Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: « Створення веб-застосунку для перевірки знань з математики»

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Котлінський О.О.
(прізвище та ініціали)

Керівник (підпис) Дмитроца Л.П.
(прізвище та ініціали)

Нормоконтроль (підпис) Литвиненко Я.В.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Карпінський М.П.
(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» червня 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Котлінський Олег Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи « Створення веб-застосунку для перевірки знань з математики»

Керівник роботи Дмитроца Леся Павлівна, кандидат технічних наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-133

2. Термін подання студентом завершеної роботи 19.06.2023 р.

3. Вихідні дані до роботи Інтернет джерела щодо створення застосунку для перевірки знань З математики та роботи з фреймворком flask

4. Зміст роботи (перелік питань, які потрібно розробити)
Розділ 1. Теорія про створення веб-застосунку для перевірки знань з математики. 1.1. Важливість та значення веб-застосунку для перевірки знань з математики. 1.2 Вибір та порівняння мови програмування для створення застосунку. 1.3 Вибір фреймворка для роботи. 1.4 Етапи розробки веб-застосунку. 1.5 Дизайн інтерфейсу та взаємодія з користувачем. 1.6 Висновок першого розділу. Розділ 2. Практична розробка веб-застосунку для перевірки знань з математики. 2.1 Створення серверної частини. 2.2 Розробка фронтенду. 2.3 Результати веб-застосунку. 2.4 Висновок другого розділу. Розділ 3. Безпека життєдіяльності та охорона праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Мета та задачі. 3. Актуальність теми. 4. Етапи розробки веб-застосунку. 5. Програмне забезпечення для створення. 6. Створення серверної частини.7.Розробка фронтенду. 8. Головне меню. 9. Сторінка «Тестування». 10. Висновок.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці		05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про створення веб-застосунку для перевірки знань з математики	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо створення веб-застосунку. Створення серверної частини, розробка фронтенду, показ результатів та тестових даних	01.02.2023-07.02.2023	*Виконано*
5.	Оформлення розділу «Теорія про створення веб-застосунку для перевірки знань з математики»	08.02.2023-09.02.2023	*Виконано*
6.	Оформлення розділу «Практична розробка застосунку для перевірки знань з математики»	10.02.2023-12.02.2023	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони праці»	07.06.2023-08.06.2023	*Виконано*
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Котлінський О.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення веб-застосунку для перевірки знань з математики // Кваліфікаційна робота освітнього рівня «Бакалавр» // Котлінський Олег Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2023 // С. 43, рис. 9 , табл. 1, кресл. 10, додат. 13 , бібліогр. 30 .

Ключові слова: flask, python, html, css, фронтенд, користувацький інтерфейс, головна сторінка, веб-застосунок.

Кваліфікаційна робота присв'ячена розробці веб-застосунку для перевірки знань з математики. В першому розділі кваліфікаційної роботи описано актуальність застосунку, теорія та навчання, педагогічні принципи. Розглянуто дизайн інтерфейсу та взаємодія з користувачем. Зроблено огляд гнучкості застосунку та його масштабованість.

В другому розділі кваліфікаційної роботи показано розробку веб-застосунку, а саме його серверної частини, створення фронтенду. Застосовано показ кінцевих результатів.

В третьому розділі кваліфікаційної роботи з безпеки життєдіяльності описано роль захисту персональних даних. З охорони праці загальна мета полягає в тому, щоб створити здорове та безпечне робоче місце для розробників, що працюють над веб-застосунком.

ANNOTATION

Creation of a web application for testing knowledge in mathematics // Qualification work of the educational level "Bachelor" // Kotlinskyi Oleh Oleksandrovich // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2023 // P. 43, fig. 9, tabl. 1, chair. 10, annexes. 13, references 30.

Keywords: flask, python, html, css, frontend, user interface, main page, web application.

The qualification work is devoted to the development of a web application for testing knowledge in mathematics. The first section of the qualification work describes the relevance of the application, theory and training, pedagogical principles. Interface design and user interaction are considered. An overview of the flexibility of the application and its scalability was made.

The second section of the qualification work shows the development of the web application, namely its server part, the creation of the front end. Display of final results applied.

The role of personal data protection is described in the third section of the qualification work on life safety. In health and safety, the overall goal is to create a healthy and safe workplace for developers working on a web application.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ТЗ – тестове завдання.

ШІ – штучний інтелект.

HTML (анг. HyperText Markup Language) – мова розмітки гіпертексту.

CSS (анг. Cascading Style Sheets) – каскадні таблиці стилів, спеціальна мова стилів, за допомогою якої формуються веб-сторінки.

DOCTYPE (анг. Document Type) - це декларація типу документа в мові HTML.

ВЗ – веб-застосунок.

ВК – веб-клієнт.

ІК – інтерфейс користувача.

ВР – веб-розробка.

ОЗУ – оперативна пам’ять.

ORM – об’єктно-реляційне відображення.

СУБД – система управління базами даних.

ЗМІСТ

ВСТУП

Актуальність теми. У зв'язку з розвитком технологій і збільшенням доступу до Інтернету, онлайн-навчання стає все більш поширеним. ВБ для перевірки знань з математики можуть бути цінним інструментом для самостійного навчання та оцінки академічного прогресу. ВЗ для перевірки знань можуть бути налаштовані для відповідності індивідуальному рівню знань і потребам кожного учня. Вони можуть надавати завдання на різні рівні складності, підказки або додаткові пояснення, що допомагають кожному учневі налагодити своє навчання. ВЗ також можуть автоматично перевіряти відповіді учнів і надавати миттєвий зворотній зв'язок. Це зменшує навантаження на вчителя і дозволяє учням отримувати негайну інформацію про свої успіхи та помилки. Також ВЗ може створювати можливості для взаємодії між учнями, наприклад, шляхом організації онлайн-конкурсів або спільної роботи над завданнями. Це може підвищити мотивацію до навчання математики та стимулювати учнів до досягнення кращих результатів.

Мета і задачі дослідження. Мета розробки ВЗ для перевірки знань з математики полягає в створенні ефективного та корисного інструменту, який допоможе студентам покращити свої математичні навички та перевірити рівень своїх знань. Провести дослідження, щоб визначити потреби учнів, вчителів та батьків щодо забезпечення перевірки та покращення знань з математики. Сформулювати основні вимоги до функціональності веб-застосунку. Розробити архітектурний план та структуру ВЗ, включаючи основні компоненти та інтерфейс. Провести тестування різних типів завдань, аналізувати результати та переконатися у правильності та ефективності роботи ВЗ. Оцінити, наскільки успішно веб-застосунок сприяє покращенню знань з математики учнів та забезпечує зручне та корисне навчання. Вдосконалити веб-застосунок, враховуючи отримані результати, зворотний зв'язок від користувачів та використання найкращих практик у розробці.

Дослідження задачі "Розробка веб-застосунку для перевірки знань з

математики" допоможе забезпечити ефективний та корисний інструмент для навчання та перевірки знань з математики, забезпечуючи користувача всім необіхдним для навчання.

Об’єкт дослідження: Об'єктом дослідження в даній темі є сам процес створення веб-застосунку для перевірки знань з математики та його вплив на стосовно покращення навчання для учнів. У самому досліджені будуть розглядатись сам веб-застосунок, його складові, функціональні можливості та спосіб взаємодії з користувачами. Основна увага буде приділено аналізу, розробці та удосконаленню веб-застосунку з метою ефективної перевірки знань з математики та поліпшення процесу.

Предмет дослідження: Предметом дослідження у вказаній темі є процес розробки веб-застосунку, спрямованого на перевірку та покращення знань з математики учнів. Дослідження включає аналіз, розробку та вдосконалення функціоналу веб-застосунку, що собою охоплює створення завдань, перевірка відповідей, забезпечення зворотного зв'язку та збереження результатів. Крім того, в рамках дослідження вивчається ефективність самого ВЗ та його вплив на перевірку та покращення знань з математики для користувачів.

РОЗДІЛ 1. ТЕОРІЯ СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ З МАТЕМАТИКИ

1.1 Важливість та значення веб-застосунку для перевірки знань з математики

Веб-застосунок для перевірки знань з математики має велику важливість та значення як для студентів, так і для звичайного користувача мережі інтернет. Застосунок може полегшити безліч процесів, але також варто розуміти і його різницю між веб-сайтом. Нижче наведено приклад (таблиця 1.1).

Таблиця 1.1 – порівняння веб-застосунку і веб-сайту

Характеристика	**Веб-застосунок**	**Веб-сайт**
1	**2**	**3**
Функціональність	Зорієнтований на виконання конкретних завдань або функцій. Може мати складний функціонал та взаємодію з користувачем.	Більш статичний, зорієнтований на надання інформації або представлення контенту. Може містити блоги, новини, сторінки компанії тощо.
Взаємодія	Забезпечує можливість взаємодії з користувачем, наприклад, введення даних, обробка форм, авторизація, персоналізація тощо.	Взаємодія з користувачем обмежена до перегляду вмісту, коментування, відправки запитів через контактні форми тощо.

Продовження таблиці 1.1

1	2	3
Керованість	Більш гнучкий, може мати адміністративну панель або інтерфейс для керування та оновлення функціоналу та вмісту.	Зазвичай керованість обмежена до оновлення вмісту (текст, зображення) через систему управління вмістом (CMS).
Динамічність	Може мати динамічний функціонал, відповідати на дії користувача та змінювати вміст.	Зазвичай має обмежену динаміку, зміна вмісту відбувається через перезавантаження.
Орієнтація	Більш орієнтований на розв'язання конкретних задач або надання конкретних послуг.	Більш орієнтований на представлення інформації та комунікацію з користувачами.
Доступність	Може бути доступним лише для обмеженого кола користувачів, наприклад, з обліковим записом або після платежу.	Зазвичай відкритий для всіх користувачів Інтернету.
Складність	Може бути більш складним у розробці через вимоги до функціоналу та взаємодії з користувачем.	Зазвичай менш складний у розробці, з фокусом на представленні вмісту.
Приклади	Електронний магазин, онлайн-ігри, платформи.	Корпоративний сайт, блог, новинний портал.

Головна перевага це зручність, тому що він може бути доступний з будь-якого пристрою, який має доступ до мережі інтернет. Далі йде покращення навчання та опрацювання матеріалу згідно своїх потреб та темпу в якому хоче навчатися користувач.

Покращення навчання, завдання, вправи та тести, які допомагають покращити розуміння концепцій, виробити навички та засвоїти матеріал більш ефективно. Користувач може отримати миттєвий зворотний зв'язок щодо своїх відповідей та помилок.

Загалом, ВЗ для перевірки знань з математики може значно полегшити процес навчання та забезпечити користувачам зручні, інтерактивні та персоналізовані можливості для вдосконалення своїх математичних навичок.

1.2 Вибір та порівняння мови програмування для створення застосунку

Існує безліч мов програмування для створення ВБ, але серед них можна виділити найпопулярніші такі як – Java, JavaScript, Ruby, Python.Тому розглянемо кожну з мов, їх переваги та недоліки:

– Java – перевагами буде широке застосування у ВР та фреймворків, велика кількість бібліотек, підтримка браузерів для клієнтської частини. Недоліки – часті зміни стандартів та вразливість до помилок безпеки.

– JavaScript - це високорівнева, інтерпретована мова програмування, яка використовується для створення живих веб-сторінок. Вона дозволяє спілкуватися з користувачем безпосередньо у веб-браузері. JavaScript допомагає розробляти веб-додатки, які реагують на події, змінюють вміст сторінки, перевіряють форми, анімують елементи, виконують запити до сервера та багато іншого. JavaScript - одна з трьох основних технологій веб-розробки разом з HTML і CSS. Вона додає динамічність та інтерактивність до веб-сторінок. JavaScript - це мова сценаріїв, тому код можна виконувати безпосередньо на стороні клієнта без потреби у компіляції. У JavaScript є велика кількість вбудованих функцій та об'єктів, які розширюють її

можливості. Також її можна поєднувати з зовнішніми бібліотеками і фреймворками, що дозволяє розробникам швидко створювати складні веб-додатки. JavaScript застосовується не тільки на веб-сторінках, але й у різних областях, таких як мобільна розробка, настільні програми, серверні додатки та інше. Вона є однією з найпопулярніших мов програмування у світі і необхідна для багатьох сучасних веб-технологій.

– Ruby – це мова програмування, створена у Японії, яка використовується для розробки різноманітних додатків. Вона має простий і зрозумілий синтаксис, що дозволяє розробникам писати код лаконічно і зручно. Ruby підтримує різні парадигми програмування, включаючи процедурне, функціональне і метапрограмування. Основним принципом Ruby є забезпечення приємного досвіду програмування, наголошуючи на простоті, зрозумілості коду і продуктивності розробника. Вона пропонує широкий набір вбудованих функцій і бібліотек, що спрощують процес розробки. Ruby широко використовується для розробки веб-додатків, існують популярні фреймворки, такі як Ruby on Rails, що спрощують процес розробки веб-додатків. Вона також застосовується у розробці настільних програм, сценаріїв, автоматизації завдань та інших областей програмування. Ruby є відкритою мовою програмування з активною спільнотою розробників, яка підтримує її розвиток і популяризацію. Вона відома своєю елегантністю, гнучкістю і зручністю використання, що робить її привабливим вибором для багатьох програмістів.

– Python – перевагами буде простий та зрозумілий синтаксис, велика кількість бібліотек та фрейворків, зручна розробка прототипів та веб-застосунків. З недоліків це – повільна швидкодія, обмежена підтримка браузерами для фронтенду, не така масштабованість, як інші мови.

Виходячи з цих порівнянь для себе я вибрав таку мову прогрумування для розробки ВБ, як Python. Він має привабливий та легкий для сприйняття синтаксис, що робить його привітним для новачків та спрощує процес розробки та підтримки коду. Python також пропонує широкий вибір функціональних бібліотек та фреймворків, таких як Django, Flask, Pyramid та інші, які надають

готові рішення для рутинних задач і сприяють прискоренню процесу розробки. Завдяки потужності Python, я зможу швидко розробляти прототипи та веб-застосунки, експериментувати з кодом та функціоналом. Крім того, наявність великої та активної спільноти розробників Python забезпечує безліч ресурсів, документації та підтримки, що допомагає знайти відповіді на запитання та вирішити проблеми. Python є переносним, що означає, що веб-застосунки, розроблені на Python, легко працюють на різних операційних системах та серверах без необхідності значних змін. Це забезпечує більшу гнучкість та доступність у різних середовищах.

1.3 Вибір фрейм-ворка для роботи

Після вибору мови програмування, потрібно вибрати фреймворк, на базі програмування Python є три фреймворки – Flask, Django і Pyramid. Розберемо кожен з них і проведемо порівняння.

Flask є веб-фреймворком, який дозволяє розробляти легкі веб-застосунки з використанням мови програмування Python. Він є одним з популярних фреймворків для розробки ВЗ у середовищі Python. Flask має привабливі риси, такі як простий синтаксис, що робить його доступним для початківців і сприяє швидкій розробці. Крім того, Flask надає велику свободу розробникам у виборі компонентів та підходів, що дозволяє створювати ВЗ за своїми потребами. Фреймворк також має багато розширень і бібліотек, які дозволяють збільшити його функціонал та швидко додавати додаткові функції у ВЗ. За допомогою простої системи маршрутизації, Flask дозволяє визначати URL-шаблони та з'єднувати їх з функціями-обробниками, що дозволяє створювати сторінки та обробляти запити ВК. Крім того, Flask має гарну інтеграцію з іншими компонентами та інструментами екосистеми Python, такими як бази даних, шаблонізатори та інші бібліотеки для розробки веб-додатків. Всі ці риси роблять Flask привабливим вибором для розробників, які шукають простоту, гнучкість та можливість швидкої розробки веб-застосунків.

Django – це веб-фреймворк високого рівня, розроблений на мові програмування Python, який дозволяє швидко та ефективно створювати складні веб-застосунки. Він надає розробникам набір інструментів, шаблонів та структур для розробки веб-додатків з різноманітним функціоналом, таким як обробка запитів, маршрутизація URL-адрес, робота з базами даних, управління аутентифікацією та авторизацією, адміністративний інтерфейс та багато іншого. Django пропонує концепцію "керованого фреймворку", що означає, що він надає структуру та стандарти для розробки, спрощуючи процес і полегшуючи підтримку проекту. Він використовує архітектуру модель-представлення-контролер (MVC) або модель-представлення-шаблон (MVT), що дозволяє розділити логіку програми, дані та представлення. Однією з сильних сторін Django є його велика кількість готових компонентів і бібліотек, які можна використовувати для розширення функціональності веб-застосунків. Наприклад, Django має вбудовану систему адміністрування, яка надає готовий інтерфейс для керування даними в базі даних. Крім того, Django підтримує роботу з різними базами даних, такими як SQLite, MySQL, PostgreSQL та інші. Ще однією перевагою Django є активна спільнота розробників, яка надає багато ресурсів, документацію, пакети розширень та підтримку. Це дозволяє розробникам швидко знайти відповіді на свої питання та отримати допомогу у вирішенні проблем. Загалом, Django є потужним фреймворком, який дозволяє ефективно розробляти веб-застосунки зі складним функціоналом. Він має велику кількість компонентів та підтримку спільноти розробників. Django ідеально підходить для середніх і великих проектів, де важлива структурована та масштабована розробка.

Pyramid – це веб-фреймворк високого рівня, розроблений на мові програмування Python. Він надає потужні інструменти для розробки веб-додатків з різноманітним функціоналом. Pyramid пропонує простоту використання, гнучкість і широкі можливості налаштування. Фреймворк сам є мінімальним, надаючи лише базовий набір інструментів,потрібних для створення веб-додатків. Це означає, що розробники мають більшу свободу

вибору і налаштування компонентів і бібліотек, які вони хочуть використовувати у своїх проектах. Pyramid базується на архітектурі WSGI (Web Server Gateway Interface), що дозволяє легко інтегрувати фреймворк з різними веб-серверами і засобами розгортання. Він також підтримує різні шаблонні механізми, такі як Jinja2 та Chameleon, для розробки користувацького інтерфейсу. Pyramid надає розробникам гнучкість у виборі бази даних, оскільки він не обмежує використання певної бази даних. Розробники можуть використовувати SQLite, PostgreSQL, MySQL або інші бази даних за своїми потребами. Це робить фреймворк доступним для навчання і розвитку. Загалом, Pyramid - це гнучкий і потужний фреймворк, який надає розробникам велику свободу у виборы на налаштуваннях. Він підыйде для розробки різноманітних веб-додатків із документацією та активними розробниками.

Виходячи з цієї інформації, мною був обраний такий веб-фреймворк як Flask, по таких причинах:

- Володіє простим та легким у використанні синтаксисом. Він не нав'язує жорстких правил та структур. Це особливо корисно для початківців, що тільки хочуть швидко почати програмувати веб-застосунки.
- Дозволяє вам самостійно обрати компоненти та розширення, які ви хочу використати у своєму проєкті. Він не має попередньо встановлених компонентів, що дозволяє вам створювати веб-застосунок з урахуванням моїх потреб, для створення веб-застосунку для перевірки знань з математики.
- Має невеликий обсяг коду та залежності, що дає змогу швидко запустити і відкрити застосунок. Не навантажує проєкт та дозволяє ефективно використовувати ОЗУ.
- Має широкий вибір розширень та бібліотек, які можна використовувати для розширення. Це дозволить мені легко додавати нові розширення до проєкту без необхідності писати код.

1.4 Етапи розробки веб-застосунку

Для того, щоб почати створення ВЗ потрібно для початку встановити фреймворк Flask, для цього в терміналі потрібно вписати наступну команду (Див.рисунок 1.1).

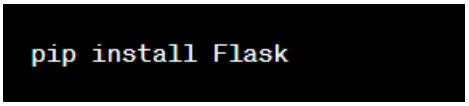


Рисунок 1.1 – встановлення фреймворка Flask.

Після того, як встановили Flask, ми можемо спокійно створити простий файл. Потім перейти за посиланням , яке буде у нас в терміналі. Після переходу за посиланням у нас відкриється наступне вікно з виводом «Hello Flask». Так ми переконаємося, що встановили все належним чином і що програма працює добре.

HTML є мовою розмітки, яка використовується для створення веб-сторінок. Вона дозволяє визначати структуру та вміст документа, включаючи заголовки, параграфи, списки, таблиці, зображення та посилання. HTML використовується за допомогою тегів, які вказують браузеру, як правильно відображати вміст сторінки. Наприклад, <p> тег використовується для відображення параграфу, а <img> - для зображень. Крім тегів, HTML також використовує атрибути, які надають додаткову інформацію про елементи. Наприклад, атрибут src вказує шлях до зображення у тезі <img src="image.jpg">. За допомогою HTML можна створювати посилання, форми для введення даних, таблиці для відображення даних у вигляді сітки, а також використовувати аудіо та відео елементи. Крім того, HTML можна стилізувати за допомогою CSS та додати динамічність та інтерактивність до сторінки за допомогою скриптів, таких як JavaScript. HTML є стандартом, і є основою для створення веб-сторінок та веб-застосунків.

Для того щоб задати маршрути ми викорикористовуємо декоратор, де кожен маршрут буде відповідати певній адресі, також пов’язаній функції, яка буде виконувати певні дії. Самі маршрути будуть розширені для виконання більш складніших дій, для відображення HTML-сторінок.

Для того щоб відображати HTML-сторінки потрібно використати функцію «render_template()». Саме ця функція дасть змогу підключати HTML шаблони до нашого ВЗ. Для прикладу (Див.рисунок 1.2).

Використовується для відображення шаблонів «home.html», «questions.html» та «results.html». Далі ми створюємо відповідні шаблони HTML у проєкті, щоб вони могли бути використані для відображення вмісту сторінок.

Далі ми обираємо базу даних або файлову систему для збереження наших результатів відповідей. База даних дозволить нам зберігати дані та здійснювати операції з ними. У Flask є підтримка різних баз даних, таких як SQLite, MySQL, PostgreSQL та багато інших. Для роботи з базою даних можна використовувати ORM (об'єктно-реляційне відображення) бібліотеку, наприклад SQLAlchemy, яка допоможе нам здійснювати операції з базою даних. Проте, якщо ми хочемо зберігати дані у файловій системі. Тоді потрібно створити теку або файли, в яких будуть зберігатись дані нашого додатку. Наприклад, створюємо файл JSON або CSV, в якому будуть зберігатись запитання та відповіді у відповідному форматі. Додаток може прочитати цей файл та використовувати дані для генерації питань та перевірки відповідей.

Далі ми обираємо базу даних або файлову систему для збереження наших результатів відповідей. База даних дозволить нам зберігати дані та здійснювати операції з ними. У Flask є підтримка різних баз даних, таких як SQLite, MySQL, PostgreSQL та багато інших. Для роботи з базою даних можна використовувати ORM (об'єктно-реляційне відображення) бібліотеку, наприклад SQLAlchemy, яка допоможе нам здійснювати операції з базою даних. Проте, якщо ми хочемо зберігати дані у файловій системі. Тоді потрібно створити теку або файли, в яких будуть зберігатись дані нашого додатку. Наприклад, створюємо файл JSON або CSV, в якому будуть зберігатись запитання та відповіді у

відповідному форматі. Додаток може прочитати цей файл та використовувати дані для генерації питань та перевірки відповідей.

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/questions')
def questions():
    return render_template('questions.html')

@app.route('/results')
def results():
    return render_template('results.html')

if __name__ == '__main__':
    app.run()
```

Рисунок 1.2 – Відображення HTML шаблонів

В обох випадках ми будемо створювати таблицю або файл для створення об’єкту. Проте коли будемо використовувати базу даних ми будемо викликати «db.create_all», щоб створити всі таблиці в базі даних, а у файловій системі будемо викликати «open()» для створення файлу і записувати вміст за командою «file.write()».

SQLAlchemy є бібліотекою для роботи з базами даних, яка надає об'єктно-реляційне відображення (ORM). Це означає, що ви можете використовувати класи Python для представлення таблиць у базі даних, а об'єкти цих класів можна зберігати, отримувати, оновлювати та видаляти з бази даних без прямого написання складних SQL-запитів. Завдяки SQLAlchemy, ви можете працювати з базою даних в об'єктно-орієнтованому стилі, що робить код більш зрозумілим та підтримує його читабельність. Ви можете

використовувати методи і атрибути класів для взаємодії з даними, замість написання складних SQL-запитів вручну. Одна з основних переваг SQLAlchemy полягає в тому, що вона підтримує багато різних систем керування базами даних, таких як MySQL, PostgreSQL, SQLite, Oracle і багато інших. Це дозволяє вам легко переносити ваш додаток між різними СУБД без необхідності переписування коду. SQLAlchemy надає широкий набір функціональності для роботи з базою даних. Ви можете використовувати його для виконання транзакцій, створення складних запитів з фільтрацією, сортуванням і об'єднанням, виконання агрегаційних функцій та багато іншого. SQLAlchemy також дозволяє вам використовувати параметризовані запити для уникнення атак типу «SQL Injection» і забезпечення безпеки вашого додатку та даних користувачів. У Flask-додатках SQLAlchemy є одним з найпопулярніших інструментів для роботи з базами даних. Він має сильну спільноту, добре документований та часто використовується в розширеннях та пакетах Flask, що надають зручну інтеграцію з SQLAlchemy. Загалом, використання SQLAlchemy спрощує роботу з базами даних у вашому веб-додатку, робить код більш зрозумілим і підтримує переносимість між різними СУБД. Вона також допомагає забезпечити безпеку вашого додатку та даних користувачів за допомогою параметризованих запитів.

1.5 Дизайн інтерфейсу та взаємодія з користувачем

Дизайн інтерфейсу користувача – це процес створення привабливого та функціонального вигляду веб-застосунків або мобільних додатків, що включає в себе використання кольорів, типографіки, макетів, графіки та інших візуальних елементів. Метою дизайну інтерфейсу є забезпечення зручної, привабливої та зрозумілої взаємодії між користувачем та додатком, що підвищує його ефективність та задоволення від використання.

Потрібно, який надає широкий набір функціональних можливостей, таких як створення різних типів елементів інтерфейсу, можливість працювати з

анімаціями, розташуванням елементів, відображенням даних. Інструмент, який буде підтримувати розробку адаптивного дизайну, що дозволить нашому ВЗ адаптуватися до різних розмірів екранів і пристроїв. Розглянути можливості інструменту щодо візуалізації даних, так як потрібно відображати графіки, діаграми або інші візуальні елементи для представлення результатів перевірки знань з математики. Також важливо, щоб інструмент був зрозумілим і зручним у використанні. Обрати інструмент з інтуїтивним інтерфейсом та добре документованою документацією, яка допоможе швидко навчитися його використовувати.

Перед тим як обрати один з інструментів такі як HTML/CSS, JavaScript, React, Angular та інші. Вони відрізняються своїми властивостями та мовою програмування, нижче наведемо приклади:

- HTML/CSS: Мови розмітки та стилізації для створення структури та дизайну, мови програмування - HTML/CSS. Прості у використані та можливість розробки статичного інтерфейсу.
- JavaScript: Мова для динамічної взаємодії з інтерфейсом, мова програмування JavaScript. Інтерактивна та має можливість взаємодії з користувачем, анімації.
- React: JavaScript бібліотека для розробки інтерфейсу, мова програмування також JavaScript. Висока швидкодія та стан компонентів.
- Angular: JavaScript-фреймворк для створення ВЗ, мови програмування JavaScript, TypeScript. Головні переваги його це двостороннє зв’язування даних та велика система.

1.6 Висновок першого розділу

У данному розділі ми провели детальний аналіз про важливість ВЗ та провели його порівняння з веб-сайтом і дійшли до висновку, що ВЗ зазвичай мають більшу можливість взаємодії з користувачем і забезпечують більш

розширені функціональність, ніж прості веб-сайти. Вони можуть мати складніші операції, виконувати обробку даних, забезпечувати інтерактивність, доступ до баз даних та інші функції. Також можуть бути розроблені з урахуванням конкретних потреб. Вони можуть працювати з персональними даними користувачів. Деякі ВЗ можуть працювати в режимі офлайн, що дозволяє користувачам використовувати їх навіть без підключення до Інтернету. Крім того, веб-застосунки можуть бути оптимізовані для швидкодії, що забезпечує миттєву відповідь. ВЗ можуть бути інтегрованими, щоб взаємодіяти з іншими програмами і системами. Вони можуть інтегруватись зі сторонніми сервісами, API, базами даних, електронною поштою, календарем та іншими ресурсами, що розширює їх можливості і функціональність.

Зробили вибір мови програмування та порівняли їх. Обрали для себе могу програмування Python по таких критеріях:

- Простота та зрозумілість.
- Велика кількість бібліотек і фреймворків.
- Зручна розробка прототипів.
- Підтримка.
- Переносимість.

Загалом, Python є популярним вибором для розробки веб-застосунків через свою простоту, широкі можливості та активну спільноту розробників.

Після вибору мови програмування ми обрали для себе також фрейворк з яким будемо працювати та зробили порівняння різних фреймворків, таких як Django, Pyramid і Flask. Враховуючи усі переваги та недоліки обрали один з цих фреймворків з ким будемо далі працювати.

Розглянули основні етапи розробки нашого ВЗ. Встановили фреймворк, задали маршрути для того, щоб кожен маршрут відповідав певній адресі, також пов’язаній функції, яка буде виконувати певні дії. Зробили функцію для додавання HTML-сторінок, для того, щоб наш додаток відображав шаблони. Також була обрана базу даних SQLAclhemy для збереження даних про результати тестування.

Було опрацьовано головні інструменти для дизайну нашого застосунку та інтерфейсу користувача, порівняли HTML/CSS, JavaScript, React та Angular. Опираючись на те, щоб інструмент був зрозумілим і зручним у використанні.

Загалом проведений аналіз допоможе нам сформувати технічне завдання для створення ВЗ для перевірки знань з математики, вибрати мову програмування, фреймворк, також розглянули головні етапи розробки ВЗ, опрацювалі головні інструменти для створення дизайну застосунку та інтерфейсу користувача.

2 РОЗДІЛ 2 ПРАКТИЧНА РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ПЕРЕВІРКИ ЗНАНЬ З МАТЕМАТИКИ

2.1 Створення серверної частини

Для створення серверної частини ми використовуємо застосунок «PYCharm», де створюємо новий проєкт (Див.рисунок 2.1).

«PyCharm» є розробленим компанією JetBrains інтегрованим середовищем розробки для мови програмування Python. Це потужний інструмент, який надає програмістам зручність та продуктивність при розробці програм на Python. «PyCharm» має широкий спектр функціональних можливостей, які допомагають прискорити процес розробки, покращити якість коду та забезпечити комфортне середовище для роботи. Завдяки своїм унікальним особливостям, простому інтерфейсу та підтримці різноманітних інструментів та фреймворків, «PyCharm» став незамінним інструментом для багатьох програмістів Python.

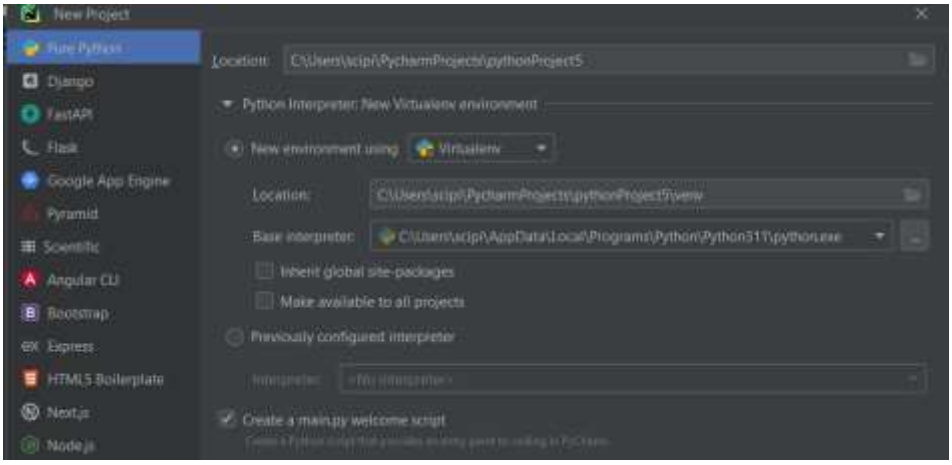


Рисунок 2.1 – Створення проєкту

Після чого ми встановлюємо Flask та задаємо маршрути для різних сторінок, таких як «home», «questions» і «about», щоб потім вказувати шлях, за яким буде доступна сторінка, та функція, яка буде виконуватись при отриманні запиту до цього шляху. В даному випадку, функції просто повертають

відповідні шаблони за допомогою функції «render_template()». Приклад в лістингу 2.1.

Лістинг 2.1 – Подання маршрутів

```
@app.route('/')
     @app.route('/home')
          def home():
          return render_template("home.html")
@app.route('/questions')
     def questions():
               return              render_template('questions.html',
     questions=questions)
```

«from flask import Flask, render_template, url_for»: Імпортуємо необхідні модулі з фреймворку Flask.

«from flask_sqlalchemy import SQLAlchemy»: Імпортуємо модуль SQLAlchemy, який дозволить нам працювати з базою даних.

«app = Flask(__name__)»: Створюємо екземпляр класу Flask з іменем app. Це є основним об'єктом нашого додатку.

«@app.route('/')» та «@app.route('/home')»: Декоратори, які вказують, що функція «home()» буде викликатись при отриманні запиту на кореневий шлях / або шлях «/home».

«def home():» функція, яка обробляє запити до шляхів «/» та «/home». У даному випадку, вона повертає шаблон home.html за допомогою функції «render_template()». Шаблон може бути візуалізований з допомогою HTML та інших шаблонізаторів.

Аналогічно для «/questions» та «/about»: Відповідні функції «questions()» та «about()» обробляють запити до шляхів «/questions» і «/about» відповідно.

«@app.route('/user/<string:name>/<int:id>')»: Декоратор, який вказує, що функція user() буде викликатись при отриманні запиту до шляху

«/user/<string:name>/<int:id>». В цьому шляху передаються два аргументи: name (який є рядком) та «id» (який є цілим числом).

«if __name__ == '__main__':»: Умова, яка перевіряє, чи файл є основним модулем, який виконується. Це дозволяє запустити сервер Flask, якщо файл запускається безпосередньо, а не імпортується як модуль.

«app.run()»: Запускає сервер Flask для обробки запитів.

Цей код встановлює маршрутизацію та обробку запитів у моєму ВЗ, вказуючи, яку функцію викликати для кожного шляху (Див.рисунок 2.2).

Рисунок 2.2 – Встановлення маршрутизації та обробка запитів

Як тільки ми встановили маршрути, створуємо відповідні HTML файли, для того, щоб розділити логіку додатку від представлення. Кожен HTML файл може представляти окрему сторінку, що спрощує розробку та підтримку коду.

Далі опишемо структуру головної сторінки нашого ВЗ.

<!DOCTYPE html> - це декларація, що вказує тип документа в мові HTML. Вона розташовується на початку HTML-документу і повідомляє браузеру, яку версію HTML використовує документ та як його слід інтерпретувати. DOCTYPE (Document Type) визначає стандарт або специфікацію HTML, яку необхідно дотримуватися при створенні документа. Використання <!DOCTYPE html> означає, що документ написаний у HTML5,

останньої версії мови HTML. Ця декларація важлива, оскільки допомагає браузерам правильно розуміти і відтворювати веб-сторінку. Вона дозволяє використовувати сучасні функції та елементи HTML5, а також забезпечує сумісність з попередніми версіями HTML, якщо браузери підтримують різні типи документів. Крім того, <!DOCTYPE html> також визначає режим стандарту, у якому браузер інтерпретує документ. Наприклад, режим строгого (strict) стандарту означає, що браузер буде дотримуватися точних правил та стандартів HTML, а режим переходу (quirks) може використовувати менш строгі правила для сумісності зі старими веб-сторінками. Загалом, <!DOCTYPE html> визначає тип документа HTML, який браузер має інтерпретувати, і гарантує правильне відображення та поведінку веб-сторінки.

«<!DOCTYPE html>» це декларація типу документа і вказує браузеру, що цей файл є HTML документом.

«<html lang="ua">» визначає початок HTML документу та встановлює мову сторінки на українську.

«<head>» Цей елемент містить метадані сторінки та посилання на зовнішні ресурси, такі як стилі CSS та скрипти JavaScript.

«<meta charset="UTF-8">» вказуватиме кодування символів, в даному випадку UTF-8, яке підтримує широкий спектр мов та символів.

«<title>{% block tittle %}{% endblock %}</title>» визначатиме заголовок сторінки, який може бути замінений відповідним контентом з блоку «{% block title %}{% endblock %}».

«<link rel="stylesheet" type="text/css" href="/static/css/menu.css">» посилання на зовнішній CSS файл, який містить стилі для веб-сторінки.

«<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css">» посилання на зовнішній CSS файл з бібліотекою Bootstrap, яка надає готові стилі та компоненти для розробки веб-інтерфейсу.

«<body>» визначатиме тіло сторінки, яке містить весь вміст, який буде видимий на веб-сторінці.

« » розпочинає контейнер для верхньої навігаційної панелі, яка містить посилання на головну сторінку та інші сторінки.

«<nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">» контейнер для навігаційних посилань.

«<a href="/" class="d-flex align-items-center link-body-emphasis text-decoration-none">» посилання на головну сторінку.

« » розпочинає контейнер для вмісту сторінки.

«<h1 class="display-5 fw-bold">Тест</h1>» заголовок сторінки.

« Після натиску на "Розпочати тестування", у вас відкриється наступна сторінка для проходження тесту перевірки знань з математики » текст, який описує вміст сторінки.

«<button class="btn btn-primary btn-lg" type="button" onclick="window.location.href='/questions'">Розпочати тестування</button>» Кнопка, яка перенаправляє користувача на сторінку з питаннями при натисканні.

«<footer class="container my-5 pt-5 text-body-secondary text-center text-small">» визначає підвал сторінки.

« © 2019–2023 Oleh Kotlinskyi » інформація про автора.

«<ul class="list-inline">» буде визначати ненумерований список посилань у підвалі.

«<li class="list-inline-item">Приватність» посилання на сторінку приватності.

«<li class="list-inline-item">Терміни» посилання на сторінку з термінами використання.

«<li class="list-inline-item">Допомога» посилання на сторінку допомоги.

Після розробки головної сторінки ми можемо відразу перейти до тестування, для цього натискаємо на «Розпочати тестування» і нас направляє на сторінку з тестуванням. Код запитань також зроблено за допомогою HTML/CSS для надання інтерфейсу (див.додаток А, А2, А3, А4, А5, А6, А7).

Сторінки «about» та «home» описані заголовком сторінки, які замінили відповідним контентом (Див.лістинг 2.2).

Лістинг 2.2 – Теги для розширення та повторного використання

```
{% extends "base.html" %}

    {% block tittle %}
    Про нас
    {% endblock %}

        {% block body %}
        <h1>Про нас</h1>
        {% endblock %}
```

У даному коді використовуються теги та конструкції з шаблонної системи Jinja, яка дозволяє розширювати та повторно використовувати HTML шаблони.

«{% extends "base.html" %}» означає, що цей шаблон наслідується від іншого шаблону з назвою "base.html". Це означає, що цей шаблон може використовувати блоки та змінні, які були визначені в базовому шаблоні.

«{% block title %} Про нас {% endblock %}» визначає блок з назвою "title" і задає текст "Про нас" для цього блоку. Цей блок може бути заміщений або розширений в базовому шаблоні.

«{% block body %} <h1>Про нас</h1> {% endblock %}» визначає блок з назвою "body" і містить HTML код з заголовком «<h1>Про нас</h1>». Цей блок також може бути заміщений або розширений в базовому шаблоні.

Загальна ідея полягає в тому, що шаблон "base.html" визначає загальну структуру та макет сторінки, а шаблон "about.html" розширює цей базовий

шаблон, заміщуючи або розширюючи блоки змісту та назви, щоб відображати відповідну інформацію про сторінку "Про нас".

2.2 Розробка фронтенду

Для розробки фронтенду використовуємо HTML/CSS, так як вони прості у використані і допоможуть зробити нам статичні об’єкти (Див.додаток Б, В, Д, Е).

CSS - це мова стилів, що використовується для опису зовнішнього вигляду веб-документів, написаних на мові розмітки, такій як HTML. Вона визначає, як елементи HTML повинні відображатися на екрані, включаючи параметри, такі як кольори, шрифти, розміри, межі, відступи, позиціонування та інші деталі стилізації. CSS дозволяє зручно та ефективно керувати зовнішнім виглядом веб-сторінок. Властивості CSS застосовуються до елементів HTML за допомогою селекторів, які вказують на конкретні елементи або групи елементів, до яких треба застосувати стилі. CSS також дозволяє встановлювати каскадні стилі, які успадковуються від батьківських елементів. Це дає можливість легко змінювати вигляд багатьох елементів одночасно, змінюючи стилі батьківського елемента. Використання CSS дозволяє розділити структуру HTML веб-документа від його представлення стилями. Це спрощує процес розробки та підтримки веб-сайтів, дозволяючи змінювати зовнішній вигляд сторінок без впливу на їх структуру. CSS є важливою складовою веб-розробки і використовується спільно з HTML та JavaScript для створення сучасних інтерактивних веб-сайтів та веб-додатків.

Взаємодія між HTML і CSS здійснюється за допомогою селекторів CSS. Селектори вказують на певні елементи або групи елементів HTML, до яких застосовуються стилі. Наприклад, селектор може охоплювати всі заголовки <h1> або елементи з класом "container". Правила CSS складаються з властивостей та їх значень. Властивості визначають конкретні аспекти стилізації, такі як "color" (колір тексту) або "font-size" (розмір шрифту), а значення встановлюють конкретні значення цих властивостей, наприклад, "red"

або "16px". CSS може бути вбудованим безпосередньо в HTML-документ за допомогою внутрішніх стилів, які включаються у тег <style>. Він також може бути зовнішнім, коли стилі знаходяться в окремому файлі CSS і підключаються до HTML-документа за допомогою тегу <link>. При обробці сторінки браузер зчитує як HTML, так і CSS. Він використовує структуру HTML для розміщення елементів на сторінці, а потім застосовує правила CSS для відображення їх з відповідним стилем. HTML і CSS співпрацюють, щоб забезпечити зовнішній вигляд сторінки, відповідний заданій стилізації.

Зараз розглянемо кожну частину нашого головного меню:

– «<body>» або ж тіло сторінки, "font-family:" задаємо шрифт для тексту всередині "<body>". У цьому випадку, використовується шрифт "Arial"."padding:" встановлюємо відступи всередині "<body>". "background-size" встановлює розмір фонового зображення. У цьому випадку, використовується значення cover, щоб зробити зображення заповнюючим всю площу."background-repeat:" встановлюємо поведінку повторення фонового зображення. В даному випадку, використовується значення no-repeat, щоб зображення не повторювалося."background-position:" встановлюємо положення фонового зображення. У цьому випадку, використовується значення center, щоб зображення було в центрі. "background-color:" встановлюємо колір фону як резервний варіант, якщо фонове зображення не завантажується. У даному випадку, використовується колір #f1f1f1.
– "container:" встановлюємо стилі для елементів з таким ж класом. "display: flex:" встановлюємо режим відображення елемента як гнучкого контейнера. "flex-direction: column:" встановлюємо напрямок гнучкого контейнера на вертикальний, щоб дочірні елементи відображалися один за одним в стовпчик. "align-items: center:" вирівнює дочірні елементи по центру контейнера. "background-color:" встановлюємо колір фону для елемента з класом container.

– "h1:" встановлюємо стилі для заголовків рівня.
– "test-heading:" робимо стилі для елементів з таким ж класом.
– "question:" створюємо стилі для елементів з класом question.
– "question p:" будуємо стилі для параграфів всередині елементів з класом "question".
– "label:" встановлюємо стилі для елементів <label>.
– "button[type="submit"]:" опрацьовуємо стилі для кнопок з атрибутом type встановленим на значення "submit".
– "button[type="submit"]:hover:" робимо стилі для кнопки з атрибутом "type" встановленим на значення "submit" при наведенні на неї курсору.

Цей CSS код можна використовувати для стилізації сторінок HTML, де використовуються елементи, класи та атрибути, які вказані в коді. Далі задаємо стилізації кнопці «Розпочати тестування» на головному меню (Див.лістинг 2.3).

Лістинг 2.3 – Кнопка «Розпочати тестування»

```
.btn {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
.btn:hover {
  background-color: #0069d9;
}
```

Надали стилізацію кнопці на головному меню нашого ВЗ, далі перейдемо до наступної сторінки, а саме самого тестування. Створивши файл «styles.css»

та додавши його у відповідний html файл ми задамо інтерфейс самому тестуванню (Див.рисунок 2.3).

```css
/* Заголовок */
h1 {
    text-align: center;
    color: #333;
}

/* Контейнер запитань */
.questions {
    margin: 20px auto;
    width: 500px;
}

/* Один запитання */
.question {
    margin-bottom: 20px;
}

/* Текст запитання */
.question p {
    font-size: 18px;
    color: #555;
}

/* Поле вводу відповіді */
.question input {
    width: 100%;
    padding: 5px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
```

Рисунок 2.3 – Інтерфейс тестування

Отже використавши CSS для розробки інтерфейсу користувача ми можемо тепер розглянути зовнішній вигляд нашого ВЗ та як він виглядає зі сторони ВК.

2.3 Результати веб-застосунку

Ми зробили серверну частину нашого ВЗ на базі мови програмування Python з підключеним фреймворком Flask та розробили фронтенд на CSS. Отже

тепер можемо розглянути повністю застосунок. Для початку запускаємо наш застосунок та переходимо за посиланням (Див.рисунок 2.4).

Рисунок 2.4 – Посилання на веб-застосунок

Після того, як ми перейшли за посиланням ми бачимо перед собою головне меню нашого застосунку, де у нас є шапка нашого застосунку, саме менюта дно (Див.додаток Ж). Також відразу у нас є кнопка для того, щоб розпочати тестування (Див.рисунок 2.5).

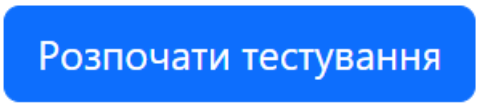


Рисунок 2.5 – «Розпочати тестування»

Як тільки ми перейшли, у нас відразу відображаються запитання з варіантами відповіді. Вони можуть бути, як розгорнутого типу, де потрібно дати відповідь текстом на запитання, так само і з вибором однієї або декількох правильних відповідей. Також, якщо ВК не обере або не запише відповідь хоча б на одне із запитань то отримає сповіщення про те, що необхідно заповнити поле (Див.рисунок 2.6). В самому тестувані вказані випадкові 10 запитань з математики різного характеру.

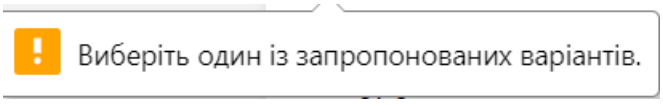


Рисунок 2.6 – Обов’язкова відповідь

Запитання зроблені таким чином, щоб будь-який ВК міг для себе самостійно підібрати та записати запитання орієнтуючись на свої знання та

вподобання.Після натискання на клавішу «Перевірити». Ми відразу отримаємо результат нашого тестування у вспливаючому вікні (Див.рисунок 2.7), де побачимо на яку кількість запитань ми відповіли правильно.

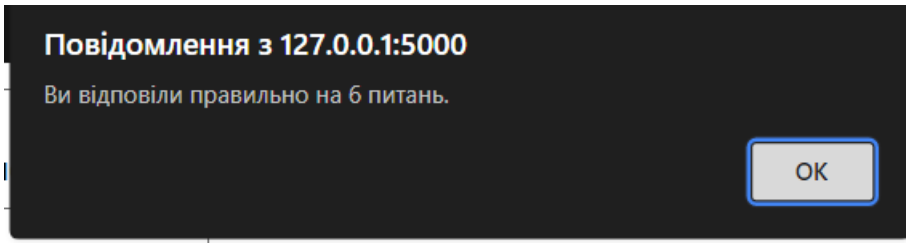


Рисунок 2.7 – Перевірка відповідей

Таким чином ми з легкістю зможемо проходити тестування знову і знову для отримання позитивного результату та перевірки своїх знань та здібностей.

2.4 Висновок до другого розділу

У даному розділі було розглянуто процес створення серверної частини нашого ВЗ на базі мови програмування Python з використанням фреймворка Flask, розробка фронтенду за допомогою CSS та проходження і перевірка результатів нашого тестування, для того аби забезпечити ефективне проходження тестування для перевірки знань з математики.

Спочатку розробили серверну частину нашого застосунку. Для створення серверної частини ми використовуємо застосунок «PYCharm». Також розробили маршрути до різних сторінок і зробили їх наповнення на html.

Далі розглянули для себе CSS та розробили фронтенд нашого ВЗ, щоб ми могли взаємодіяти з користувачем та забезпечили його функціоналом для проходження тестування для перевірки знань з математики. Додали сторінку головного меню та сторінку самого тестування.

Опрацювавши результат нашого застосунку ми тепер можемо проходити тестування для перевірки знань з математики та в подальшому використовувати його і змінювати під власні побажання та знання.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Питання щодо безпеки життєдіяльності

Захист персональних даних відіграє дуже важливу роль, особливо при перевірці знань у нашому застосунку. Тому варто зазначити такі пункти щодо захисту користувачів:

Конфіденційність: Забезпечення конфіденційності персональних даних означає, що доступ до цих даних має бути обмежений лише авторизованим особам. Використовуйте механізми автентифікації та авторизації, щоб переконатися, що тільки відповідні користувачі мають доступ до особисто ідентифікованих даних.

Шифрування: Застосовуйте шифрування для захисту персональних даних під час їх передачі між користувачем і сервером. Використовуйте протокол HTTPS з SSL / TLS шифруванням для забезпечення безпеки під час передачі даних через мережу Інтернет.

Зберігання даних: Забезпечте безпечне зберігання персональних даних на сервері. Використовуйте належні методи хешування та шифрування для зберігання паролів користувачів та інших конфіденційних даних.

Політика конфіденційності: Розробіть і публікуйте політику конфіденційності, в якій чітко визначаються правила використання та зберігання персональних даних користувачів. Повідомте користувачів про те, які дані збираються, як вони використовуються та як їх можна захистити.

Обмеження доступу: Мінімізуйте доступ до персональних даних, надавайте доступ лише до тих даних, які необхідні для функціонування веб-застосунку. Не зберігайте надмірну кількість персональних даних.

Аудит та моніторинг: Встановіть механізми аудиту та моніторингу для виявлення можливих порушень безпеки даних. Це допоможе вчасно виявляти та реагувати на потенційні загрози безпеки.

Врахування законодавства: Дотримуйтесь вимог законодавства щодо захисту персональних даних, зокрема загального регламенту про захист персональних даних (GDPR) у Європейському Союзі.

Враховуючи цей перелік ми зможемо зазначити, що веб застосунок відповідає стандартам і приватність користувачів буде захищена.

Захист від шкідливих завантажень є важливим аспектом при розробці веб-застосунку з математики. Ось декілька практик безпеки, які можна використовувати для забезпечення захисту від шкідливих завантажень:

- Валідація файлів: Перевірте типи файлів, які користувачі можуть завантажувати на сервер. Використовуйте механізми валідації файлових розширень та MIME-типів, щоб перевірити, чи відповідають завантажені файли допустимим форматам, таким як PDF, JPEG або PNG. Використання бібліотек або фреймворків, які мають вбудовану функціональність валідації файлів, може спростити цей процес.
- Очистка та перевірка файлів: Перед збереженням завантажених файлів на сервері слід провести очищення та перевірку файлів. Видаліть небезпечні символи та скриптовий код з імен файлів. Використовуйте антивірусне програмне забезпечення для перевірки завантажених файлів на віруси або шкідливий код. Також можна обмежити розмір файлів або встановити максимальний розмір завантаження для запобігання завантаженню великих файлів, які можуть становити загрозу безпеці.
- Збереження файлів у безпечному місці: Завантажені файли слід зберігати у безпечному розташуванні на сервері, відокремленому від основних системних файлів. Це допоможе запобігти виконанню завантажених файлів, які можуть містити шкідливий код.

- Оновлення та патчі: Періодично оновлюйте всі компоненти, які використовує ваш веб-застосунок, такі як бібліотеки або фреймворки. Слід слідкувати за виходом оновлень та патчів безпеки і вчасно їх встановлювати, щоб усунути вразливості, які можуть бути використані для атак на завантаження файлів.
- Обмеження доступу до завантажених файлів: Забезпечте обмежений доступ до завантажених файлів, щоб уникнути можливості прямого доступу до них через URL. Використовуйте права доступу до файлів та налаштування сервера для забезпечення обмеженого доступу лише до авторизованих користувачів.
- Моніторинг та журналювання: Ведіть журнал подій щодо завантаження файлів, щоб виявити потенційні загрози безпеці. Моніторіть активність користувачів та шукайте ознаки підозрілої діяльності, наприклад, незвичайно велику кількість завантажень або спроби завантажити файл з підозрілим іменем.

Загальною метою є запобігання завантаженню шкідливих або небезпечних файлів, а також забезпечення безпеки системи та даних користувачів, використовуючи комбінацію валідації, очищення, перевірок та обмежень на рівні сервера.

3.2 Питання з основ хорони праці

Важливо також дотримувати і основ охорони праці для працівників, щоб забезпечити їх безпекою та здоров'ям коли вони працюють над проектом. Це включає розгляд таких напрямків:

Робоче середовище має бути забезпечене усіма необхідними умовами. Це означає належну освітленість, оптимальний температурний режим, ефективну вентиляцію та дотримання ергономічних принципів. Необхідно уникати надмірного навантаження на м'язи та спину працівників.

Забезпечення обладнанням, яке відповідає вимогам безпеки. Регулярна перевірка обладнання на відповідність стандартам безпеки та його планове обслуговування є важливими аспектами. Надійне та безпечне обладнання допомагає уникнути можливих небезпек і травм.

Завчасне виявлення потенційних ризиків, пов'язаних з розробкою веб-застосунку. Наприклад, резервне копіювання даних та застосування механізмів для запобігання втраті інформації. Ретельний аналіз можливих загроз і прийняття відповідних заходів допоможе запобігти непередбачуваним проблемам.

Забезпечення розробників достатнім рівнем знань про охорону праці. Надання необхідної освіти та навчання з питань безпеки праці допомагає зрозуміти принципи та виконувати необхідні процедури для забезпечення безпечного робочого середовища.

Планування екстрених ситуацій. Важливо мати план дій у разі виникнення непередбачуваних ситуацій, таких як пожежа або аварія. Це включає наявність засобів евакуації, планів наведення порядку, наявність необхідного медичного приладдя та засобів пожежогасіння.

Дотримання та застосування цих заходів щодо охорони праці під час розробки веб-застосунку забезпечить працівників від усіх ризиків та створить безпечне та здорове робоче середовище.

3.3 Висновок до третього розділу

У даному розділі роботи були розглянуті основи охорони праці, а саме основи охорони праці для працівників і з безпеки життєдіяльності - захист персональних даних. Сама розробка веб-застосунку повинна відбуватися згідно нормам безпеки і охорони праці.

Необхідно встановити процедури безпеки для користувачів веб-застосунку. Це може включати регулярне оновлення програмного забезпечення,

шифрування даних, використання міцних паролів та захист від несанкціонованого доступу.

Захист персональних даних користувачів є надзвичайно важливим аспектом. Розробка веб-застосунку повинна дотримуватися вимог законодавства про захист персональних даних та використовувати механізми шифрування та анонімізації даних.

Запобігання виникненню травм і аварій. Розробка веб-застосунку повинна враховувати можливі ризики і вживати заходів для їх попередження. Наприклад, управління ризиками безпеки даних, проведення регулярних аудитів безпеки та навчання персоналу правилам безпеки.

Впровадження ефективної системи управління безпекою. Застосування методів оцінки ризиків, регулярні аналізи та вдосконалення процесів допоможуть забезпечити безпеку життєдіяльності та охорону праці.

Тобто, охорона праці та безпека життєдіяльності є дуже важливими при розробці веб-застосунку. Дотримання усіх вимог забезпечить успішну роботу для розробників та користувачів.

ВИСНОВКИ

Головною метою кваліфікаційної роботи було створення такого середовища, що допоможе користувачам перевірити свої знання з математики та інших тем.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» визначили важливість нашого застосунку на основі порівняльної таблиці із веб-сайтом і визначили, що ВЗ для перевірки знань з математики може значно полегшити процес навчання та забезпечити користувачам зручні, інтерактивні та персоналізовані можливості для вдосконалення своїх математичних навичок.

Далі ми розглянули, які є мови програмування для створення ВЗ та порівняли їх, щоб обрати для себе ту, яка підійде для виконання моєї роботи. Для себе я обрав Python, тому що він має привабливий та легкий для сприйняття синтаксис, що робить його привітним для новачків та спрощує процес розробки та підтримки коду. Python також пропонує широкий вибір функціональних бібліотек та фреймворків, таких як Django, Flask, Pyramid та інші, які надають готові рішення для рутинних задач і сприяють прискоренню процесу розробки.

Опрацював, які є фреймворки для створення ВЗ на базі мови програмування Python, провівши також їх порівняння було обрано фреймворк Flask. Так як він має ряд переваг, які і посприяли кінцевому вибору.

Було розглянуто етапи розробки ВЗ, а саме:

– Як встановвити фреймворк;

– Як визначити маршрути;

– Як наповнити сторінку застосунку.

Далі розглядав інтерфейс користувача, яким він має бути та що найкраще використати для його створення, порівнювалися такі застосунки, як HTML/CSS, JavaScript, React, Angular.

У другому розділі на основі практичної розробки ВЗ для перевірки знань з математики було зроблено серверну частину проєкту, фронтент застосунку та перевірка кінцевого результату.

При створені серверної частини ми використовували мову програмування Python з використанням фреймворка Flask. Де були визначені всі маршрути та заповнено сторінки нашого застосунка на базі HTML.

Розробили фронтенд на основі CSS та надали інтерфейс користувача для нашого ВЗ. Для того, щоб користувач міг одразу зорієнтуватися по застосунку і відразу почати проходження тестування з математики.

Опрацювавши результат нашого застосунку ми тепер можемо проходити тестування для перевірки знань з математики та в подальшому використовувати його і змінювати під власні побажання та знання.

У третьому розділі було розглянуто з безпеки життєдіяльності захист персональних даних та захист від шкідливих завантажень, їх важливість та вплив при створені ВЗ. З основ охорони праці розглянули саме охорону праці самих працівників, а саме забезпечити їх безпекою та здоров'ям коли вони працюють над проектом.

З цього зробили висновок, що важливими усі ці питання є важливими, так як вони впливають на кінцевий результат при створені ВЗ для перевірки знань з математики.

ПЕРЕЛІК ДЖЕРЕЛ

1. JavaScript.Info – Електронний ресурс – Режим доступу: URL: https://uk.javascript.info/intro
2. SunSoft. Чим веб-додаток відрізнається від сайту – Електронний ресурс – Режим доступу: URL: http://sunsoft.com.ua/uk/Article/38
3. WebCase – Електронний ресурс – Режим доступу: URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/#f1
4. Pythonguide – Електронний ресурс – Режим доступу: URL: https://pythonguide.rozh2sch.org.ua
5. Bootsrapcdn – Електронний ресурс – Режим доступу: URL: https://www.bootstrapcdn.com
6. Djangogirls – Електронний ресурс – Режим доступу: URL: https://tutorial.djangogirls.org/uk/django/
7. Ukwikipedia.com – Що таке Flask – Електронний ресурс – Режим доступу: URL: https://uk.wikipedia.org/wiki/Flask
8. CSS.com.ua – Як працювати з html – Електронний ресурс – Режим доступу: URL: https://css.in.ua/article/shcho-take-html_10
9. Digitalocean – Як правильно задавати маршрути та користуватися завданнями на Flask – Електронний ресурс – Режим доступу: URL: https://www.digitalocean.com/community/tutorials/how-to-use-templates-in-a-flask-application
10. Pymbook – how to use arguments Flask – Електронний ресурс – Режим доступу: URL: https://pymbook.readthedocs.io/en/latest/flask.html
11. Vegibit – How to create routes Flask – Електронний ресурс – Режим доступу: URL: https://vegibit.com/how-to-create-routes-in-flask/
12. MySql – Електронний ресурс – Режим доступу: URL: https://www.mysql.com

13. Dou.ua – Що потрібно знати про алгоритми – Електронний ресурс – Режим доступу: URL: https://dou.ua/lenta/articles/what-you-should-know-about-algorithms/
14. Otakoyi – Особливості створення веб-систем – Електронний ресурс – Режим доступу: URL: https://otakoyi.ua/osoblyvosti-stvorennya-web-system-z-dopomohoyu-mean-stek
15. Pythonguide – Електронний ресурс – Режим доступу: URL: https://pythonguide.rozh2sch.org.ua
16. Hasdork – Електронний ресурс – Режим доступу: URL: https://hashdork.com/uk/best-web-application-languages/
17. Apix-Drive – Що таке JSON – Електронний ресурс – Режим доступу: URL: https://apix-drive.com/ua/blog/useful/scho-take-json
18. Flask-sqlalchemy – Електронний ресурс – Режим доступу: URL: https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/
19. React.Dev – Електронний ресурс – Режим доступу: URL: https://react.dev/learn
20. ITWDN – Електронний ресурс – Режим доступу: URL: https://itvdn.com/ru/blog/article/what_angular
21. Jetbrains – Електронний ресурс – Режим доступу: URL: https://www.jetbrains.com/pycharm/download/#section=windows
22. CSS.in.ua – Електронний ресурс – Режим доступу: URL: https://css.in.ua/html/tag/doctype
23. tutorial.drjango – Електронний ресурс – Режим доступу: URL: https://tutorial.djangogirls.org/en/template_extending/
24. Hi-news.pp – Електронний ресурс – Режим доступу: URL: http://hi-news.pp.ua/kompyuteri/10069-yak-pdklyuchati-css-html-statika-dinamka-veb-stornki.html
25. Nicepage – CSS Templates – Електронний ресурс – Режим доступу: URL: https://nicepage.com/css-templates

26. wiki.legalaid.com.ua – Електронний ресурс – Режим доступу: URL: https://wiki.legalaid.gov.ua/index.php/Захист_персональних_даних
27. manageengine – Електронний ресурс – Режим доступу: URL: https://www.manageengine.com/browser-security/knowledge-base.html
28. eset.com – Електронний ресурс – режим доступу: URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/zashchita-ot-vredonosnykh-programm/
29. cuspu.edu – Електронний ресурс – Режим доступу: URL: https://phm.cuspu.edu.ua/images/Основи_охорони_праці_2017.pdf
30. vin.gov.ua – Електронний ресурс – Режим доступу: URL: https://www.vin.gov.ua/upravlinnia-u-spravakh-natsionalnostei-ta-relihii/55073-zakhyst-vashykh-gadzhetiv-vid-shkidlyvykh-prohram

ДОДАТКИ

А

Контрольні запитання html

```
<!DOCTYPE html>
<html>
<head>
  <style>
     body {
      background-color: #2ee0e0;
    }
  body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      padding: 20px;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
    }

    .container {
      max-width: 600px;
      background-color: #fff;
      padding: 20px;
      border-radius: 5px;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }

    h1 {
      text-align: center;
      color: #333;
    }

    .question {
      margin-bottom: 20px;
    }

    .question p {
      font-weight: bold;
      margin-bottom: 10px;
    }
```

```
    input[type="radio"],
    input[type="text"],
    input[type="checkbox"] {
      margin-bottom: 10px;
    }

    input[type="submit"] {
      display: block;
      width: 100%;
      padding: 10px;
      background-color: #4caf50;
      color: #fff;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }

    .result {
      margin-top: 20px;
      text-align: center;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Перевірка знань з математики</h1>

    <div class="question">
      <p>1. Чому дорівнює корінь квадратний з 36?</p>
      <input type="radio" name="answer1" value="a">
      <label for="answer1">a) 5</label><br>
      <input type="radio" name="answer1" value="b">
      <label for="answer1">b) 7</label><br>
      <input type="radio" name="answer1" value="c">
      <label for="answer1">c) 6</label>
    </div>

   <style>

    body {
```

A3

```
  font-family: Arial, sans-serif;
  background-color: #f2f2f2;
  padding: 20px;
}

.container {
  max-width: 600px;
  margin: 0 auto;
  background-color: #fff;
  padding: 20px;
  border-radius: 5px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

h1 {
  text-align: center;
  color: #333;
}

.question {
  margin-bottom: 20px;
}

.question p {
  font-weight: bold;
  margin-bottom: 10px;
}

input[type="radio"],
input[type="text"],
input[type="checkbox"] {
  margin-bottom: 10px;
}

input[type="submit"] {
  display: block;
  width: 100%;
  padding: 10px;
  background-color: #4caf50;
  color: #fff;
  border: none;
```

```
 border-radius: 5px;
      cursor: pointer;
    }

    .result {
      margin-top: 20px;
      text-align: center;
      font-weight: bold;
    }
  </style>
</head>
  <title>Перевірка правильних відповідей</title>
</head>
<body>
  <h1>Перевірка правильних відповідей</h1>

  <form>
    <div class="question">
      <p>1. Чому дорівнює корінь квадратний з 36?</p>
      <input type="radio" name="answer1" value="a" required>
      <label for="answer1">a) 5</label><br>
      <input type="radio" name="answer1" value="b">
      <label for="answer1">b) 7</label><br>
      <input type="radio" name="answer1" value="c">
      <label for="answer1">c) 6</label>
    </div>

    <div class="question">
      <p>2. Дано квадрат, одна його сторона дорівнює 5см. Знайдіть його площу.</p>
      <input type="text" name="answer2" required>
    </div>

    <div class="question">
      <p>3. Скільки буде 4 помножити на 2?</p>
      <input type="text" name="answer3" required>
    </div>

    <div class="question">
      <p>4. Яке число є наступним після 10?</p>
      <input type="text" name="answer4" required>
    </div>
```

А5

```
<div class="question">
    <p>5. Яке число є перед 7?</p>
    <input type="text" name="answer5" required>
  </div>

  <div class="question">
    <p>6. Яка сума чисел 5 + 4?</p>
    <input type="text" name="answer6" required>
  </div>

  <div class="question">
    <p>7. Яка різниця чисел 12 - 8?</p>
    <input type="text" name="answer7" required>
  </div>

  <div class="question">
    <p>8. Скільки буде 6 помножити на 3?</p>
    <input type="text" name="answer8" required>
  </div>

  <div class="question">
    <p>9. Яке число є наступним після 15?</p>
    <input type="text" name="answer9" required>
  </div>

  <div class="question">
    <p>1 або 2 відповіді. Яке або які з наведених чисел діляться на 3 :</p>
    <input type="checkbox" name="colors" value="red"> 125<br>
    <input type="checkbox" name="colors" value="blue"> 560<br>
    <input type="checkbox" name="colors" value="green"> 135<br>
    <input type="checkbox" name="colors" value="ghost"> 282<br>
  </div>

  <button type="submit">Перевірити</button>
</form>

<script>
  document.querySelector('form').addEventListener('submit', function(event) {
    event.preventDefault(); // Зупиняємо стандартну відправку форми

    // Отримуємо значення відповідей з форми
```

```
var answer1 = document.querySelector('input[name="answer1"]:checked');
      var answer2 = document.querySelector('input[name="answer2"]').value;
var answer3 = document.querySelector('input[name="answer3"]').value;
      var answer4 = document.querySelector('input[name="answer4"]').value;
      var answer5 = document.querySelector('input[name="answer5"]').value;
      var answer6 = document.querySelector('input[name="answer6"]').value;
      var answer7 = document.querySelector('input[name="answer7"]').value;
      var answer8 = document.querySelector('input[name="answer8"]').value;
      var answer9 = document.querySelector('input[name="answer9"]').value;
      var colors = document.querySelectorAll('input[name="colors"]:checked');

      // Перевірка правильних відповідей
      var correctAnswers = 0;
      if (answer1 && answer1.value === "c") {
        correctAnswers++;
      }
      if (answer2 === "25") {
        correctAnswers++;
      }
      if (answer3 === "8") {
        correctAnswers++;
      }
      if (answer4 === "11") {
        correctAnswers++;
      }
      if (answer5 === "6") {
        correctAnswers++;
      }
      if (answer6 === "9") {
        correctAnswers++;
      }
      if (answer7 === "4") {
        correctAnswers++;
      }
      if (answer8 === "18") {
        correctAnswers++;
      }
      if (answer9 === "16") {
        correctAnswers++;
      }
      if (colors.length === 2) {
```

А7

```
correctAnswers++;
      }

      // Виведення результату
      var resultMessage = "Ви відповіли правильно на " + correctAnswers + " питань.";
      alert(resultMessage);

      // Скидання форми
      document.querySelector('form').reset();
    });
  </script>
</body>
</html>
```

Маршрути від головного меню – main.py

Додаток Б

```
from flask import Flask, render_template, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)


@app.route('/')
@app.route('/home')
def home():
    return render_template("home.html")


@app.route('/questions')
def questions():
    return render_template('questions.html',
questions=questions)



@app.route('/about')
def about():
    return render_template("about.html")


@app.route('/user/<string:name>/<int:id>')
def user(name, id):
    return "user name " + name + " - " + str(id)


if __name__ == '__main__':
    app.run()
```

Код css кнопки запуску тестування

Додаток В

```
.btn {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.btn:hover {
  background-color: #0069d9;
}
```

Вигляд CSS сторінки з тестуванням

Додаток Д

```
/* Заголовок */
h1 {
    text-align: center;
    color: #333;
}

/* Контейнер запитань */
.questions {
    margin: 20px auto;
    width: 500px;
}

/* Один запитання */
.question {
    margin-bottom: 20px;
}

/* Текст запитання */
.question p {
    font-size: 18px;
    color: #555;
}

/* Поле вводу відповіді */
.question input {
    width: 100%;
    padding: 5px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

/* Кнопка відправки */
.btn-submit {
    display: block;
    margin: 20px auto;
    padding: 10px 20px;
    font-size: 16px;
    color: #fff;
    background-color: #337ab7;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

.btn-submit:hover {
    background-color: #23527c;
}
```

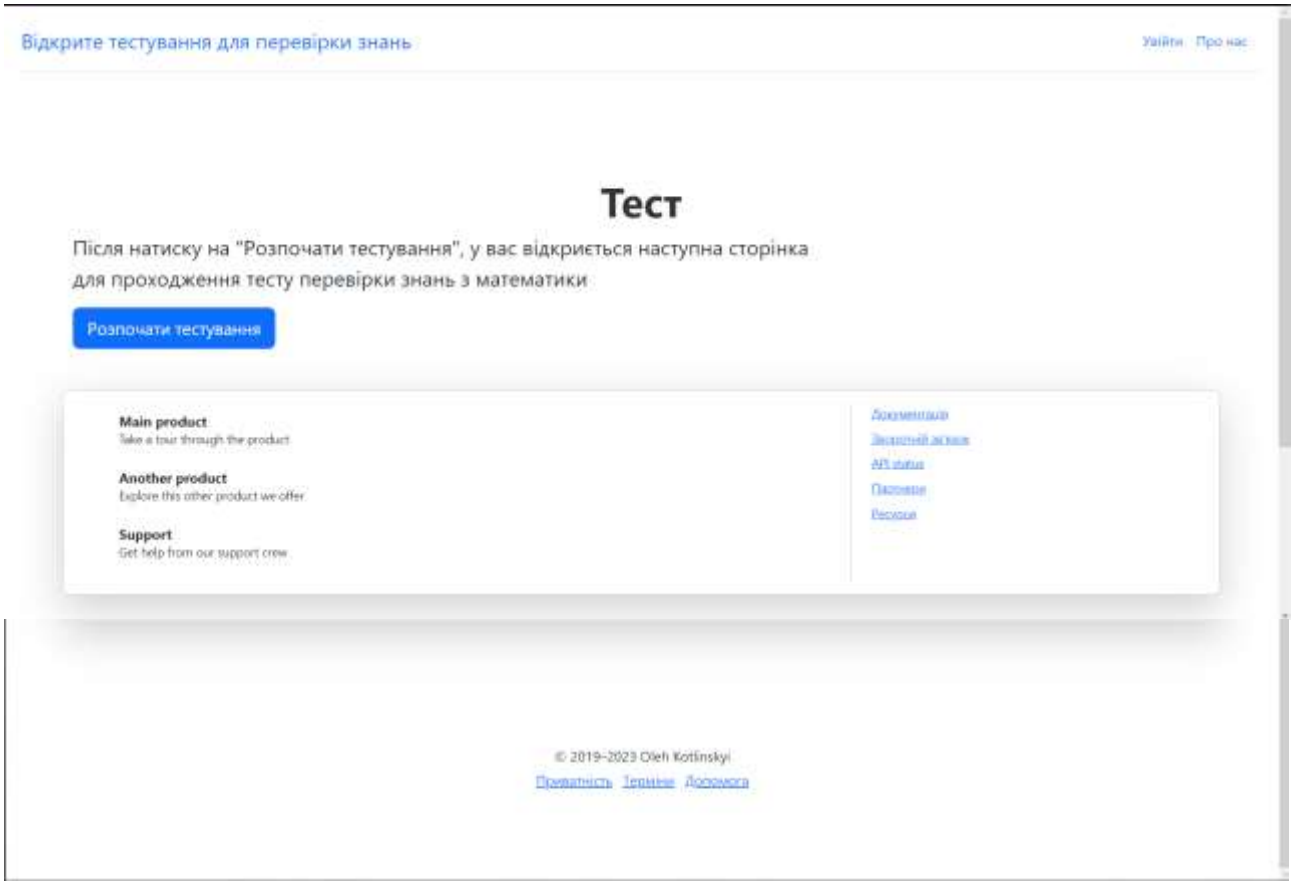
Код головного меню – main.css

Додаток Е

```
body {
    font-family: Arial, sans-serif;
    padding: 20px;
    background-image:
url("https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.rawpixel.com%2Fsearch
%2Fabstract%2520background&psig=AOvVaw12xglUtLeZoO5Mf61C0_1n&ust=168697482680600
0&source=images&cd=vfe&ved=0CBEQjRxqFwoTCNDv3ov1xv8CFQAAAAAdAAAAABAE"); /*
замініть "background.jpg" на шлях до вашого зображення */
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center;
    background-color: #f1f1f1; /* колір фону як резервний варіант */
}

.container {
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: rgba(255, 255, 255, 0.8); /* колір фону контейнера */
    padding: 20px;
}

h1 {
    text-align: center;
    color: #333;
    margin-bottom: 20px;
}

.test-heading {
    text-align: center;
    margin-bottom: 20px;
}

.question {
    margin-bottom: 20px;
}

.question p {
    font-weight: bold;
}

label {
    display: block;
    margin-bottom: 10px;
}

button[type="submit"] {

 padding: 10px 20px;
    background-color: #337ab7;
    color: #fff;
    border: none;
    cursor: pointer;
}

button[type="submit"]:hover {
    background-color: #23527c;
}
```

Головне меню та дно застосунку

Додаток Ж



Сторінка з тестами

Додаток З

Перевірка правильних відповідей

1. Чому дорівнює корінь квадратний з 36?

- a) 5
- b) 7
- c) 6

2. Дано квадрат, одна його сторона дорівнює 5см. Знайдіть його площу.

3. Скільки буде 4 помножити на 2?

4. Яке число є наступним після 10?

5. Яке число є перед 7?

6. Яка сума чисел 5 + 4?

7. Яка різниця чисел 12 - 8?

8. Скільки буде 6 помножити на 3?

8. Скільки буде 6 помножити на 3?

9. Яке число є наступним після 15?

1 або 2 відповіді. Яке або які з наведених чисел діляться на 3 :

- ☐ 125
- ☐ 560
- ☐ 135
- ☐ 282

Перевірити