію про клієнтів. На цьому етапі важливо приділити особливу увагу безпеці. Дуже важливо мінімізувати вірогідність доступу до особистих данних користувача постороннім людям.

На відміну від від сайтів, веб-додатки частіше вимагають авторизації, тому що вони пропонують користувачам більше можливостей, ніж веб-сайти. Наприклад, при використанні соціальних мереж, системи попереджують вас про ненадійність пароля. Ігнорування подібних повідомлень може привести до того, що хакери можуть отримати доступ до вашого аккаунту.

Більшість інформаційних сайтів використовують авторизацію. У деяких випадках авторизація використовується для того, щоб дати більше можливостей, які недоступні неавторизованим користувачем. Якщо незареєстровані користувачі можуть лише переглядати статті, зареєстровані користувачі можуть залишити коментарі, ділитися статтями у соціальних мережах та ін. Це також відмінне рішення для блокування спаму. Таким чином, авторизація необхідна як для веб-сайтів, так і для веб-додатків. Але в цей ж час авторизація потрібна веб-додаткам в цілях безпеки інформації.

У сучасному світі інтернету майже не залишилось сайтів, які б не мали жодних інтерактивних елементів. І навпаки, багато веб-додатків часто включають у себе пошук інформації. Тим не менш, веб-сайти як і раніш являються інформаційними джерелами, а веб-додатки залишаються користувальницькими інструментами. [7]

Можна сказати що всі веб-додатки є веб-сайтами, та не всі веб-сайти є веб-додатками. Те що робить їх такими унікальними, так це той факт, що

неможливо розробити веб-додаток з чистого листа самостійно. У будь-якому разі доведеться використовувати не тільки звичні мови розмітки та програмування, як HTML, CSS та JavaScript, але й цілі бібліотеки технологій від сторонніх розробників. Є велика кількість компаній, які сконцентровані на створені подібних технологій для стартапів і малого бізнесу.

1.7. Архітектура та принципи роботи типового веб-додатку

Веб-додаток являється клієнт-серверним, в якому клієнтом виступає браузер, а сервером веб-сервер. Логіка веб-додатку розподілена між сервером та клієнтом. Збереження даних переважно здійснюється на сервері. Обмін інформації проходить у мережі інтернет.

В веб-додатках існує дві програми, працюючі одночасно: код, який знаходиться на сервері і відповідає на HTTP запити; код, який знаходиться в браузері і реагує на введення даних користувачем.

Серверний код не видимий для клієнта, може відповідати тільки на HTTP запити конкретного URL, а не на будь-який тип користувацького введення. Зазвичай цей код написаний наступними мовами програмування та фреймворками: JavaScript(Node.js), Python(Django), PHP, Ruby On Rails, Java, C# та інші. [37]

Клієнтський код розбирається браузером користувача. На відміну від серверного він може реагувати на користувацькі введення, доступний для перегляду та редагування користувачеві у повному обсязі. Не може читати файли сервера напряму, повинен звертатися з сервером через HTTP запити. Мови програмування: HTML, CSS, JavaScript.

Технологічний стек представляє собою: операційну систему (файлову систему). Частіше всього це Linux; веб-сервер (Apache, NGINX, IIS та ін.); база данних (MySql, MongoDB, MS Sql, Oracle, PostgreSql, Redis та ін.)

Шлях запиту:

— DNS;
— HTTP, SPDY;
— Front Server (NGINX);
— Back Server or Application Server (Apache, PHP-FPM);
— PHP code, opcode;
— Response.

DNS (Domain Name System) – ієрархічна розподілена система перетворення імені хоста (комп'ютера або іншого мережевого пристрою) в IPадресу. Всі комп'ютери, підключені до інтернету, включаючи смартфони, настільні комп'ютери і сервери, що надають контент для величезних торгових веб-сайтів, знаходять один одного і обмінюються інформацією за допомогою цифр. Ці цифри називаються IP-адресами. Щоб відкрити веб-сайт в браузері, не потрібно запам'ятовувати довгі набори цифр. Досить ввести доменне ім'я, наприклад example.com, і браузер відкриє потрібну сторінку. [7]

Шлях користувацького запиту починається з DNS:

— користувач звертається до DNS для отримки IP адреси;
— DNS віддає IP адресу користувачеві;
— користувач звертається до веб-додатка по IP через TCP/IP;
— відкривається мережеве з’єднання (HTTP, SPDY).

Стек протоколів TCP/IP – це набір мережевих протоколів, що забезпечують передачу даних, які використовуються в мережах, включаючи мережу інтернет. Назва походить від двох найважніших протоколів: TCP (Transmission Control Protocol), IP (Internet Protocol). [7]

IP – протокол, який лежить у основі інтернета, його назва так і розшифровується: Internet Protocol. Наразі використовується дві версії протоколу IP:

— IPv6 – порівняно нова (опублікована в 1998); IP-адреса має розрядність 128 біт і записується у виді восьми 16-ти бітних полів, з використанням

шістнадцяткової системи числення і з можливістю скороченням двох чи більше послідовних нульових полів; приклад: 2001:db8:1234:5::1:1.

— IPv4 – опублікована в 1981р; IP-адреса має розрядність 32 біта і записується у вигляді чотирьох десяткових чисел в діапазоні 0…255 через точку; приклад: 192.0.3.43.

Кожен вузол може напряму з'єднуватись лише з вузлами своєї мережі (наприклад, підключеними до одному і тому ж сегменту інтернету), для визначення яких використовується адреса мережі – частина IP-адреси, обумовлена маскою мережі. Зв'язок з вузлами інших мереж здійснюється через допоміжні вузли – маршрутизатори.

TCP – це протокол, який базується на IP для доставки пакетів, але добавляє дві важливі речі: встановлення з'єднання – це дозволяє йому, на відміну від IP, гарантувати доставку пакетів; порти – дозволяють обмінюватись пакетами між застосунками, а не просто вузлами. Протокол TCP призначений для обміну даними – це надійний протокол, тому що:

— забезпечує надійну доставку даних, так як передбачає встановлення логічного з'єднання;

— нумерує пакети і підтверджує їх прийом квитанцією, а у випадку загублення пакету організує повторну передачу;

— ділить переданий потік байтів на частини – сегменти, і передає їх нижньому рівню, на прийомній стороні знову збирає їх у потік байтів.

З'єднання двух вузлів починається з handshake (рукопотискання):

— вузол A посилає вузлу B спеціальний пакет SYN – запрошення до з'єднання;

— вузол B відповідає пакетом SYN-ACK – згодою на установлення з'єднання;

— вузол A посилає пакет ACK – підтвердження, що згоду отримано.

Після цього TCP з'єднання рахується установленим, і додатки, працюючі на цих вузлах, можуть відправляти один одному пакети з даними. З'єднання означає, що вузли пям'ятають один одного, нумерують усі пакети, які ідуть в обидві сторони, посилають підтвердження про отримання кожного

пакету і знову посилають загублені по дорозі пакети. Для вузла А це з'єднання називається вихіднім, а для вузла В – вхідним.

HTTP (Hyper Text Transfer Protocol) – протокол прикладного рівня передачі даних (початково – у виді гіпертекстових документів в форматі «HTML», наразі використовується для передачі довільних даних). [7]

SPDY (читається як «speedy», «спіді») – протокол прикладного рівня для передачі веб-контенту. [7] Протокол розроблено корпорацією Google. За задумом розробників, даний протокол позиціонується як заміна деяких частин протоколу HTTP – таких, як управління з'єднаннями і форматами передачі даних.

Клієнт спілкується з сервером через HTTP-повідомлення. Обмін повідомленнями йде по звичайній схемі «запит-відповідь».

Кожне HTTP-повідомлення складається із чотирьох частин, які передаються у вказанному порядку:

— стартовий рядок (англ. Starting line) – визначає тип повідомлення;
— заголовки (англ. Headers) – характеризує тіло повідомлення, параметри передачі та інші дані;
— тіло повідомлення (англ. Message Body) – безпосередньо дані повідомлення. Обов'язково має відділятись від заголовків пустим рядком.

Заголовки і тіло повідомлення можуть бути відсутніми, але стартовий рядок являється обов'язковим елементом, так як вказує на тип запиту-відповіді. Виключенням є версія 0.9 протоколу, у якого повідомлення запиту містить тільки стартовий рядок, а повідомлення відповіді тільки тіло повідомлення.

Стартові рядки розрізняються для запиту і для відповіді. Рядок запиту виглядає так:

GET URI – для версії 0.9;

Метод URI HTTP/Версія – для інших версій.

Тут: Метод – назва запиту, одне слово заглавними буквами. У версії HTTP 0.9 використовується тільки метод GET;

URI визначає шлях до запитаного документу;

Версія – пара розділених точкою цифр. Наприклад: 1.0.

Стартовий рядок відповіді має наступний формат: HTTP/Версія, код стану, пояснення, де:

Версія – пара розділених точкою цифр, як в запиті;

Код стану – три цифри. По коду стану визначається подальший вміст повідомлення і поведінка клієнта;

Пояснення – текстове коротке пояснення до коду відповіді для користувача. Ніяк не впливає на повідомлення і є необов’язковим.

Методи HTTP – послідовність із будь-яких символів, окрім управляючих і роздільників, яка указує на основну операцію над ресурсом. [7] Кожний сервер забов’язаний підтримувати як мінімум методи GET і HEAD. Якщо сервер не розпізнав вказанний клієнтом метод, то він має вернути статус 501 (Not implementer). Якщо серверу метод відомий, але він не може його примінити до конкретного ресурсу, то повертається повідомлення з кодом 405 (Method Not Allowed). Окрім методів GET і HEAD, дуже часто використовується метод POST. [7]

1.8. Особливості тестування веб-додатків

Веб-додатки – це сфера, що динамічно розвивається. Не всі підходи і методи, що застосовуються для тестування класичних додатків можуть бути застосовні для тестування веб-додатків.

Веб-додаток – клієнт-серверний додаток, в якому клієнтом виступає браузер, а сервером веб-сервер, що вже є по суті двома окремими програмами, які необхідно тестувати як окремо, так і в зв'язці. [8]

Майже всі сучасні програми орієнтовані на роботу з мережею. Зберігання даних веб-додатків здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Коли користувач бачить помилку в мережевому середовищі, то часто складно точно вказати, де саме вона відбулася, і тому

режим роботи, або повідомлення про помилку, яке ми отримуємо, може бути результатом помилок, що сталися в різних частинах мережевої системи.

Маючи багато спільного з тестуванням класичних програм, тестування веб-орієнтованих додатків має свої особливості, пов'язані насамперед із середовищем функціонування. Маючи компонентні, структурні та технологічні особливості, веб-додатками притаманні особливості режимів роботи, інсталяції, запуску, зупинки і видалення, а також формування інтерфейсів. Працюючи завжди з мережею і з великою кількістю користувачів, веб-додатки мають на увазі під собою різні права доступу для різних користувачів. Логіка веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється на сервері, обмін інформацією відбувається по мережі. Одним з переваг підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформними сервісами. [8]

Особливості тестування веб-додатків.

Технологічні відмінності. Класичний додаток працює з використанням однієї або сімейства споріднених технологій. Веб-додаток працює з використанням принципово різних технологій.

Структурні відмінності. Класичний додаток "монолітний". Складається з одного або невеликої кількості модулів. Не використовує сервери БД, веб-сервери і т. д. Веб-додаток – "багатокомпонентний". Складається з великої кількості модулів. Обов'язково використовує сервери БД, Веб-сервери, сервери додатків. [36]

Відмінності режимів роботи. Класичний додаток працює в режимі реального часу, тобто відомо про дії користувача відразу ж, як тільки вони виконані. Веб-додаток працює в режимі "запит-відповідь", тобто відомо про деякий набор дій тільки після запиту на сервер. [9]

Відмінності формування інтерфейсу. Класичний додаток використовує для формування інтерфейсу користувача щодо усталені і стандартизовані

технології. Веб-додаток використовує для формування інтерфейсу технології, які стрімко розвиваються, безліч яких конкурує між собою.

Відмінності роботи з мережею. Класичний додаток практично не використовує мережеві канали передачі даних. Веб-додаток активно використовує мережеві канали передачі даних.

Відмінності запуску і зупинки. Класичний додаток запускається і зупиняється рідко. Веб-додаток запускається і зупиняється за фактом надходження кожного запиту, тобто дуже часто. Різниця в кількості користувачів. Класичний додаток: кількість користувачів, які одночасно використовують додаток, піддається контролю, обмежена і є легко прогнозованою. [10] Веб-додаток: кількість користувачів, які одночасно використовують додаток, важко прогнозується і може стрибкоподібно змінюватися в широких діапазонах.

Особливості збоїв і відмов. Класичний додаток: вихід з ладу тих чи інших компонентів відразу стає очевидним. Веб-додаток: вихід з ладу деяких компонентів надає непередбачуваний вплив на працездатність програми в цілому.

Відмінності в інсталяції. Класичний додаток – процес інсталяції стандартизований і максимально орієнтований на широку аудиторію користувачів. Не вимагає специфічних знань. Додавання компонентів програми виконується стандартним способом з використанням одного і того ж інсталятора. Веб-додаток – процес інсталяції часто недоступний кінцевому користувачеві. Інсталяція вимагає специфічних знань. Процес зміни компонент програми не передбачається або потребує кваліфікації користувачів. Інсталятор відсутній. [11]

Особливості середовища функціонування. Класичний додаток: середовище функціонування стандартизовано і не сильно впливає на функціонування програми. Веб-додаток: середовище функціонування дуже різноманітне і може мати серйозний вплив на працездатність і серверної, і клієнтської частини.

2. ОГЛЯД ТА ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ПОДІБНИХ ПРОГРАМНИХ ПРОДУКТІВ

2.1. Google Forms

Google Forms – це зручний інструмент, за допомогою якого можна легко і швидко планувати заходи, складати опитування та анкети (Рисунок 2.1). [12] Прості опитування або багаторівневі тести легко створювати на ПК і мобільних пристроях. Можна міняти шаблон на власний розсуд, вставляти зображення і відео. Статистика результатів відображається прямо в формі і за бажанням оформляється в таблицю. Крім анкетування можна збирати e-mail респондентів. З мінусів потрібно відзначити те, що вставити Google Форми на сайт вийде тільки посиланням. Але зате кількість опитувань необмежено. [13]

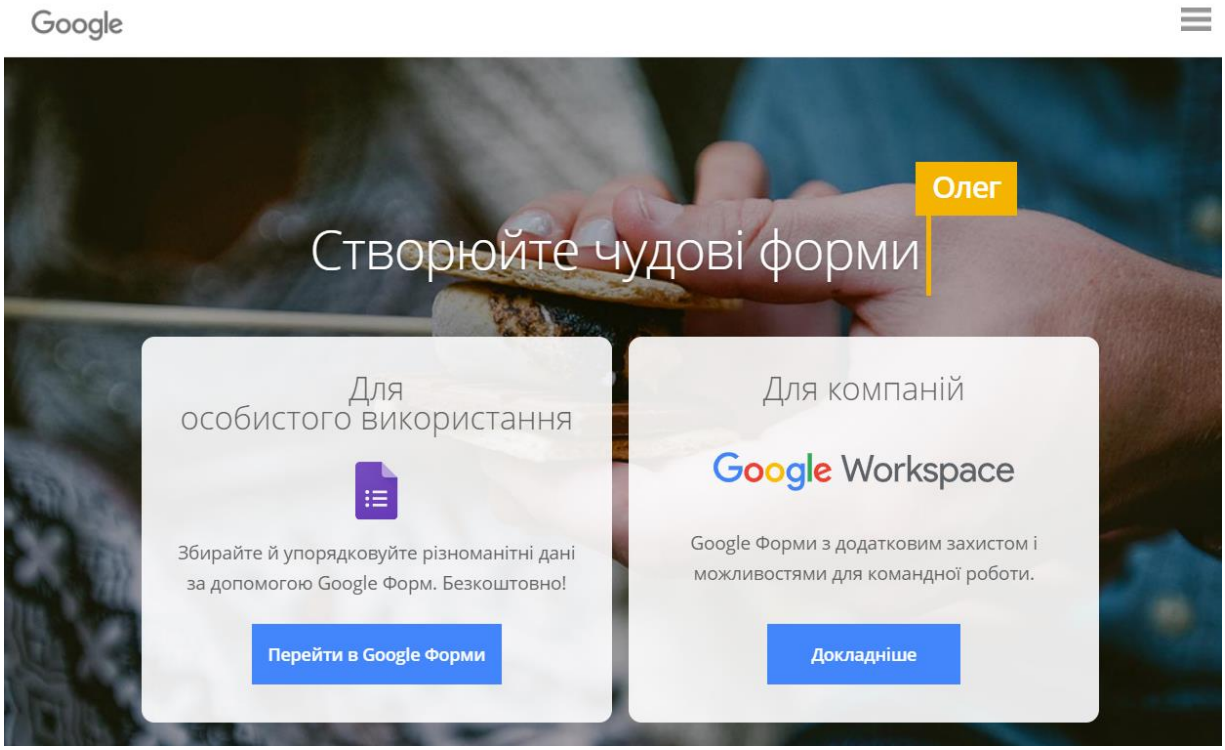


Рисунок 2.1 – Інтерфейс сторінки «Google Forms»

Переваги «Google Forms»:

— гарний та зручний інтерфейс;
— сумісність з усіма браузерами;

— легкість у використанні;
— доступність 24/7;
— індивідуальне оформлення;
— безкоштовність;
— мобільність;
— необмеженість створення опитувань.

Недоліки:

— для використання на сайті можливо вставити тільки посилання.

2.2. Online Test Pad

Online Test Pad – онлайн-тести, опитування, кросворди. Онлайн конструктор тестів, опитувань, кросвордів (Рисунок 2.2). [14] Адаптивні опитування можна оформляти для будь-яких цілей. Є 10 типів питань. Є статистика відповідей. У шаблонах можна змінювати відображення тексту, додавати зображення. Результати вивантажуються в Excel. Доступ до опитування надається за посиланням. Форму можна розміщувати на сайті віджетом або вбудовувати через html-код. [13]

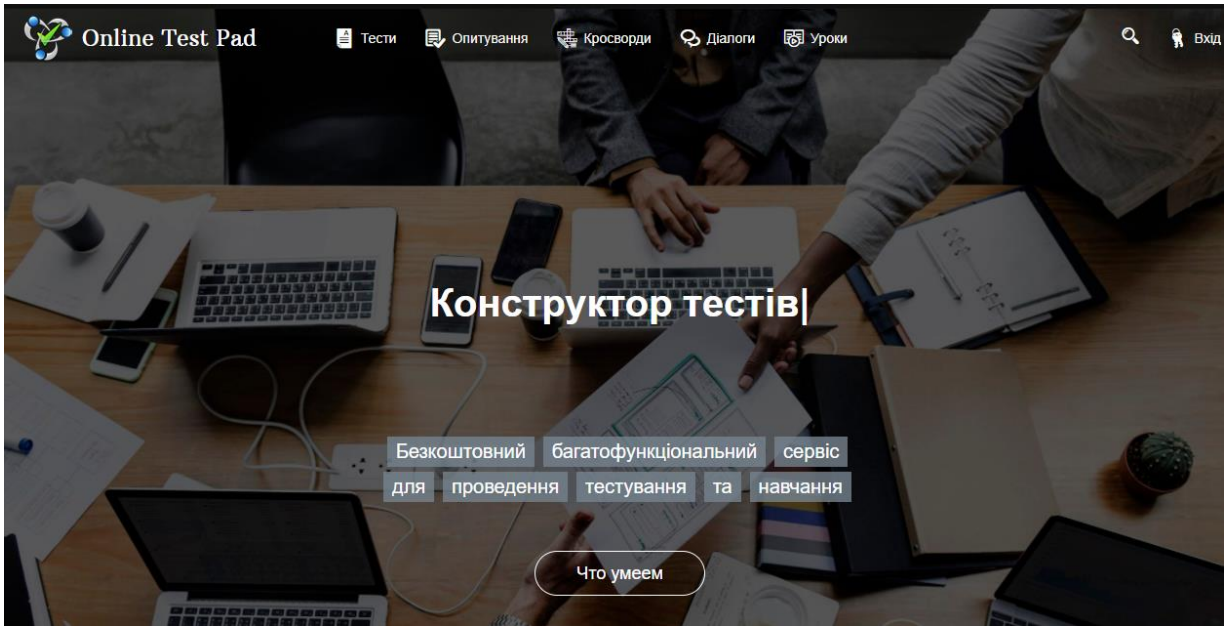


Рисунок 2.2 – Інтерфейс сторінки «Online Test Pad»

Переваги «Online Test Pad»:

— сумісність з усіма браузерами;
— легкість у використанні;
— легко вбудовуються на сайти;
— безкоштовність;
— доступний україномовний інтерфейс;
— велика функціональність.

Недоліки:

— інколи виникають технічні проблеми і сайт зависає.

2.3. Survey Monkey

SurveyMonkey – один з найпопулярніших онлайн-сервісів для проведення опитувань. З його допомогою можна отримувати інформацію про цільову аудиторію, визначати ступінь задоволеності клієнтів товарами і послугами (Рисунок 2.3) [15]. У безкоштовної версії є максимум 10 питань і 100 респондентів. На преміум-планах опитування необмежені. У сервісі є статистика відповідей і можливість колективної роботи над формами. Тема шаблону змінюється. Є експорт відповідей в .xls, .pdf, .ppt, .csv. Можна інтегрувати форму з популярними програмами. Створеним опитуванням легко ділитися в соціальних мережах або відкривати доступ за посиланням. [13]

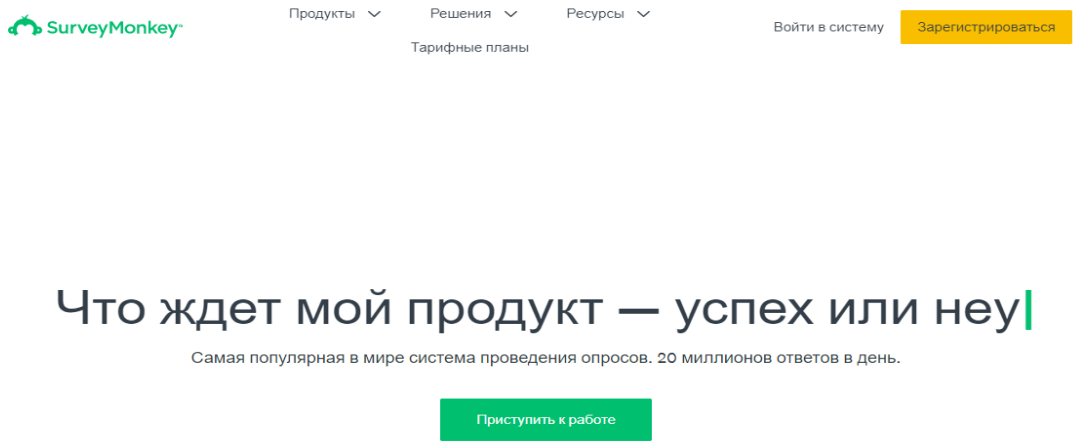


Рисунок 2.3 – Інтерфейс сторінки «Survey Monkey»

Переваги «Survey Monkey»:

— сумісність з усіма браузерами;
— легкість у використанні;
— велика функціональність;
— доступні різні види запитань та сценаріїв проведення анкетування;
— можливість фільтрації та групування результатів;
— мобільність;
— легкість у підключенні до сайту;

Недоліки:

— незручний інтерфейс;
— російськомовний інтерфейс;
— перегруженість сайту внаслідок великої функціональності;
— деякі функції доступні тільки у платній версії.

2.4. Survio

Survio – сервіс для створення опитувань, голосувань і анкет (Рисунок 2.4). Інструмент використовується як невеликими фірмами і індивідуальними підприємцями, так і великими компаніями. [16] Передбачено 100 варіацій шаблонів. У premium-версії кількість питань не лімітовано, є налаштування доступу по паролю. Налаштовується дизайн адаптований для різних пристроїв. Результати експортують в PDF, DOCx, PPTx. Для поширення анкети використовують посилання або електронну пошту. Інтерфейс дуже простий і інтуїтивно зрозумілий. У додатку Survio зможе створити анкету кожна людина, навіть той, хто її ще ніколи не створював. Користувачам не потрібно що-небудь завантажувати або встановлювати. Survio доступний з будь-якого браузера. Крім того, користувачі можуть безкоштовно розміщувати анкети на своїх сайтах або на сторінці Facebook. [17]

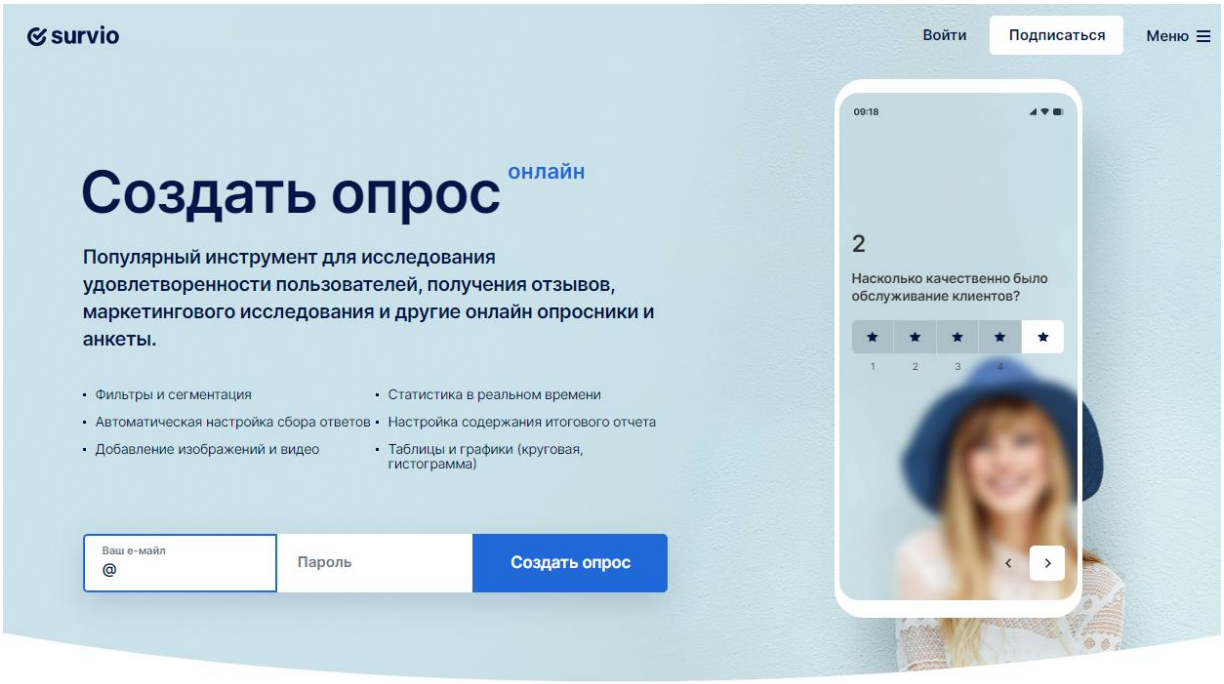


Рисунок 2.3 – Інтерфейс сторінки «Survio»

Переваги «Survio»:

— сумісність з усіма браузерами;
— легкість у використанні;
— наявність бази готових шаблонів;
— створення тестових опитувань з можливістю попереднього перегляду;

Недоліки:

— незручний інтерфейс;
— російськомовний інтерфейс;
— дуже обмежена функціональність для безкоштовної версії.

Порівняльна характеристика по ключовим пунктам розглянутих програмних продуктів наведена у додатку А.

3. ОГЛЯД ФУНКЦІОНАЛЬНОСТІ МАЙБУТНЬОГО ПРОГРАМНОГО ПРОДУКТУ

3.1. Загальні положення

Специфікація вимог до програмного забезпечення – специфікація вимог для програмної системи – це повний опис поведінки системи що розробляється. Вона включає множину прецедентів, що описують всі взаємодії, які користувачі мають з програмним забезпеченням, а також функціональними вимогами. [18]

Прецеденти – це технологія документування потенційних вимог до програмної системи. Кожен прецедент надає один чи більше сценаріїв які виражають те, як система взаємодіє з користувачем чи іншою системою щоб досягти конкретної цілі. Прецеденти також відомі як функціональні вимоги, які формуються відповідно до технічних платформ, операційних систем, апаратного забезпечення системи. [19]

Функціональність – це сукупність властивостей, які визначають спроможність ПЗ виконувати в заданому середовищі упорядковану послідовність дій для задоволення споживчих властивостей, замовлених користувачем, відповідно до вимог обробки і загальносистемних засобів. Атрибути функціональності ПЗ:

— функціональна повнота – атрибут, який показує ступінь достатності основних функцій для вирішення спеціальних завдань відповідно до призначення ПЗ;
— правильність – атрибут, який показує, як забезпечується досягнення правильних та погоджених результатів;
— інтероперабельність або сумісність – атрибути, які вказують на спроможність ПЗ взаємодіяти з іншими системами і середовищами;
— захищеність – атрибути, які вказують на можливість запобігати несанкціонованому доступу до програм і даних;

— узгодженість – атрибут, який вказує на відповідність заданим стандартам, угодам, правилам, законам і розпорядженням. [20]

Як правило, вимоги включають в себе функціональні вимоги до зовнішніх інтерфейсів, форм, апаратного забезпечення, протоколів, баз даних, захисту тощо. Сформулюємо ряд вимог яким повинен відповідати програмний продукт:

— створення інтуїтивно зрозумілого інтерфейсу;
— створення зрозумілих та зручних форм робіт з первинною інформацією;
— точність виведення даних відповідно до переходу по сторінкам;
— якісний зв’язок із базою даних;
— виведення інформації за визначеними запитами;
— точний аналіз введених даних;
— правильна обробка інформації;
— захист від некоректного введення інформації.

Діаграма варіантів використання програмного продукту наведена у додатку Б, прототип майбутнього програмного продукту представлений у додатку В.

ФУНКЦІОНАЛЬНІСТЬ РІШЕННЯ

Розроблений веб-ресурс має надавати можливість для створення та проходження тестів з обов`язковою вимогою – авторизацією користувача на сайті. На цьому сайті повинна бути функція створення запитань тесту по шаблону та перегляд списку уже створених тестів. Окрім цього, також, повинен бути доступний перегляд кількості пройдених тестів, та тестів що залишилися. Користувачем може здійснюватися проходження активних тестів, а також він може переглядати свої відповіді на вже пройдені тестування. На веб-сайті повинен бути організований пошук потрібних опитувань. Користувач повинен обов’язково входити до аккаунту та мати можливість зміни певних даних в ньому.

Для даного веб-ресурсу була виділена наступна функціональність:

Функціональність:

— авторизація;
— внесення певних змін до свого аккаунту;
— створення тестування;
— створення запитань по готовим шаблонам;
— керування створеним тестуванням;
— перегляд відповідей користувачів на створене тестування;
— проходження певних тестів;
— пошук тестування;
— перегляд історії проходження тестів та власних відповідей.

Загальний опис: Веб-ресурс для створення та проходження тестів з обов`язковою вимогою – авторизацією користувача на сайті. Створення запитань тесту по шаблону та перегляд списку уже створених тестів. Перегляд кількості пройдених тестів, та тестів що залишилися. Проходження активних тестів. Перегляд відповідей для пройдених тестів. Пошук потрібних опитувань. Вхід до аккаунту та можливість зміни певних даних в ньому.

3.2. Постановка задачі

Одним з найбільш важливих кроків програмування є постановка задачі. Вона виконує функції контракту між користувачем і програмістом (програмістами). Як юридично погано складений контракт, погана постановка задачі марна. При гарній постановці завдання як користувач, так і програміст ясно і недвозначно представляють завдання, яке необхідно виконати, тобто в цьому випадку враховуються інтереси як користувача, так і програміста.

Користувач може планувати використання ще нествореного програмного забезпечення, спираючись на знання його функціональності. Гарна постановка завдання служить основою для формування її рішення.

Постановка завдання (специфікація програми), по суті, означає точний, повний і зрозумілий опис того, що відбувається при виконанні конкретної

програми. Користувач зазвичай дивиться на комп'ютер, як на чорний ящик: для нього неважливо, як працює комп'ютер, а важливо, що може комп'ютер з того, що цікавить користувача. При цьому основна увага фокусується на взаємодії людини з машиною. [19]

На даний момент дуже багато людей проводять різні тести, які можуть слугувати як для збору різної інформації, статистичних даних, для підведення якихось підсумків чи просто в розважальних цілях.

Користувачами такого програмного продукту можуть бути підприємства, котрі зосереджені на продаж продукції, та навчальні заклади, котрі хочуть проаналізувати знання здобувачів освіти.

Отже, для розробки кваліфікаційного програмного продукту сформовано наступне завдання:

Назва: Веб-ресурс для визначення рівня знань здобувачів освіти.

Загальний опис: Веб-ресурс для створення та проходження тестів з обов`язковою вимогою – авторизацією користувача на сайті. Створення запитань тесту по шаблону та перегляд списку уже створених тестів. Перегляд кількості пройдених тестів, та тестів що залишилися. Проходження активних тестів. Перегляд відповідей для пройдених тестів. Пошук потрібних опитувань. Вхід до аккаунту та можливість зміни певних даних в ньому.

Введення даних: вхід до свого аккаунту за допомогою облікового запису Google.

Під час створення тестування вводиться загальна інформація про нього; введення запитань тесту, варіантів відповідей на нього та вибір правильної відповіді на відповідне запитання.

Під час проходження тесту вводяться вибрані відповіді на відповідні запитання.

Введення даних для користувача, який створює тест: виведення створеного тестування, та виведення даних про проходження даного тестування певним користувачем та відповіді на нього від відповідного користувача. Перегляд кількості користувачів, які пройшли даний тест.

Введення даних для користувача, який проходить тестування: виведення назначеного тесту, перелік питань цього тестування та варіантів відповідей на нього. Після закінчення тестування перегляд історії пройдених тестувань та вибраних відповідей на відповідне тестування. Виведення кількості пройдених та активних тестів.

Функціональність:

— авторизація;
— внесення певних змін до свого аккаунту;
— створення тестування;
— створення запитань по готовим шаблонам;
— керування створеним тестуванням;
— перегляд відповідей користувачів на створене тестування;
— проходження певних тестів;
— пошук тестування;
— перегляд історії проходження тестів та власних відповідей.

3.3. Опис та побудова діаграми прецедентів роботи з програмним продуктом

Існує велика кількість інструментальних засобів, які використовуються для реалізації програмних продуктів від етапу аналізу до створення програмного коду. Окремо виділяють так звані CASE-засоби верхнього рівня (upper CASE tools) і CASE-засоби нижнього рівня (lower CASE tools). Засобом, що дозволяє об’єднати ці підходи, є уніфікована мова об’єктно-орієнтованого моделювання (Unified Modeling Language – UML). До переваг мови UML можна віднести різноманітні інструментальні засоби, які як підтримують життєвий цикл програмного продукту, так і дозволяють налаштувати і відобразити специфіку діяльності розробників різних елементів проекту. [21]

Однією з діаграм, що застосовуються на етапі проектування логічної моделі програмного продукту, є діаграма прецедентів (use case diagram),

призначена для побудови на концептуальному рівні моделі того, як функціонує система в навколишньому середовищі.

Основними елементами для побудови моделі прецедентів на діаграмі є:

— Актор (actor) – елемент, що позначає ролі користувача, який взаємодіє з певною сутністю.
— Прецедент – елемент, що відображає дії, що виконуються системою, які призводять до результатів, спостережуваним акторами.

Між прецедентами в моделі можуть бути встановлені зв’язки, такі, як:

— Узагальнення (Generalization) – вказує спільність ролей.
— Включення (include) – вказує взаємозв’язок декількох варіантів використання, базовий з яких завжди використовує функціональне поведінка пов’язаних з ним прецедентів.
— Розширення (extend) – вказує взаємозв’язок базового варіанту використання і варіантів використання, які є його спеціальними випадками.

Розроблена на етапі побудови модель діаграми прецедентів може коригуватися внаслідок виявлення нових подробиць в описі під час процесу автоматизації на етапах аналізу і проектування. [19]

Діаграма прецедентів розроблена відповідно до технічного завдання та представлена у додатку Б.

Опис прецедентів:

Прецедент: АВТОРИЗАЦІЯ.

Ектор: користувач.

Передумова: перехід на сайт.

Післяумова: перехід на основну сторінку.

Сценарій:

1. Перехід на сторінку веб-ресурсу.
2. Клік по кнопці «Авторизація за допомогою Google аккаунту».
3. Перехід на основну сторінку сайту.

Прецедент: ВНЕСЕННЯ ЗМІН ДО АККАУНТУ.

Ектор: користувач.

Передумова: перехід на сторінку профілю.

Післяумова: внесення змін до свого аккаунту.

Сценарій:

1. Клік в навігаційному меню по кнопці переходу до профілю.
2. Зміна дозволених пунктів.

Прецедент: СТВОРЕННЯ ТЕСТІВ.

Ектор: користувач.

Передумова: реєстрація в системі.

Післяумова: створення тесту.

Сценарій:

1. Клік в меню по кнопці переходу до сторінки створення тестів.
2. Внесення інформації про тест.
3. Створення питань тестування за шаблонами.
4. Публікація тесту.

Прецедент: СТВОРЕННЯ ЗАПИТАНЬ ПО ШАБЛОНАМ.

Ектор: користувач.

Передумова: початок створення тесту.

Післяумова: закінчення часу можливості тестування.

Сценарій:

1. Вибір певного шаблону запитання.
2. Введення запитання.
3. Ведення варіантів відповідей
4. Вибір правильної відповіді.
5. Клік по кнопці «Створити».

Прецедент: КЕРУВАННЯ ВІДПОВІДЯМИ НА ТЕСТ.

Ектор: користувач.

Передумова: закінчення часу можливості тестування.

Післяумова: перегляд статистики.

Сценарій:

1. Клік по кнопці переходу до створеного тесту.
2. Формування таблиці з відповідями користувачів.
3. Можливість порівняння декількох груп.
4. Можливість формування статистичного графіку

Прецедент: ПЕРЕГЛЯД ВІДПОВІДЕЙ КОРИСТУВАЧІВ.

Ектор: користувач.

Передумова: закінчення часу можливості тестування.

Післяумова: аналіз результатів.

Сценарій:

1. Клік по кнопці переходу до створеного тесту.
2. Перегляд сформованої таблиці з кількістю балами відповідних користувачів.
3. Перегляд відповідей користувачів.
4. Можливість формування статистичного графіку.
5. Аналіз отриманих даних.

Прецедент: ПРОХОДЖЕННЯ ПЕВНИХ ТЕСТІВ.

Ектор: користувач.

Передумова: існування назначених тестів.

Післяумова: закінчення часу можливості тестування.

Сценарій:

1. Перехід на основну сторінку.
2. Вибір активного тесту.
3. Клік по кнопці «Пройти»

Прецедент: ПОШУК ТЕСТУ.

Ектор: користувач.

Передумова: існування назначених тестів.

Післяумова: перехід до необхідного тесту.

Сценарій:

1. Перехід на основну сторінку.

2. Введення назви чи ключового слова до пошукового поля.
3. Отримання певних тестів, що відповідають введеному запитові.
4. Вибір та перехід до потрібного тесту.

Прецедент: ПЕРЕГЛЯД ІСТОРІЇ ПРОЙДЕНИХ ТЕСТІВ.

Ектор: користувач.

Передумова: проходження певних тестів.

Післяумова: аналіз інформації.

Сценарій:

1. Перехід на основну сторінку.
2. Клік в навігаційному меню по кнопці переходу до сторінки історії пройдених тестів.
3. Автоматичний перехід на сторінку з тестами, які були пройдені даним користувачем.
4. Перегляд інформації.

Прецедент: ПЕРЕГЛЯД ВЛАСНИХ ВІДПОВІДЕЙ.

Ектор: користувач.

Передумова: перехід до сторінки історії пройдених тестів.

Післяумова: аналіз інформації.

Сценарій:

1. Вибір певного пройденого тесту.
2. Перегляд отриманої інформації.

У процесі розв'язання поставленої задачі було виконано розробку базової діаграми мови UML. Поведінка системи описується за допомогою діаграми прецедентів, яка відображає системні прецеденти, системне оточення (діючі особи або актори) і зв'язки між прецедентами й акторами. Основна задача моделі прецедентів – бути єдиним засобом, що дає можливість замовнику, кінцевому користувачеві й розробнику разом обговорювати функціональність і поведінку системи. Далі на етапі опрацювання модель доповниться детальною інформацією до існуючих прецедентів, а в разі потреби додадуться нові.

3.4. Використання БД

Firebase – це платформа розробки мобільних додатків з величезним функціоналом. [22] Починалася вона як стартап, а сьогодні її використовують при розробці кращих кроссплатформенних додатків. Головна перевага платформи в тому, що вона дозволяє розробнику не відволікатися на створення бекенду, тобто прихованої від користувача програмної частини проекту, наприклад, серверного коду. І це спрощує і прискорює створення мобільних додатків, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді. Саме зв'язка Firebase з фреймворком Flutter дозволяє програмістам створювати швидкі програми для Android і iOS, що дозволяють вирішувати найрізноманітніші завдання.

Firebase – це одне з BaaS-рішень (Backend as a Service), яке дає розробнику масу можливостей.

Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані додатки між клієнтами і зберігає їх в хмарному сховищі.

Додаток підключається до бази даних через ВебSocket, який відповідає за синхронізацію даних протягом усього сеансу.

Також Firebase виступає в якості сховища файлів. Firebase Storage забезпечує надійну завантаження і вивантаження файлів для додатка. Хмарне зберігання файлів відео, аудіо або будь-якого іншого типу підтримується Google Cloud Storage. Вміст хмарного сховища надійно захищене власною системою безпеки.

При створенні нового мобільного застосування взагалі багато уваги приділяється питанням безпеки. Створювати систему аутентифікації кожен раз з нуля досить затратно, причому витрати ці найчастіше не виправдані. Чи впорається з більшістю викликів дозволяє система аутентифікації Firebase Auth, в якій можлива аутентифікація користувача додатки по паролю і електронній пошті. Підтримує Firebase Auth також відкритий протокол

авторизації OAuth 2.0, який використовується Google, Twitter, Facebook. Система аутентифікації Firebase інтегрується безпосередньо в базу даних.

Статичні файли програми розміщуються на хостингу Firebase. Підтримується хостинг файлів JavaScript, HTML, CSS та інших. Через Cloud Functions реалізована динамічна підтримка Node.js. Передача файлів здійснюється через мережу доставки контенту з використанням захищених протоколів SSL і HTTPS.

Кросплатформне рішення Firebase Messaging дозволяє відправляти повідомлення на пристрої користувачів програми. Повідомлення можуть бути відправлені на пристрої будь-якого типу, в тому числі на ПК – як на окремі, так і на групи або на всі пристрої, на яких встановлено додаток. Також рішення підтримує можливість відправки повідомлень навіть в окремих темах. Рішення легко масштабується і дозволяє розсилати величезна кількість повідомлень в гранично короткі терміни або кастомизировать відправку повідомлень, наприклад, з урахуванням часового поясу одержувача. [22]

4. ОГЛЯД АРХІТЕКТУРИ МАЙБУТНЬОГО ПП

4.1. Планування системи

Розробка програмного продукту відбувалась в декілька етапів: аналіз, проєктування, програмування, тестування, впровадження та супровід.

Спочатку програмний продукт перебував на стадії аналізу вимог.

Було проаналізовано подібні продукти та виділено основні критерії для створення власного програмного продукту.

Визначення стилю інтерфейсу та використання зображень. На цьому етапі відбулося створення прототипу інтерфейсу (Додаток В).

На стадії проєктування відбувся етап збору інформації для наповнення сайту, створення дизайн-макету веб-сторінки та остаточного завершення макету.

На етапі програмування за допомогою HTML, CSS та JavaScript було створено саму сторінку. Вихідні коди продукту наведено в додатку Г. А також в додатку Д наведені відповідні скріншоти створеного програмного продукту.

Після цього відбувся етап тестування, де сайт було протестовано на різних інтернет-платформах.

В подальшому планується супровід програмного продукту з підтримкою його актуальності та новизни.

4.2. Інструменти програмування

JSX

JSX – це розширення синтаксису для JavaScript. Його часто використовують в React, щоб описати, як повинен виглядати інтерфейс користувача. JSX може нагадувати мову шаблонів, але з усіма перевагами JavaScript.JSX створює “React-елементи”. [40]

React використовує той факт, що логіка виводу пов’язана з іншою логікою інтерфейсу користувача: як обробляються події, як змінюється стан з часом і як дані готуються для рендерингу.

Замість того, щоб штучно відокремити технології, розмістивши розмітку і логіку в окремих файлах, React розділяє відповідальність між вільно зв’язаними одиницями, що містять обидві технології і називаються “компонентами”. [39]

React не вимагає використання JSX, але більшість людей цінують його за візуальну допомогу при роботі з інтерфейсом користувача в коді JavaScript. Він також дозволяє React показати зрозуміліші повідомлення про помилки та попередження. [23]

CSS

CSS – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних.

CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript. [34]

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі – сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки – розділення змісту сторінки (даних) та їхньої візуальної презентації. [24]

SASS

Sass – це скриптова метамова, яка компілюється в звичайні CSS-стилі. Всі, хто стикається з CSS розміром більше 500 рядків, мають справу з головним болем на тему того, як би його спростити. На жаль, з часів розробки стандартів каскадних стилів їх структура кардинально не змінювалася. Розширення SASS-файлів можуть бути .sass і .scss – це залежить від обраного синтаксису. Браузер, втім, не розуміє жодного з них, тому для взаєморозуміння потрібно використовувати компілятор. Його завдання – привести SASS в зрозумілий класичний CSS, який буде розпізнано будь-яким браузером. [34]

У мови є два основних «діалекти»: SASS і новіший SCSS. Відмінності між ними невеликі, проте порушення правил синтаксису не дозволить скомпілювати файл. У SASS-синтаксисі немає фігурних дужок, вкладеність елементів в ньому реалізована за допомогою відступів, а стильові правила обов’язково відокремлені новими рядками.

Незалежно від синтаксису, SCSS назад сумісний з CSS. Тобто будь-який CSS обов’язково буде дійсним SCSS-кодом. [25]

Переваги SASS:

— змінні;
— вкладені правила;
— доповнення;
— аргументи;
— наслідування;

JAVASCRIPT

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки. [32]

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу. [26]

REACT

React – це JavaScript-бібліотека для створення інтерфейсів користувача. React – це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду – “компонентів”. З самого початку React був спроектований так, щоб його можна було впроваджувати поступово. Тобто розробник можете додавати так мало або так багато React-у, як потрібно. [27]

TYPESCRIPT

TypeScript – мова програмування, представлений Microsoft в 2012 році і позиціонується як засіб розробки веб-додатків, що розширює можливості JavaScript. [33]

TypeScript є сумісним з JavaScript і компілюється в останній. Фактично, після компіляції, програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверної платформою Node.js. Код експериментального компілятора, який транслює TypeScript в JavaScript, поширюється під ліцензією Apache. Його розробка ведеться в публічному репозиторії через сервіс GitHub. TypeScript відрізняється від JavaScript можливістю явного статичного призначення типів, підтримкою використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах), а також підтримкою підключення модулів, що покликане підвищити швидкість розробки, полегшити читаність, рефакторинг і повторне використання коду, допомогти здійснювати пошук помилок на етапі розробки

і компіляції, і, можливо, прискорити виконання програм. [38] TypeScript це розширення мови ECMAScript 6. Додані наступні опції:

— анотації типів перевірки їх узгодження на етапі компіляції;
— вивід типів;
— класи;
— інтерфейси;
— перераховуються типи;
— домішки;
— узагальнене програмування;
— модулі;
— скорочений синтаксис «стрілок» для анонімних функцій;
— розширені можливості пошуку і параметри за замовчуванням;
— кортежі;

Синтаксично, TypeScript дуже схожий на JScript .NET, чергову реалізацію Microsoft мовного стандарту ECMA-262, що забезпечує підтримку статичної типізації та класичних об'єктно-орієнтованих можливостей мови, таких як класи, спадкування, інтерфейси і простору імен.

TypeScript є надбудовою над JavaScript. Таким чином, програма JavaScript також є правильною програмою TypeScript, і програми TypeScript можуть легко включати JavaScript. TypeScript компілює ES3-сумісний JavaScript. За замовчуванням компілюється ECMAScript 3, як переважної стандарт; також є можливість створювати конструкції, які використовуються в ECMAScript 6. [35]

З TypeScript можна використовувати існуючий JavaScript-код, включати популярні бібліотеки JavaScript, і викликати TypeScript-код, згенерований з інших JavaScript. Оголошення типів для цих бібліотек поставляються разом з вихідним кодом. [28]

4.3. Середовище розробки та система керування версіями

WEBSTORM

WebStorm – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плагінами JavaScript, котрі доступні для PhpStorm безкоштовно.

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) – в таких конструкціях підтримується коректний рефакторинг. [29]

GIT

Git – це програмне забезпечення для контролю версій, яке робить співпрацю з колегами по команді дуже простим.

Git дозволяє відстежувати кожну ревізію (зміну), яку користувач і його команда робить під час розробки програмного забезпечення. Всі члени команди використовують один репозиторій коду, який працює незалежно, а потім об’єднується. Також не потрібно постійно підключатися до сховища, тому що проект зберігається і локально на кожній машині, і віддалено (як варіант, на Github).

Git є особливим (і в значній мірі обов’язковим), тому що є можливість повернутися до будь-якої попередньої версії коду, перейти на іншу гілку і розробити певну функцію, не зачіпаючи що-небудь або кого-небудь ще, і виключити можливість пошкодження даних через їх розподіл.

Він також забезпечує легко реалізоване шифрування, а також асинхронні, нелінійні робочі процеси, тому незалежно від того, де ви знаходитесь, ви можете працювати з потрібними розділами певного проекту. [30]

GITHUB

GitHub – один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome).

Сервіс безкоштовний для проєктів з відкритим вихідним кодом, з наданням користувачам усіх своїх можливостей (включаючи SSL), а для окремих індивідуальних проєктів пропонуються різні платні тарифні плани. [31]

5. РОЗРАХУНОК ЕКОНОМІЧНИХ ПОКАЗНИКІВ

5.1. Аналіз ринку

Основними техніко-економічними та експлуатаційними характеристиками створеного програмного продукту є використання даного ресурсу для створення та проходження тестів.

Даний виріб є покращеною версією вже існуючих аналогів на світовому ринку та надає покращені можливості та розширену функціональність.

Виріб повинен задовольняти користувачів у потребі створення та проходження онлайн-тестування в даному веб-ресурсі.

Потенційним замовником чи покупцем можуть бути заклади освіти, котрі планують проводити тестування для своїх учнів чи здобувачів освіти.

Даний товар найкраще буде реалізовуватись на ринку послуг.

На розроблений веб-ресурс очікується високий попит, так як на сучасному ринку існує небагато схожих програмних продуктів.

Головним конкурентом на даний момент є сервіс зі створення онлайн-опитувань GoogleForm.

Для створеного продукту рівень цін досить високий, адже товар має унікальний інтерфейс та гарний дизайн, швидка взаємодія з користувачами та широка функціональність.

5.2. Розрахунок трудомісткості програмного продукту

Трудомісткість розробки продукту враховує певні витрати часу, такі як:

— підготовка опису завдання;

— дослідження і розробка алгоритму розв’язання задачі;

— розробка блок-схеми алгоритму;

— програмування по готовій блок-схемі;

— налагодження програми на ПЕОМ;

— підготовка документації по завданню;

— пошук помилок тощо. [41]

Загальну трудомісткість $T_{заг}$, люд-год., можна визначити за формулою (5.1)

$$T_{заг} = F_{еф} \cdot k_{скл} \cdot k_{м} \cdot k_{станд} \cdot k_{станд\ ПП} \quad (5.1)$$

де$T_{заг.}$ – загальна трудомісткість, людино-години;

$F_{еф}$ – ефективний фонд часу роботи розробника за відповідний календарний період, год. Визначається як добуток норми часу ($N_{час)}$ і коефіцієнта використання робочого часу ($k_{врч}$=0,9). Норма часу ($N_{час}$) встановлюється у людино-годинах і визначається, у свою чергу, як добуток відповідного затраченого часу на розроблення програмного продукту та кількості розробників за умови виникнення співавторства.

Відповідно до Листа Міністерства розвитку економіки, торгівлі та сільського господарства України від 12.08.2020 року № 3501-06/219 «Про розрахунок норми тривалості робочого часу на 2021 рік» при 39-годинному робочому тижні тривалість робочого місяця складає відповідно: квітень – 171,6 год., травень – 140,4 год. поточного року.

$k_{скл.}$ – коефіцієнт, що корегує проблеми контролю вхідної та вихідної інформації (складає 1,08 – 1,1);

$k_{м.}$ – коефіцієнт, що корегує використання мови певного рівня складності (мова високого рівня дорівнює 1);

$k_{станд.}$ – коефіцієнт використання стандартних програм. У процесі розробки ПП використовуються стандартні модулі та/або пакети прикладних програм, чи типові програми, тому норму часу коригують за допомогою коефіцієнта $k_{станд}$ = 0,6 – 0,8.

$k_{станд.ПП}$ – коефіцієнт розробки стандартного ПП (1,2 – 1,6).

Виконуємо розрахунки:

$F_{еф} = N_{час} \cdot k_{врч}$

$F_{еф} = 140{,}4 * 0{,}9 = 126{,}36$ год

$T_{заг} = 126{,}36 * 1{,}08 * 1 * 0{,}6 * 1{,}3 = 106{,}45$ люд-год

5.3. Розрахунок собівартості програмного продукту

Фактична або повна собівартість ПП визначається у процесі проведення калькулювання собівартості та є сумою виробничої собівартості, адміністративних витрат та витрат на збут. [41]

Виробнича собівартість $С_{вир}$, грн., визначається на підставі розрахунку поточних витрат на розробку (або функціонально-необхідних витрат на створення ПП) за формулою (5.2):

$$С_{вир} = С_{ЗП} + С_{ЄСВ} + С_{мч} + С_{зв} + С_{м} \quad (5.2)$$

де $С_{ЗП}$ – заробітна плата розробників ПП, грн.;

$С_{ЄСВ}$ – єдиний соціальний внесок, грн.;

$С_{мч}$ – вартість машинного часу, необхідного для розробки та налаштування ПП, грн.;

$С_{зв}$ – загальновиробничі (накладні) витрати (витрати на оплату праці управлінського персоналу, оплату службових відряджень, консультаційно-інформаційні витрати, ремонт і технічне обслуговування інших основних фондів, окрім ПК, оренда приміщення тощо), грн.;

$С_{м}$ – вартість матеріалів, комплектуючих, грн.

До заробітної плати розробників ПП ($С_{ЗП}$) належать витрати на виплату основної та додаткової зарплати виконавців, що обчислюються відповідно до системи оплати праці, яка застосовується в організації, включаючи будь-які види матеріальних та грошових доплат. Заробітна плата розробників, $С_{ЗП}$, грн., визначається за формулою (5.3):

$$С_{ЗП} = ЗП_{осн} + ЗП_{дод} \quad (5.3)$$

де $ЗП_{осн}$ – основна заробітна плата розробників ПП, грн.;

$ЗП_{дод}$ – додаткова заробітна плата розробників, грн.

Для розрахунків основної заробітної плати спочатку визначаємо годинну заробітну плату $ЗП_{год}$, *грн*., виходячи зі встановлених місячних окладів за формулою (5.4):

$$ЗП_{год} = \frac{О_{міс} \cdot 12}{Ф_{рч}} \quad (5.4)$$

де $О_{міс}$– місячний оклад розробника ПП, грн;

$Ф_{рч}$ – річний фонд робочого часу. Відповідно до Листа Міністерства розвитку економіки, торгівлі та сільського господарства України від 12.08.2020 року № 3501-06/219 «Про розрахунок норми тривалості робочого часу на 2021 рік» при 39-годинному робочому тижні тривалість робочого часу у 2021 році становить 1950 годин.

Основну заробітну плату $ЗП_{осн}$, грн., визначаємо за формулою (5.5)

$$ЗП_{осн} = ЗП_{год} \cdot Т_{заг} \quad (5.5)$$

де $ЗП_{год}$– годинна заробітна плата програміста, грн.;

$Т_{заг.}$ – загальна трудомісткість розробки ПП, людино-години (дивись формулу (7.1).

Додаткова заробітна плата (премії, одноразові заохочення тощо) розраховується згідно з нормативами, що встановлені підприємством (складають 10–20 % від основної зарплати. Витрати на додаткову заробітну плату $ЗП_{дод}$, грн., визначаються за формулою (5.6):

$$ЗП_{дод} = k_{дод} \cdot ЗП_{осн} \quad (5.6)$$

де $k_{дод}$ – нормативний коефіцієнт додаткової заробітної плати (0,1–0,2);

$ЗП_{осн}$ – витрати на основну заробітну плату, грн.

Виконуємо розрахунки.

(Послідовність у розрахунках є такою: $ЗП_{год}$ (5.4), $ЗП_{осн}$ (5.5), $ЗП_{дод}$ (5.6), $С_{ЗП}$ (5.3)

$$ЗП_{год} = \frac{11000 * 12}{1950} = 67{,}69 \text{ грн}$$

$ЗП_{осн} = 67,69 * 106,45 = 7205,6$ грн

$ЗП_{дод} = 0,1 * 7205,6 = 720,56$ грн

$С_{ЗП} = 7205,6 + 720,56 = 7926,16$ грн

Заробітну плату розробників ПП можна також визначити іншим способом, а саме з урахуванням того, що програміст може отримувати погодинну зарплату. У цьому випадку заробітну плату $C_{ЗП}$, грн., визначаємо за формулою (5.7):

$$С_{ЗПр.} = С_{погод} \cdot Т_{заг} \cdot 8 \cdot (1 + k_{дод}) \quad (5.7)$$

де $C_{погод}$ – погодинна заробітна плата програміста, грн.;

$T_{заг.}$ – трудомісткість розробки ПП (ІС,БД), людино-години;

$k_{дод}$ – нормативний коефіцієнт додаткової заробітної плати (0,1–0,2).

До витрат на сплату єдиного соціального внеску належать витрати, що здійснюються у порядку та розмірах, передбачених чинним законодавством України. Дані нарахування проводяться від суми основної та додаткової зарплати і складають 22% (відповідно до діючого законодавства можуть змінюватися). [42]

Витрати на сплату єдиного соціального внеску $C_{ЄСВ}$, грн., визначаються за формулою (5.8):

$$C_{ЄСВ} = k_{ЄСВ} \cdot (ЗП_{осн} + ЗП_{дод}) \quad (5.8)$$

де $k_{ЄСВ}$ – коефіцієнт витрат на сплату ЄСВ ($k_{ЄСВ} = 0,22$);

Виконуємо розрахунки $C_{ЄСВ:}$

$C_{ЄСВ} = 0,22 * 7926,16 = 1743,76$ грн

Вартість машинного часу $С_{мч}$, грн., необхідного для розробки та налаштування ПП визначається за формулою (5.9):

$$C_{мч} = C_{мг} \cdot t_{мч} \quad (5.9)$$

де $t_{мч}$– тривалість машинного часу (сума часу машинних і машинно-ручних операцій), необхідного для розробки ПП, год. У даному дослідженні тривалість машинного часу дорівнює загальній трудомісткості $T_{заг}$;

$C_{мг}$ – собівартість однієї машино-години роботи ПК при використанні власного ПК і величина орендної плати за 1 год. роботи ПК за умови, що машина орендується. $C_{мг}$, грн., визначається за формулою (5.10).

$$C_{мг} = \frac{C_{експ.}}{F_д} \tag{5.10}$$

де $C_{експ}$ – експлуатаційні поточні витрати на обслуговування ПК за визначений календарний період, грн.

$F_д$ – дійсний фонд часу роботи ПК у годинах за календарний період, що використовується для утворення інтелектуального продукту. Визначається виходячи з календарного фонду часу, зменшеного з урахуванням вихідних, святкових днів і з урахуванням втрат часу на виконання обслуговування та поточного ремонту за формулою (5.11)

$$F_д = Д_{роб} \cdot t_{ном} \cdot (1 - k_{обсл}) \tag{5.11}$$

де $Д_{роб}$ – кількість робочих днів. протягом яких експлуатується ПК при програмуванні;

$t_{ном.}$ – номінальна тривалість цілодобової роботи обладнання, $t_{ном.} = 8$ *год.*

$k_{обсл}$ – коефіцієнт, що враховує втрати часу на обслуговування ПК; $k_{обсл} = 0{,}15$

Визначаємо дійсний фонд часу:

$F_д = 18 * 8 * (1 - 0{,}15) = 122{,}4$ грн

Експлуатаційні поточні витрати $C_{експ}$, грн., на обслуговування ПК розраховуємо за формулою (5.12)

$$C_{експ} = C_{ЗПінж} + C_{ЄСВ} + C_{АВ} + C_{рем} + C_{ел} \tag{5.12}$$

де $C_{ЗПінж}$ – основна та додаткова заробітна плата спеціаліста (інженера-електронника), який обслуговує машину з урахуванням його зайнятості на обслуговування ПК, грн.. Визначається за формулою (5.13)

$$C_{\text{ЗПінж}} = \text{О}_{\text{міс}} \cdot \text{Ч}_i \cdot t_{\text{розроб}} \cdot \left(1 + k_{\text{дод}}\right) \cdot k_{зайн} \quad (5.13)$$

де $O_{міс}$ – місячний оклад інженера-електронника, який обслуговує машину;

$Ч_i$ – кількість інженерів-електронників; Рекомендується – 1 чол.;

$t_{розр}$ – час розробки продукту у місяцях (2 місяці);

$k_{дод}$ – нормативний коефіцієнт додаткової заробітної плати (0,1–0,2);

$k_{зайн}$ – коефіцієнт, що враховує зайнятість інженера у даному проекті, що може складати від 20% до 30% часу його роботи.

Відповідно $k_{зайн}$ = 0,2–0,3.

Виконуємо розрахунки $C_{ЗПінж}$:

$\text{С}_{\text{ЗПінж}} = 8900 * 1 * (1 + 0{,}1) * 0{,}3 = 2937$ грн

$C_{ЄСВ}$ – витрати на сплату єдиного соціального внеску ($k_{ЄСВ}$ = 22 % від фонду оплати праці), грн.. Визначаємо за формулою (5.14):

$$C_{ЄСВ} = k_{ЄСВ} \cdot \text{С}_{\text{ЗПінж}} \quad (5.14)$$

Виконуємо розрахунки $C_{ЄСВ}$:

$C_{ЄСВ} = 0{,}22 * 2937 = 646{,}14$ грн

$C_{АВ}$ – амортизаційні відрахування, що розраховуються від залишкової вартості ПК і норми амортизаційних відрахувань (річна норма амортизації складає 50% від балансової вартості ПК; відповідно місячна норма амортизації складає 4,16%, а для двох місяців – 8,33%.

Амортизаційні відрахування визначаємо за формулою (5.15)

$$C_{АВ} = \frac{Ц_{комп} \cdot N_a}{100} \quad (5.15)$$

Виконуємо розрахунки $C_{АВ}$:

$C_{АВ} = \frac{12000*4{,}16}{100} = 499{,}2$ грн

$C_{рем}$ – витрати на ремонт і профілактику ПК, грн.. Визначаються за формулою (5.16)

$$C_{рем} = k_{рем} \cdot Ц_{комп} \cdot n_i \quad (5.16)$$

де $k_{рем}$ – коефіцієнт поточного ремонту та обслуговування ПК, що залежить від середньостатистичного нормативу витрат на поточний ремонт і обслуговування ПК, який складає 4% ($k_{\text{рем}} = 0{,}04$);

$Ц_{комп}$ – ціна ПК, грн.;

n_i – кількість необхідних комп'ютерів.

Виконуємо розрахунки $C_{рем}$:

$$C_{рем} = 0{,}04 * 12000 * 1 = 480 \text{ грн}$$

$С_{ел}$ – витрати на оплату електроенергії, що за формулою (5.17). Відповідно до Постанови НКРЕКП від 09.12.2000 року № 2356 «Про порядок розрахунку роздрібних тарифів на електричну енергію, тарифів на розподіл електричної енергії (передачу електричної енергії місцевими (локальними) електромережами), тарифів на постачання електричної енергії за регульованим тарифом», роздрібні тарифи на електроенергію для непромислових споживачів становлять 1,68 грн. за 1 кВт–год. електроенергії.

$$С_{ел} = N_{ПК} \cdot F_д \cdot Т_{ел} \quad (5.17)$$

де $N_{ПК}$ – потужність ПК, кВт;

$Т_{ел}$ – тариф на оплату електроенергії, грн.;

Виконуємо розрахунки:

$$C_{ел} = 0{,}5 * 122{,}4 * 1{,}68 = 102{,}82 \text{ грн}$$

Розраховуємо експлуатаційні поточні витрати відповідно до (7.12):

$$C_{експл} = 2937 + 646{,}14 + 499{,}2 + 480 + 102{,}82 = 4665{,}16 \text{ грн}$$

Визначаємо вартість машино-години за формулою (7.10):

$$C_{мг} = \frac{4665{,}16}{122{,}4} = 38{,}11 \text{ грн}$$

Визначаємо вартість машинного часу за формулою (7.9):

$t_{\text{мч}} = \text{Т}_{\text{заг}} = 106{,}45$ люд-год

$C_{мч} = 38{,}11 * 106{,}45 = 4057{,}24$ год

Загальновиробничі (накладні) витрати ($\text{С}_{\text{зв}}$) – це витрати на диски, картриджі, папір для роздрукування тощо. До складу загальновиробничих витрат також можуть належати витрати на освоєння нової розробки; відшкодування зносу спеціальних інструментів і пристроїв цільового призначення тощо. [42]

Враховуючи комплексний склад загальновиробничих витрат, їх норматив можна визначити у відсотковому значенні, що складає 20%–40% від основної заробітної плати розробників ПП. Загальновиробничі (накладні) витрати визначаються за формулою (5.18), грн.:

$$\text{С}_{\text{зв}} = \text{ЗП}_{\text{осн}} \cdot k_{\text{зв}} \tag{5.18}$$

де $k_{\text{зв}}$ – коефіцієнт загальновиробничих (накладних) витрат (0,2–0,4).

Виконуємо розрахунки $\text{С}_{\text{зв}}$:

$C_{зв} = 7205{,}6 * 0{,}2 = 1441{,}12$ грн

Вартість витратних матеріалів, комплектуючих ($C_м$) рекомендовано взяти у розмірі 10 % від фонду заробітної плати ($\text{С}_{\text{ЗП}}$).

$$\text{С}_{\text{м}} = \text{С}_{\text{зп}} \cdot 0{,}1$$

$C_м = 7926{,}16 * 0{,}1 = 792{,}62$ грн

Отже, виробнича собівартість складає (за формулою (5.2):

$C_{вир} = 7926{,}16 + 646{,}14 + 4057{,}24 + 1441{,}12 + 792{,}62 = 14863{,}28$ грн

Окрім вказаних поточних витрат на розробку ПП, собівартість розробки та реалізації ПП передбачає розрахунки:

— адміністративних витрат (організаційні витрати, витрати на службові відрядження, страхування, амортизацію, опалення, освітлення, водопостачання, охорону; винагорода за професійні послуги: юридичні, аудиторські; витрати на зв’язок; витрати за послуги банку);

— витрат на збут (на рекламу та дослідження ринку: маркетинг; витрати на гарантійний ремонт і гарантійне сервісне обслуговування; комісійні

витрати; витрати, пов'язані з безпосереднім постачанням: страхування, амортизація, охорона);

— повної (фактичної) собівартості (у грошовому виразі індивідуальні витрати певного розробника ПП у даних умовах). Повна собівартість формується у процесі щоденного оперативно-технічного та бухгалтерського обліку витрат на розробку ПП, виконання робіт і забезпечення всіма матеріально-технічними, трудовими ресурсами. [41]

Адміністративні витрати складають 10–20 % від основної заробітної плати програміста. Адміністративні витрати визначаються за формулою (5.19), грн.:

$$\text{С}_{\text{адмін}} = k_{\text{адмін}} \cdot \text{ЗП}_{\text{осн}} \tag{5.19}$$

де $k_{адмін}$ – коефіцієнт адміністративних витрат (0,1–0,2).

Виконуємо розрахунок $C_{адмін}$:

$C_{адмін}$ =0,2 ∗ 7205,6 = 1441,12 грн

Витрати на збут складають 2,5–5% від виробничої собівартості. Витрати на збут визначаються за формулою (5.20), грн.:

$$C_{збут} = k_{збут} \cdot C_{вир.} \tag{5.20}$$

де $k_{збут}$ – коефіцієнт витрат на збут ($k_{збут}$=0,025–0,05)

$C_{вир}$ – виробнича собівартість, грн. (див. табл. 5.1 –7-ма строчка).

Виконуємо розрахунок $C_{збут}$:

$C_{збут}$ = 0,05*14863,28 = 743,16 грн

Перелік і склад статей калькулювання виробничої собівартості ПП установлюються підприємством-розробником. Сума за всіма наведеними вище статтями становить повну собівартість продукції.

Також потрібно встановити частку кожного елемента витрат (питому вагу) у загальній сумі собівартості продукції у відсотках.

Результати виконаних розрахунків заносимо у таблицю 5.1.

Таблиця 5.1 – Калькуляція собівартості ПП

Статті витрат	Сума, грн.	Питома вага, %
Основна заробітна плата	7205,6	42,27
Додаткова заробітна плата	720,56	4,23
Витрати на сплату єдиного соціального внеску	646,14	3,79
Вартість машинного часу, необхідного для розробки та налаштування ПП	4057,24	23,8
Загальновиробничі (накладні) витрати	1441,12	8,45
Вартість витратних матеріалів, комплектуючих	792,62	4,65
Виробнича собівартість ($С_{вир}$)	14863,28	87,19
Адміністративні витрати	1441,12	8,45
Витрати на збут	743,16	4,36
Повна (фактична) собівартість ($С_{пов}$)	17047,56	100

5.4. Розрахунок договірної ціни програмного продукту на основі вартості його розробки

Ціна ПП ($Ц_{дог}$) формується на основі економічно обґрунтованої собівартості його розробки, норми рентабельності, прибутку (певного відсотку торговельної надбавки) та податку на додану вартість (ПДВ) за формулою (5.21) або (5.22)

$$Ц_{дог} = (C_{пов} + P + m) + ПДВ \tag{5.21}$$

$$Ц_{дог} = \left[C_{пов} \cdot \left(1 + \frac{P_{норм} + m_{\%}}{100}\right)\right] \cdot ПДВ \tag{5.22}$$

де $C_{пов}$ – повна собівартість або поточні витрати на розробку ПП, грн.;

P – рентабельність (нормативний рівень $P_{норм}$ = 20% від повної собівартості), грн.;

m – торговельна надбавка (5–10% від повної собівартості), грн.;

До ціни ПП (системи, мережі) входить ПДВ (20 %), проте, якщо розробник є приватною особою, він не є платником податку на додану вартість. Тому, у цьому випадку ПДВ не нараховується.

$$Ц_{дог} = 17047{,}56 + (17047{,}56 * 0{,}2) + (17047{,}56 * 0{,}1) = 22161{,}83 \text{ грн}$$

5.5. Оцінка конкурентоспроможності ПП

5.5.1. Визначення основних параметрів базового і нового варіантів ПП

У процесі дослідження виділяються основні технічні та економічні параметри базового та нового варіантів ПП. Також наводяться додаткові функції нового ПП (Таблиця 5.2 є прикладом).

Таблиця 5.2 – Характеристика основних техніко-економічних параметрів базового та нового варіантів ПП.

Назва параметру	Варіант		Характеристика зміни параметра нового варіанта відносно базового (↑, ↓ чи =)
	Базовий (аналог)	Новий	
Вартість ПП, грн.	30000	22161,83	↑
Простота та зручність інтерфейсу, бали	8	10	↑
Кількість функцій ПП, шт.	5	5	=
Вага ПП, мбайт	25	10	↑
Час виконання запиту, мкс	18	12	↑
Можливість нарощування функціональних характеристик, бали	5	7	↑
Час відновлення системи після збою, сек	30	30	=
Кількість людей, необхідних для обслуговування, чол.	1	2	↓

Виходячи з отриманих результатів, розробка є кращою відносно базового варіанта за такими п’ятьма параметрами:

— вартість ПП;

— простота та зручність інтерфейсу;

— вага ПП;

— час виконання запиту;

— можливість нарощування функціональних характеристик;

За двома параметрами новий і базовий варіанти ПП є ідентичними (3 – кількість функцій ПП та 7 – час відновлення системи після збою), а за параметром 8 (кількість людей, необхідних для обслуговування) новий варіант продукту поступається базовому.

Соціальний ефект запропонованого програмного продукту отриманий у результаті появи додаткових функцій та зручності інтерфейсу.

5.5.2. Конструювання еталону конкурентоспроможності ПП

Еталоном є точка багатовимірного простору (вектор), що утворена за таким правилом: серед показників-стимуляторів (здійснюється позитивний вплив на конкурентоспроможність) відбираємо дані з максимальним значенням, а серед показників дестимуляторів (чинить негативний плив на конкурентоспроможність) відбираємо дані з мінімальним значенням. Визначення еталонного значення наведено в таблиці 5.3 (колонки 2, 3).

5.5.3. Розрахунок інтегрального показника конкурентоспроможності базового і нового варіантів ПП

Інтегральний показник конкурентоспроможності відносно еталона ($I_{КСП}$) визначається за формулою (5.23)

$$I_{КСП} = \sum_{i=1}^{n} \frac{q_{баз(нов)i}}{q_{еталон\ i}} \qquad (5.23)$$

де $q_{баз.(нов).i}$, $q_{еталон\ \ i}$ – величини за i-тим параметром відповідно базового, нового варіанта ПП та еталона.

Результати розрахунків заносимо до таблиці 5.3 (колонки 4, 5).

Проведені розрахунки будуть свідчити про те, що базовий та новий ПП може переважати еталон за параметрами. Результатом буде одна із наступних

умов: якщо $I_{КСПбаз} > I_{КСПнов}$, то базовий ПП перевищує новий зразок за конкурентоспроможністю, якщо $I_{КСПбаз} < I_{КСПнов}$, то поступається йому, а при $I_{КСПбаз} = I_{КСПнов}$, новий ПП знаходиться на одному рівні з базовим.

Таким чином, базовий ПП переважає за еталон параметрами на 3,86, а новий – на 5. Тобто новий ПП є безперечно конкурентоспроможним порівняно з базовим.

5.5.4. Визначення ефективності нового ПП порівняно з базовим

Загальна ефективність виробництва нового ПП порівняно з базовим визначається за формулами (5.24–5.25)

$$Еф_{заг} = Еф_{min} + Еф_{max} \tag{5.24}$$

$$Еф_{min} = \sum_{i=1}^{n} \left(1 - \frac{q_{нов\,i}}{q_{баз\,i}}\right) \tag{5.25}$$

де $q_{нов\,i}, q_{баз\,i}$ – величини за i-тим параметром відповідно нового та базового варіанту ПП.

Проте, якщо серед параметрів є такі, для яких максимальне значення є найбільш ефективним, то рівняння $\left(1 - \frac{q_{нов\,i}}{q_{баз\,i}}\right)$ набуває такого вигляду (формула 5.26)

$$Еф_{max} = \sum_{i=1}^{n} \left(\frac{q_{нов\,i}}{q_{баз\,i}} - 1\right) \tag{5.26}$$

Для розрахунку загальної ефективності визначимо відносне значення параметрів нового ПП відносно базового (Таблиця 5.3, колонка 6).

Розраховуємо ефективність окремого параметра нового ПП за формулами 5.25 і 5.26, результати розрахунків заносимо до таблиці 5.3 (колонки 7, 8):

1–й параметр: оскільки в новому продукті вдалось зменшити його загальну вартість, виграш від цього склав: 1 – 0,74 = 0,26.

2–й параметр: вдалося покращити інтерфейс (вдосконалення призвело до спрощення та зручності користування), тому виграш нового продукту складає: 1,25 – 1 = 0,25.

3–й параметр: кількість функцій розробленого ПП є такою ж, як і в аналога, тобто лишилась незмінною (1 –1 = 0).

4–й параметр: за рахунок скорочення ваги розробленого ПП отримали виграш у розмірі: (1 – 0,6= 0).

5–й параметр: вдосконалення ПП дозволило покращити (скоротити) час виконання запиту на 0,33 (1– 0,67 = 0,33).

6–й параметр: вдалось підвищити можливість нарощення функціональних характеристик, виграш склав 1,4 – 1 = 0,4.

7–й параметр: час відновлення системи у випадку збою в роботі комп'ютера залишився незмінним: 1 – 1 = 0.

8–й параметр: незважаючи на проведені вдосконалення, не вдалось зменшити кількість людей, необхідних для обслуговування і, тому програш складає: 1 – 2 = –1.

Таблиця 5.3 – Визначення параметрів конкурентоспроможності ПП та показника його ефективності (Приклад)

Назва параметра	Еталонне значення (max або min)		Інтегр. показник за варіантами		$\frac{q_{нов\,і}}{q_{баз\,і}}$	$1-\frac{q_{нов\,і}}{q_{баз\,і}}$	$\frac{q_{нов\,і}}{q_{баз\,і}}-1$
			$\frac{q_{баз\,і}}{q_{еталон\,і}}$	$\frac{q_{нов\,і}}{q_{еталон\,і}}$			
1	2	3	4	5	6	7	8
1. Вартість ПП	min	22161,83	–1,35	+1	0,74	0,26	
2. Простота та зручність інтерфейсу	max	10	–0,8	+1	1,25		0,25
3. Кількість функцій системи	max	5	+1	+1	1		0
4. Вага ПП	min	10	– 2,5	+1	0,4	0,6	

Продовження таблиці 5.3

1	2	3	4	5	6	7	8
5. Час виконання запиту	min	12	–1,5	+1	0,67	0,33	
6. Можливість нарощування функціональних характеристик	max	7	–0,71	+1	1,4		0,4
7. Час відновлення системи після збою	min	30	+1	+1	1	0	
8. Кількість людей, необхідних для обслуговування	min	1	+1	–2	2	1	
ВСЬОГО			–3,86	5		2,84	

Відповідно до цього, загальна економічна ефективність складе:

$$\text{Еф}_{заг} = 0{,}26 + 0{,}25 + 0{,}6 + 0{,}33 + 0{,}4 + 1 = 2{,}84$$

Отже, ефективність виробництва нового (розробленого) програмного продукту відносно базового (аналогу) складає 2,84 що у відсотковому вираженні становитиме 102,84%.

ВИСНОВКИ

Завданням кваліфікаційної роботи було створення ПЗ для тестування студентів. Під час розробки програмного продукту було створено дизайн-макет, спроектовано даний програмний продукт та наповнено відповідним контентом. Проаналізовано сучасні технології у створенні веб-сайтів. Проаналізовано існуючі засоби для розробки веб-сайтів, різноманітних середовищ розробки та легких редакторів. Протестовано веб-ресурс у найпопулярніших браузерах та була проведена перевірка валідації даних. В ході тестування виявлено, що сайт працює коректно. Для готового програмного продукту було проведене повноцінне тестування: перевірка гіперпосилань, перевірка відповідності текстів заданій темі, перевірка відповідності сайту індивідуальному завданню, перевірка на кросбраузерність та юзабіліті. Отже, в ході виконання роботи було продемонстровано практичні навички створення та розробки програмного продукту.

Прикладом використання програмного продукту є створення тестів для перевірки рівня знань здобувачів освіти.

Під час виконання завдання було використано навички з проектного навчання з курсу Інтернет-технології, сучасні парадигми програмування, стандарти в інформаційних технологіях та ін. Крім цього, було вдосконалено вміння створювати макети та дизайни для веб-сторінок, в також користуватися відкритими ресурсами.

Для створення даного програмного продукту використовувалось таке програмне забезпечення як WebStorm.

Також були вдосконалені та поглиблені знання і вміння в сфері веб-програмування, роботи з мовою тегів – HTML, спеціальною мовою стилю сторінок – CSS та мовою програмування JavaScript. А також покращені вміння з використання різних фреймворків.

В майбутньому планується розширення функціональних можливостей та подальший супровід створеного програмного продукту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Що таке веб-ресурс? [Електронний ресурс]. – Режим доступу: URL: https://www.glossary-internet.ru/terms/%C8/internet_resurs (дата звернення 08.06.2021). – Назва з екрану.
2. Створення та редагування веб-ресурсів для розширення застосунку [Електронний ресурс]. – Режим доступу: URL: https://docs.microsoft.com/uk-ua/dynamics365/customerengagement/on-premises/customize/create-edit-веб-resources (дата звернення 08.06.2021). – Назва з екрану.
3. Що точно таке веб-додаток? [Електронний ресурс]. – Режим доступу: URL: https://uk.go-travels.com/73084-what-is-a-веб-application-3486637-6454152 (дата звернення 08.06.2021). – Назва з екрану.
4. Веб-ресурси [Електронний ресурс]. – Режим доступу: URL: http://gymnasium82.edu.kh.ua/shkiljna_biblioteka/mediateka/internet-resursi (дата звернення 08.06.2021). – Назва з екрану.
5. Веб-розробка [Електронний ресурс]. – Режим доступу: URL: https://ukrinfosystems.com.ua/uk/telecommunication-services/веб-development (дата звернення 08.06.2021). – Назва з екрану.
6. Веб-розробка: етапи, стандарти, реальні проекти [Електронний ресурс]. – Режим доступу: URL: https://веб-systems.solutions/blog/веб-development-stages-standards-projects (дата звернення 08.06.2021). – Назва з екрану.
7. Веб-додаток «Система контролю задоволеності співробітників» [Електронний ресурс]. – Режим доступу: URL: https://er.nau.edu.ua/bitstream/NAU.pdf (дата звернення 08.06.2021). – Назва з екрану.
8. Каскадна модель [Електронний ресурс]. – Режим доступу: URL: https://ru.wikipedia.org/wiki/Каскадная_модель (дата звернення 08.06.2021). – Назва з екрану.

9. A brief Introduction to Scrum Methodology [Електронний ресурс]. – Режим доступу: URL: https://mtceduhub.com/a-brief-introduction-to-scrum-methodology (дата звернення 08.06.2021). – Назва з екрану.
10. Scrum [Електронний ресурс]. – Режим доступу: URL: https://ru.wikipedia.org/wiki/Scrum (дата звернення 08.06.2021). – Назва з екрану.
11. Блиск та злидні автоматизації тестування [Електронний ресурс]. – Режим доступу: URL: https://habr.com/ru/company/wrike/blog/321290 (дата звернення 08.06.2021). – Назва з екрану
12. Googlee Forms [Електронний ресурс]. – Режим доступу: URL: https://www.google.com/intl/uk_ua/forms/about/ (дата звернення 09.04.2021). – Назва з екрану.
13. Топ-10 найкращих конструкторів форм для онлайн-опитувань [Електронний ресурс]. – Режим доступу: URL: https://vc.ru/services/66557-top-10-luchshih-konstruktorov-form-dlya-onlayn-oprosov (дата звернення 28.05.2021). – Назва з екрану.
14. Online Test Pad [Електронний ресурс]. – Режим доступу: URL: https://onlinetestpad.com (дата звернення 09.04.2021). – Назва з екрану.
15. Survey Monkey [Електронний ресурс]. – Режим доступу: URL: https://gde-saas.ru/applications/surveymonkey (дата звернення 28.05.2021). – Назва з екрану.
16. Survio [Електронний ресурс]. – Режим доступу: URL: https://gde-saas.ru/applications/survio (дата звернення 28.05.2021). – Назва з екрану.
17. Знайомство з сервісом Survio [Електронний ресурс]. – Режим доступу: https://www.slideshare.net/madinatukasheva/survio-36340720 (дата звернення 28.05.2021). – Назва з екрану.
18. Життєвий цикл програмного забезпечення [Електронний ресурс]. – Режим доступу: URL: https://studfile.net/preview/4532867 (дата звернення 28.05.2021). – Назва з екрану.

19. Проектування інформаційних систем на основі уніфікованої мови моделювання [Електронний ресурс]. – Режим доступу: URL: https://sites.google.com/site/analizvimogdopz/lekciie/uml (дата звернення 28.05.2021). – Назва з екрану.
20. Аналіз програмного забезпечення [Електронний ресурс]. – Режим доступу:https://pidru4niki.com/1194121347734/informatika/analiz_yakosti_programnogo_zabezpechennya (дата звернення 28.05.2021). – Назва з екрану.
21. Методи архітектурного проектування [Електронний ресурс]. – Режим доступу: URL: https://osvita.ua/vnz/reports/arhitektura/25291 (дата звернення 28.05.2021). – Назва з екрану.
22. Що таке Firebase [Електронний ресурс]. – Режим доступу: URL: https://avada-media.ua/ua/services/firebase (дата звернення 07.06.2021). – Назва з екрану.
23. Вступ до JSX [Електронний ресурс]. – Режим доступу: URL: https://uk.reactjs.org/docs/introducing-jsx.html (дата звернення 09.06.2021). – Назва з екрану.
24. CSS [Електронний ресурс]. – Режим доступу: URL: https://uk.wikipedia.org/wiki/CSS (дата звернення 31.03.2021). – Назва з екрану.
25. Керівництво по SASS [Електронний ресурс]. – Режим доступу: URL: https://tokar.ua/read/6672 (дата звернення 28.05.2021). – Назва з екрану.
26. Javascript [Електронний ресурс]. – Режим доступу: URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення 31.03.2021). – Назва з екрану.
27. Знайомство з React [Електронний ресурс]. – Режим доступу: URL: https://uk.reactjs.org/tutorial/tutorial.html (дата звернення 31.03.2021). – Назва з екрану.

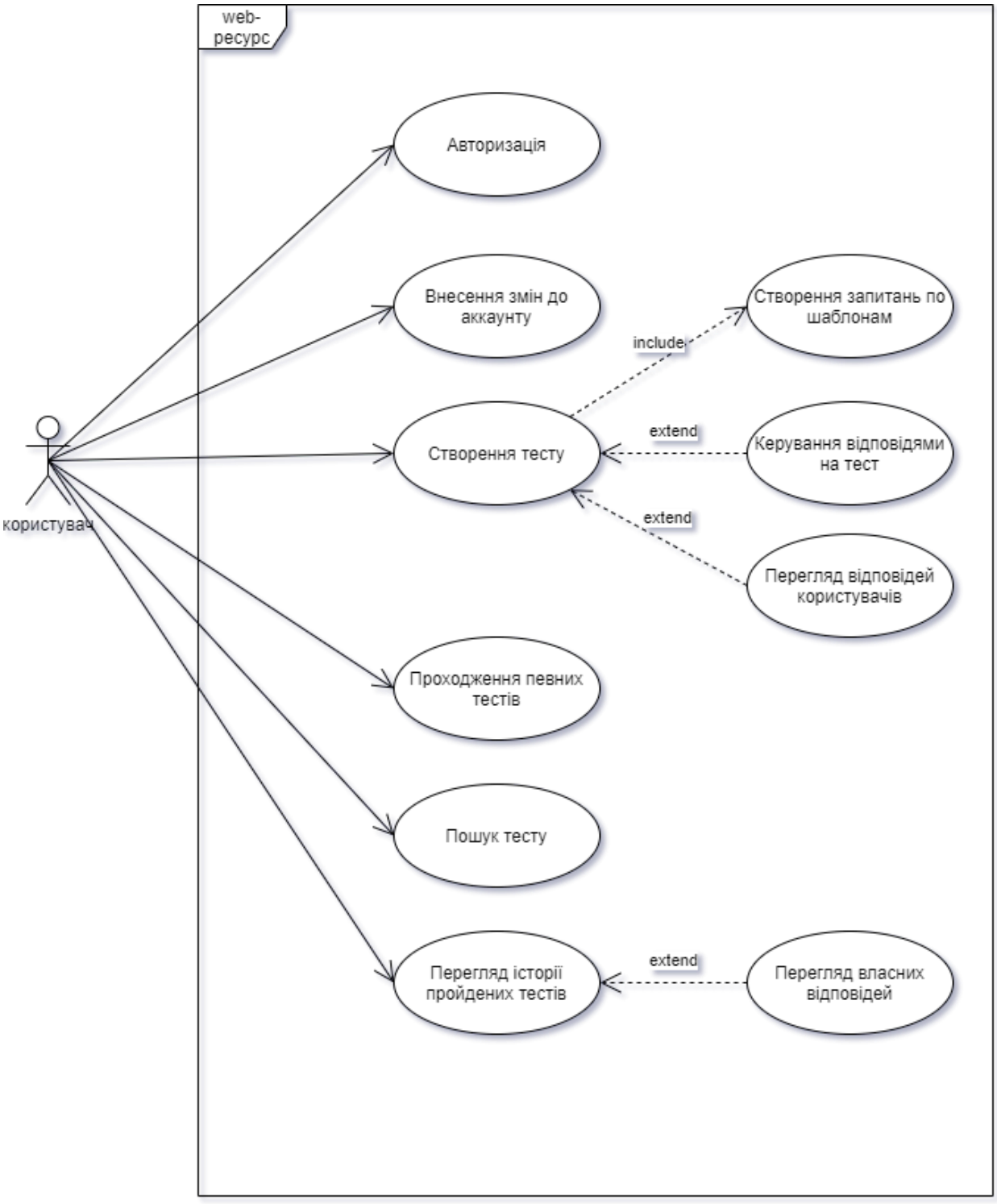
28. Що таке TypeScript? [Електронний ресурс]. – Режим доступу: URL: http://веб.spt42.ru/index.php/chto-takoe-typescript (дата звернення 28.05.2021). – Назва з екрану.
29. ВебStorm [Електронний ресурс]. – Режим доступу: URL: https://uk.wikipedia.org/wiki/ВебStorm (дата звернення 28.05.2021). – Назва з екрану.
30. GitHub [Електронний ресурс]. – Режим доступу: URL: https://sebweo.com/scho-take-git-ta-github-kerivnitstvo-dlya-pochatkivtsiv (дата звернення 28.05.2021). – Назва з екрану.
31. Git та GitHub [Електронний ресурс]. – Режим доступу: URL: https://tproger.ru/translations/difference-between-git-and-github (дата звернення 28.05.2021). – Назва з екрану.
32. Адам Крут. Coding HTML CSS JavaScript Made Easy. Web, Apps and Desktop: навч. посіб. / Адам Крут, Фредерік Джонсон. – Лондон: Flame Tree, 2016. – 256 с.
33. Роман Мельник. Програмування веб-застосувань (фронт-енд та бек-енд): навч. посіб. / Роман Мельник. –Львів: Львівська Політехніка, 2018. – 248 с.
34. Dorling Kindersley. Beginner's Step-by-Step Coding Course: навч. посіб. / Dorling Kindersley. – Лондон: DK, 2020. – 360 с.
35. Ірина Бородкіна. Інженерія програмного забезпечення: навч. посіб. / Ірина Бородкіна, Георгій Бородкин. – Київ: Центр навчальної літератури, 2018. – 204 с.
36. Сергій Кривий. Вступ до методів створення програмних продуктів: навч. посіб. / Сергій Кривий. – Чернівці: Букрек, 2012. – 424 с.
37. Микола Матвієнко. Комп'ютерна логіка: навч. посіб. / Микола Матвієнко. – Київ: Ліра-К, 2012. – 288 с.
38. David Flanagan. JavaScript: The Definitive Guide: навч. посіб. / David Flanagan. – California: O'reilly Media, 2011. – 110 с.

39. David Griffiths. React Cookbook: Recipes for Mastering the React Framework.: навч. посіб. / David Griffiths, Dawn Griffiths. – California: O'reilly Media, 2021. – 500 с.
40. Stoyan Stefanov. React: Up & Running: Building Web Applications: навч. посіб. / Stoyan Stefanov. – California: O'reilly Media, 2021. – 222 с.
41. Зоя Левченко. Облікова політика підприємства: навч. посіб. / Зоя Левченко, Вікторія Кулик. – Київ: SBA-print, 2019. – 238 с.
42. Ігор Яремко. Історія обліку, аналізу та аудиту: навч. посіб. / Ігор Яремко. – Львів: Львівська політехніка, 2018. – 236 с.
43. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня бакалавра / О. В. Ольховська, О. О. Черненко. – Полтава : ПУЕТ, 2022. – 67 с. – 1 електрон. опт. диск (CVD-ROM).

ДОДАТОК А. ТАБЛИЦЯ – ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ПОДІБНИХ ПРОГРАМНИХ СИСТЕМ

Назва веб-ресурсу	Google Form	Online Test Pad	Survey Monkey	Survio	TSTG
Посилання на сайт	https://www.google.com/intl/uk_ua/forms/about	https://onlinetestpad.com/ua	https://www.surveymonkey.ru	https://www.survio.com/ru/svoistva	https://ppc-thesis.веб.app
Оплата	-	-	+	+	-
Вітвління та логіка	-	-	+	+	+
Імпорт/експорт форм	-	-	+	+	-
Україномовний інтерфейс	+	+	-	-	+
Звіт та аналітика	+	-	+	+	+
Мобільність	+	-	+	-	+
Повідомлення	+	-	-	+	-
Зручність інтерфейсу	+	+	-	-	+
Широка функціональність	-	+	+	-	+

ДОДАТОК Б. ДІАГРАМА ПРЕЦЕДЕНТІВ



ДОДАТОК В. ПРОТОТИП ІНТЕРФЕЙСУ

ДОДАТОК Г. ВИХІДНІ КОДИ СИСТЕМИ

```
import React, {createContext, FC} from 'react'
import ReactDOM from 'react-dom'
import {BrowserRouter as Router, Redirect} from 'react-router-dom'
import 'firebaseConfig'
import 'style/index.sass'
import Navbar from 'components/Navbar'
import {Routes} from 'routes'
import {useAuthState} from 'react-firebase-hooks/auth'
import firebase from 'firebase/app'
import {FirebaseContextProps} from 'types/dbTypes'
import 'firebase/firestore'
import Loader from 'components/loader/Loader'
import {HomeSvg, PersonSvg, StudySvg, HistorySvg} from 'constant/icons'
import {authPageId} from 'pages/AuthPage'

const navList = [
  {
    path: '/',
    icon: <HomeSvg/>,
  },
  {
    path: '/study',
    icon: <StudySvg/>,
  },
  {
    path: '/profile',
    icon: <PersonSvg/>,
  },
  {
    path: '/history',
    icon: <HistorySvg/>,
  },
]

const auth = firebase.auth()
const db = firebase.firestore()

export const FirebaseContext = createContext<FirebaseContextProps>({firebase,
auth, user: null, db})

const App: FC = () => {
  const [user, loading] = useAuthState(auth)

  if (loading) return <Loader/>

  if (!user) return (
    <div className="app">
      <Redirect to={`/${authPageId}`}/>
      <Routes/>
    </div>
  )

  return (
    <FirebaseContext.Provider value={{firebase, auth, user, db}}>
      <div className="app">
        <Navbar navigationList={navList}/>
        <div className="main-content">
```

```
          <Routes/>
        </div>
      </div>
    </FirebaseContext.Provider>
  )
}


ReactDOM.render(
  <React.StrictMode>
    <Router>
      <App/>
    </Router>
  </React.StrictMode>,
  document.getElementById('root')
)


import {Switch, Route} from 'react-router-dom'
import React from 'react'
import HomePage from 'pages/HomePage/HomePage'
import UserProfilePage, {userProfilePageId} from 'pages/UserProfilePage'
import AuthPage, {authPageId} from 'pages/AuthPage'
import Study, {studyPageId} from 'pages/StudyPage'
import HistoryPage, {historyPageId} from 'pages/HistoryPage'
import CreateTestPage, {createTestPageId} from
'pages/createTestPage/CreateTestPage'
import MultiTestPage, {multiTestPageId} from
'pages/takingTestPages/MultiTestPage'
import HistoryViewPage from 'pages/HistoryViewPage'

export const Routes = () => {
  return (
    <Switch>
      <Route path={`/`} component={HomePage} exact/>
      <Route path={`/${userProfilePageId}`} component={UserProfilePage}/>
      <Route path={`/${studyPageId}`} component={Study} exact/>
      <Route path={`/${studyPageId}/${multiTestPageId}/:slug`}
component={MultiTestPage}/>
      <Route path={`/${authPageId}`} component={AuthPage}/>
      <Route path={`/${historyPageId}`} component={HistoryPage} exact/>
      <Route path={`/${historyPageId}/:slug`} component={HistoryViewPage}/>
      <Route path={`/${createTestPageId}/:slug`} component={CreateTestPage}/>
    </Switch>
  )
}


import React, {useContext, useEffect} from 'react'
import 'firebase/auth'
import {authUser, getOldUser, setUsersData} from 'api'
import {Redirect} from 'react-router-dom'
import {FirebaseContext} from 'index'
import useFirestoreSet from 'hooks/useFirestoreSet'
import {IUserInitialData} from 'types/dbTypes'
import {noUserImg} from 'constant/icons'
import {APIUrls} from 'constant/api_urls'
import {noGroup} from 'constant/api_constants'
```

```
export const authPageId: string = 'auth'
const AuthPage = () => {
  const {user} = useContext(FirebaseContext)
  const {setDB, response} = useFirestoreSet(APIUrls.users)

  useEffect(() => {
    ;(async () => {
      if (!user) return
      const res = await getOldUser(user.uid)
      if (res?.data()) return
      const body: IUserInitialData = {
        group: noGroup,
        displayName: user.displayName!,
        idDoc: user.uid,
        photoURL: user.photoURL ?? noUserImg,
        completeTestId: []
      }
      await setUsersData(user.uid, body)
    })()
    return () => {
    }

  }, [user, response, setDB])

  const login = async () => {
    await authUser()
  }

  if (user) {
    return <Redirect to={`/`}/>
  }
  return (
    <div className="auth-page">
      <button className="auth-button" onClick={login}>Войти через
Google</button>
    </div>
  )
}

export default AuthPage
```

ДОДАТОК Д. ЗНІМКИ ЕКРАНУ

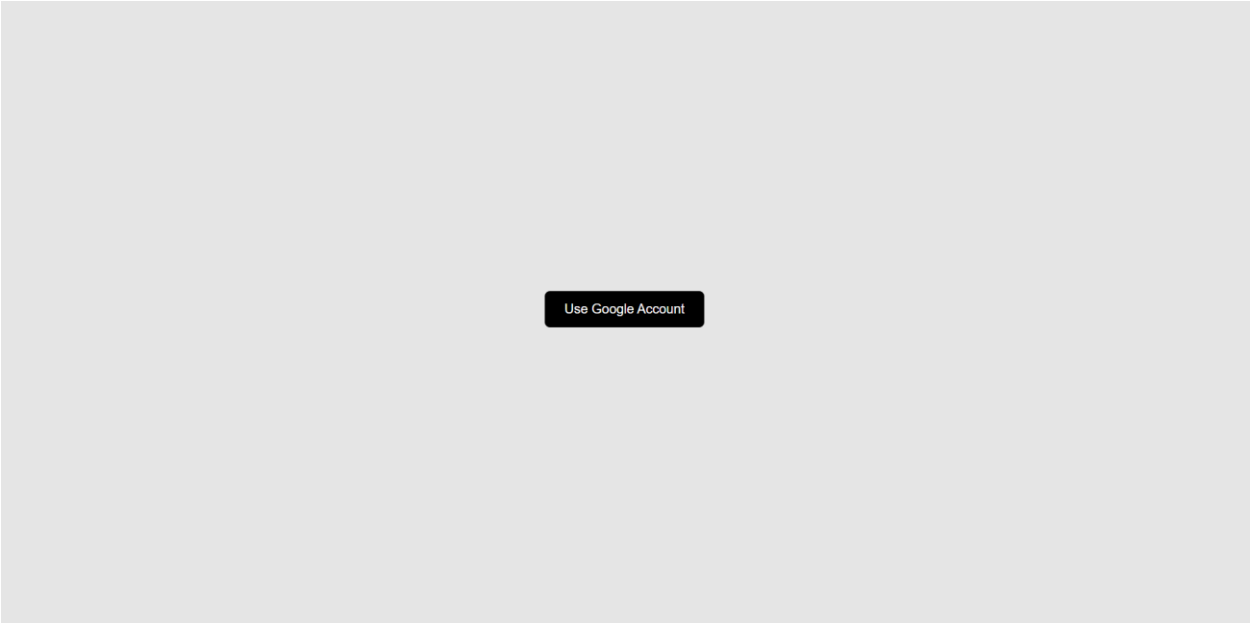


Рисунок Д.1 – Сторінка авторизації

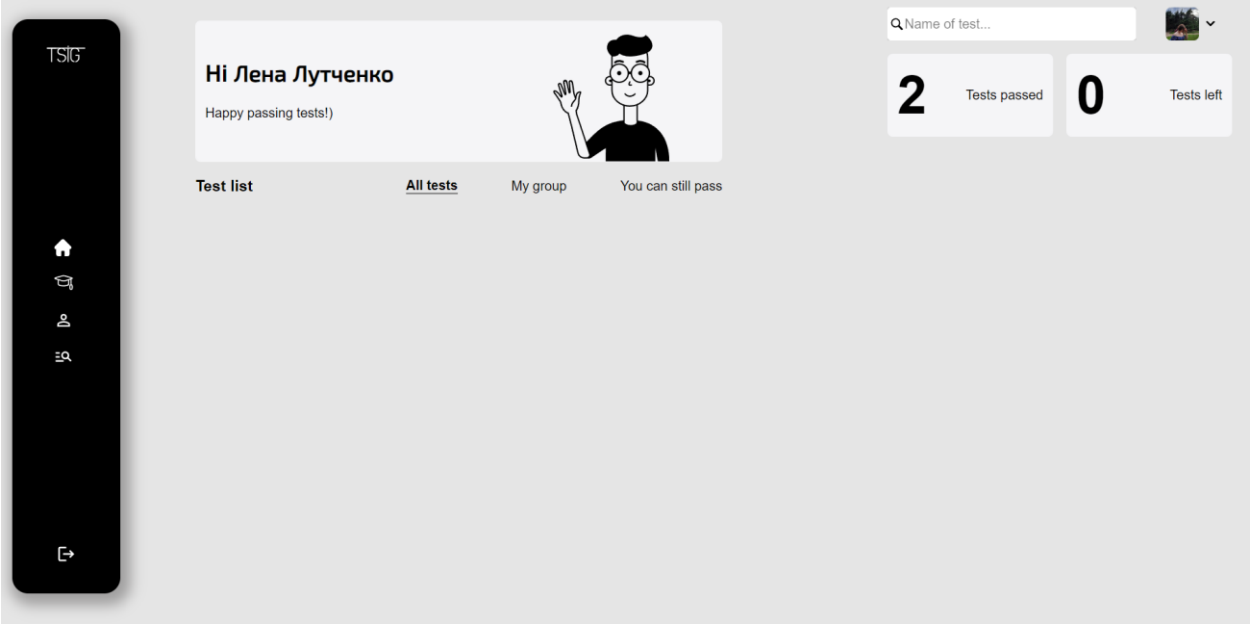


Рисунок Д.2 – Головна сторінка

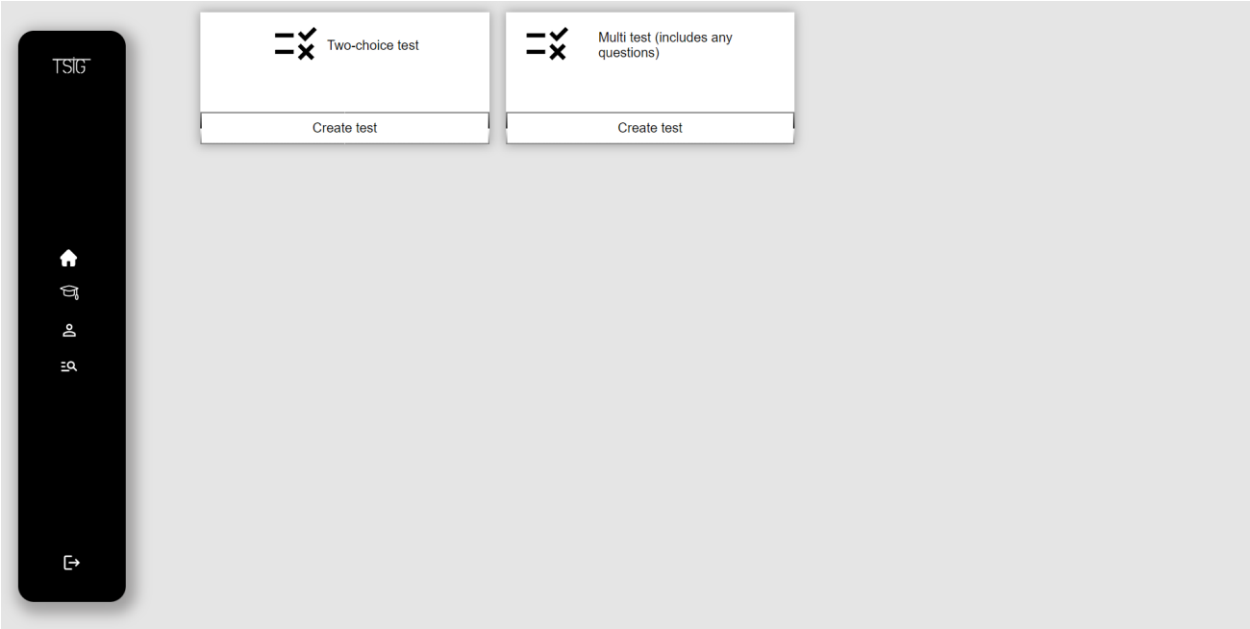


Рисунок Д.3 – Сторінка створення тесту

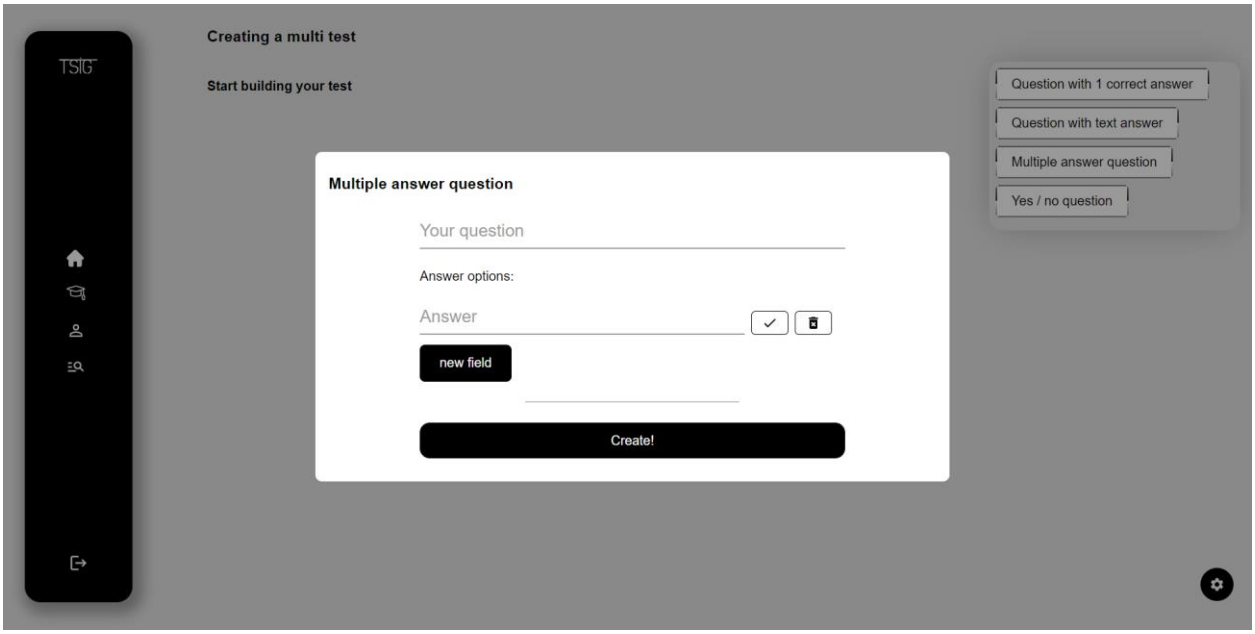


Рисунок Д.4 – Один із шаблонів запитання тесту

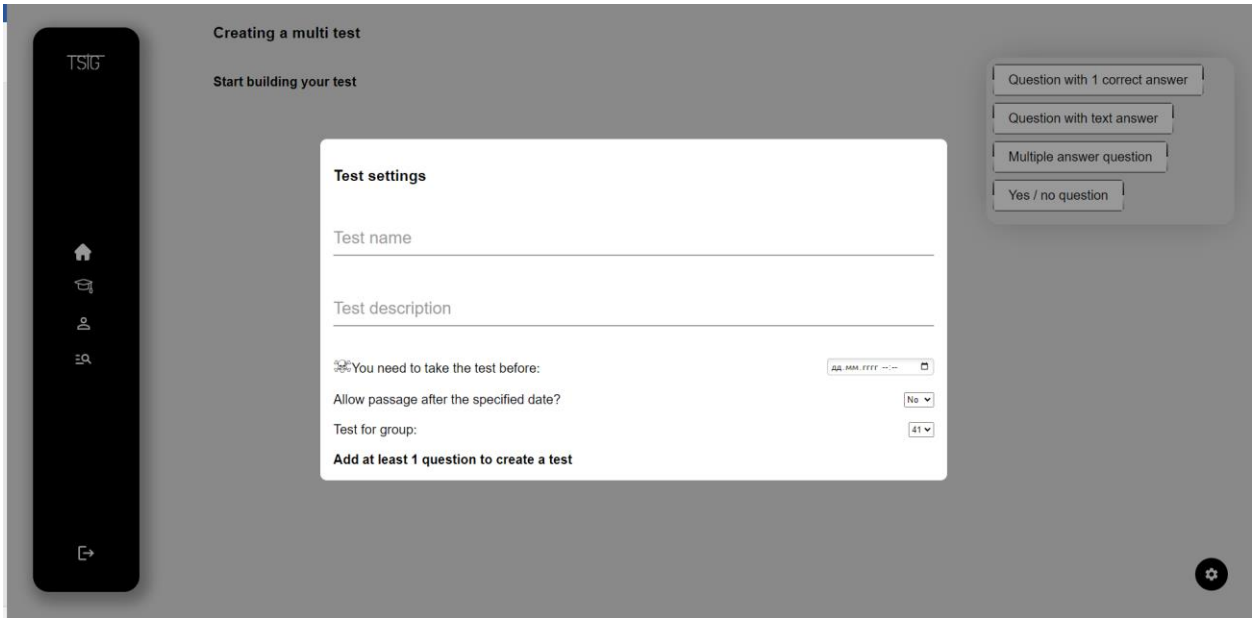


Рисунок Д.5 – Налаштування тесту

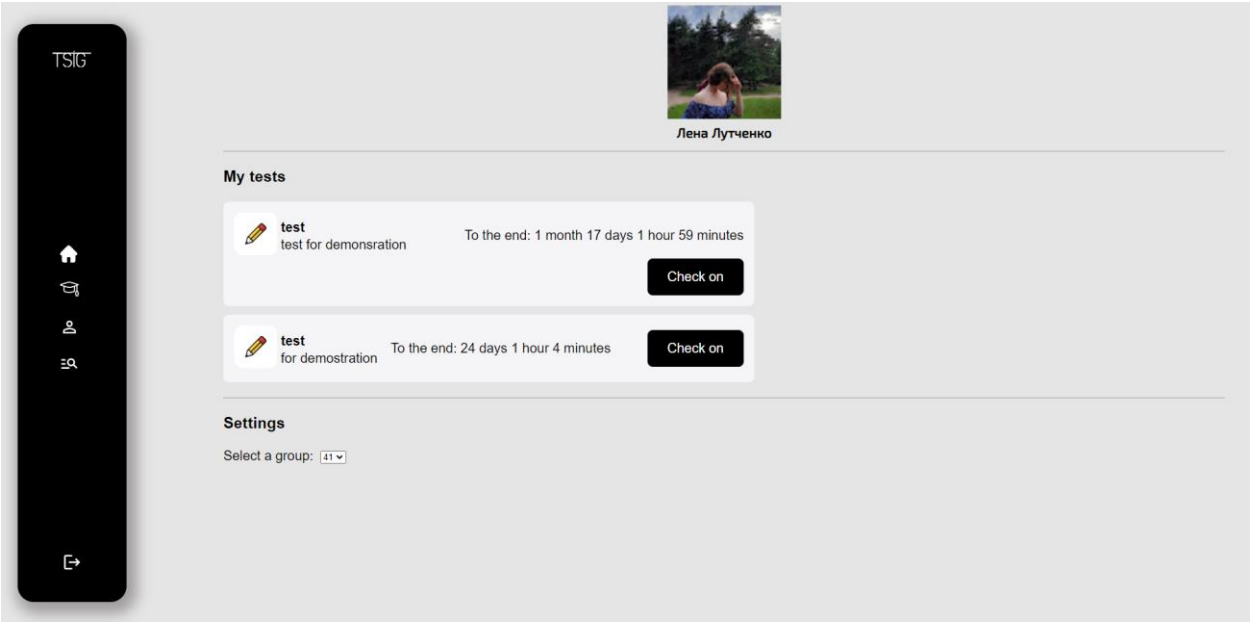


Рисунок Д.6 – Профіль користувача

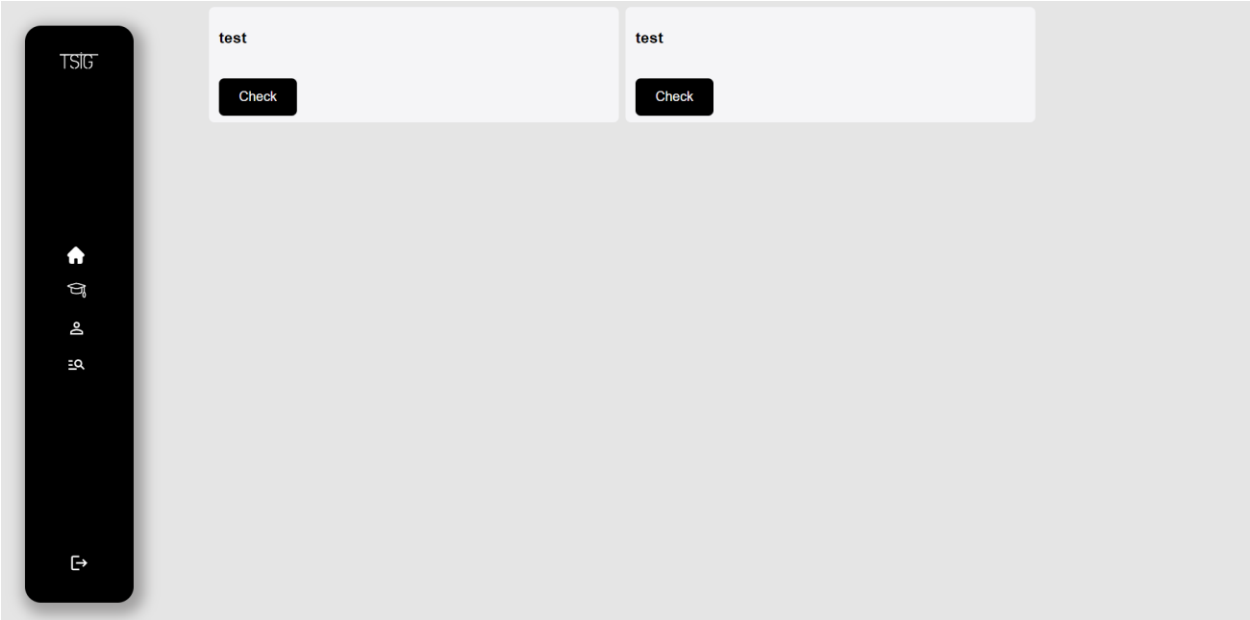


Рисунок Д.7 – Сторінка з історією пройдених тестів

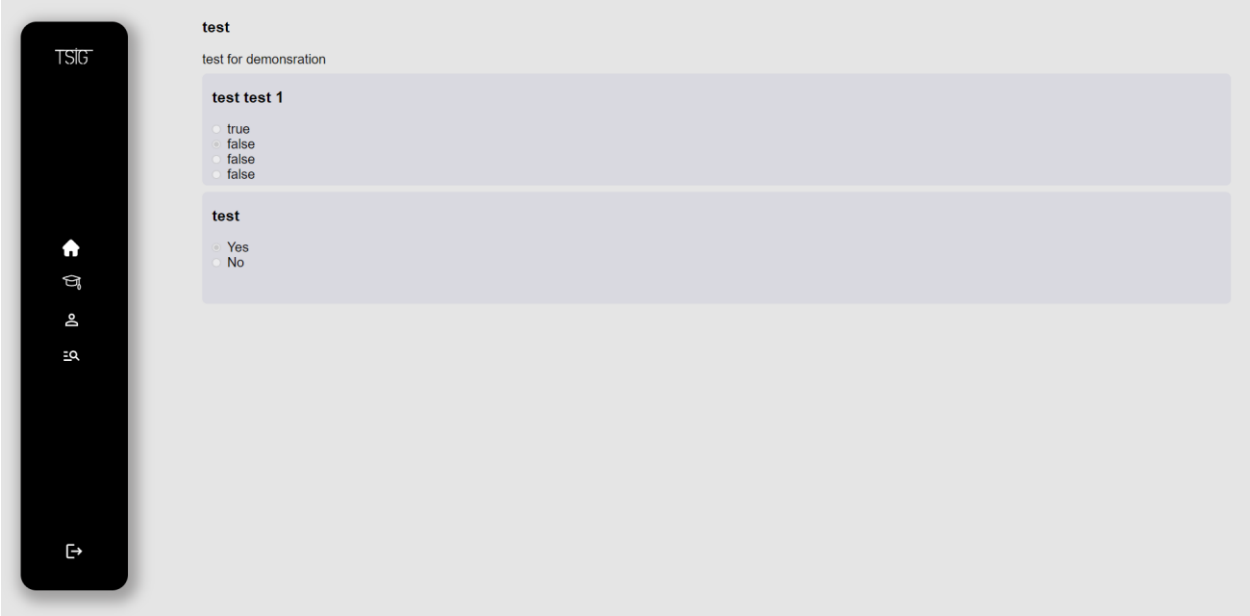


Рисунок Д.8 – Відображення історії проходження певного тесту

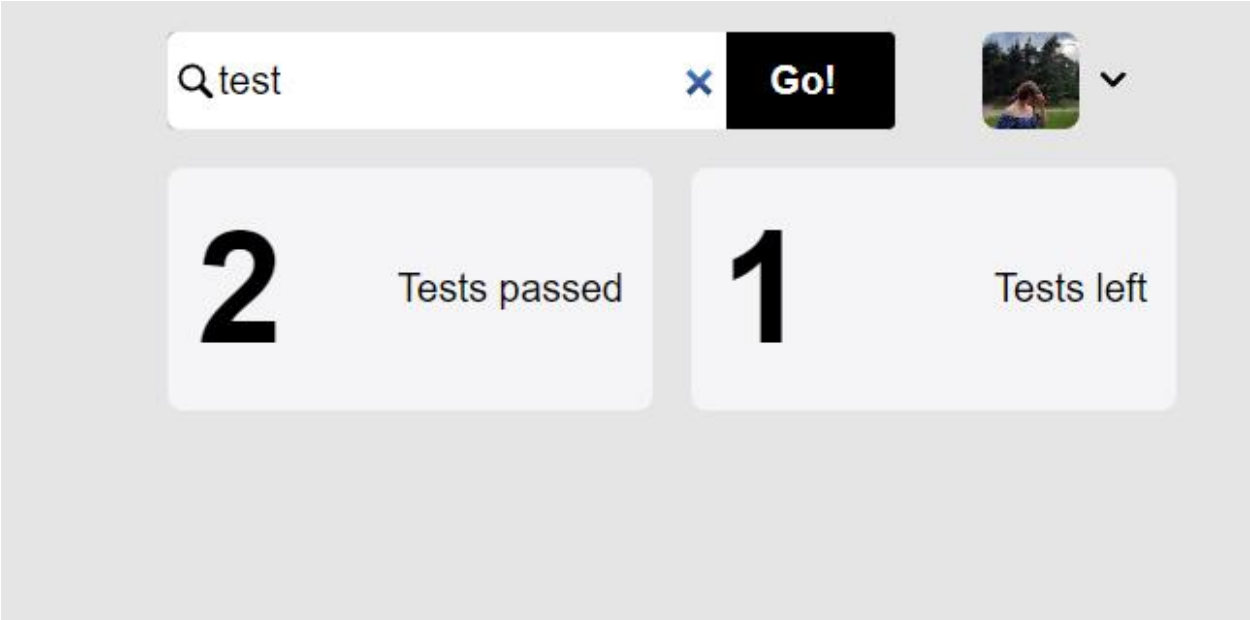


Рисунок Д.9 – Пошук теста та меню профіля.

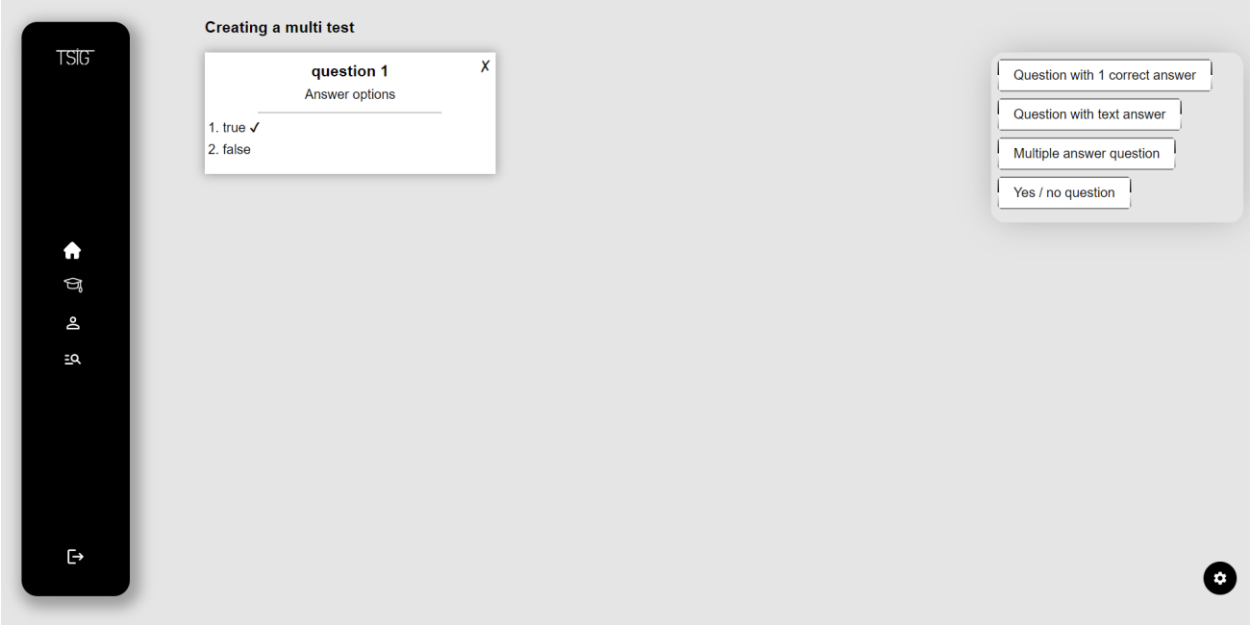


Рисунок Д.10 – Відображення створених питань тесту