Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до бакалаврської дипломної роботи

бакалавр

(ступінь вищої освіти)

на тему: «Розробка веб-системи для вивчення основ web-програмування»

Виконав: студент 4 курсу
групи ЗПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Мазуренко Ю.С.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)
Рецензент: д.т.н., доц. каф. КН Арсенюк І.Р.
(прізвище та ініціали)

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти – бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"_25_" __березня__2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мазуренко Юрію Сергійовичу

1. Тема роботи – розробка веб-системи для вивчення основ web-програмування.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від "_24_" _березня_2022 року №_66._

2. Строк подання студентом роботи _____13 червня 2022 року_____

3. Вихідні дані до роботи: середовище розробки Sublime Text, мова розробки JavaScript, операційна система – Windows 10, система управління базою даних PhpMyAdmin, програма-збирач Prepros, для компіляції препроцесора SASS синтаксису SCSS.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану навчальних web-систем для вивчення web-програмування; порівняльний аналіз аналогів; розробка структури інтерфейсу web-системи; розробка методу тестування для користувача; розробка моделі роботи web-системи; розробка алгоритмів роботи web-системи; розробка програмних засобів для впровадження

адаптивної верстки; розробка блоку реєстрації; розробка блоку тестування; тестування web-системи; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу: графічний інтерфейс веб-системи; метод тестування для користувача; модель роботи веб-системи; блок-схеми алгоритмів роботи веб-системи; тестування веб-системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ракитянська Г.Б., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання _____25 березня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
	Аналіз стану навчальних web-систем для вивчення web-програмування	26.03.2022 – 10.04.2022	Вик.
	Розробка алгоритмів web-системи	11.04.2022 – 01.05.2022	Вик.
	Розробка блоку реєстрації	02.05.2022 – 18.05.2022	Вик.
	Розробка блоку тестування	19.05.2022 – 10.06.2022	Вик.

Студент _____ Мазуренко Ю.С.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Ракитянська Г.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 117 сторінок формату А4, на яких є 81 рисунок, 7 таблиць, список використаних джерел містить 23 найменування.

У процесі виконання бакалаврської дипломної роботи проведено детальний аналіз навчальних web-систем для вивчення web-програмування. Встановлено об’єкт, предмет, завдання та методи дослідження.

У результаті було розроблено спеціалізовану навчальну web-систему «StudyWEB».

Розроблено модель роботи системи та метод тестування користувача. Модель дозволяє підвищити ефективність навчання користувача шляхом об’єднання теоретичних та практичних матеріалів для вивчення технологій web-програмування, таких як HTML CSS та JS в одній навчальній web-системі. Реалізований метод дозволяє проводити тестування знань користувача без необхідності у серверній частині.

Створену web-систему реалізовано з використанням мови розмітки гіпертексту HTML, каскадних таблиць стилів CSS та скриптової метамови SASS синтаксису SCSS, мов програмування JavaScript та PHP в середовищі розробки Sublime Text.

Ключові слова: web-розробка, навчальна система, технології web-програмування.

ABSTRACT

The bachelor's thesis consists of 117 A4 pages, which have 81 figures, 7 tables, the list of used sources contains 23 titles.

In the process of completing the bachelor's thesis, a detailed analysis of educational web-systems for the study of web-programming. The object, subject, tasks and research methods are established.

As a result, a specialized training web-system "StudyWEB" was developed.

The model of system operation and the method of user testing have been developed. The model allows to increase the efficiency of user training by combining theoretical and practical materials for studying web programming technologies, such as HTML CSS and JS, in one educational web-system. The implemented method allows testing the user's knowledge without the need for the server part.

The created web-system is implemented using HTML hypertext markup language, cascading CSS style sheets and SASS SCASS syntax metalanguage, JavaScript and PHP programming languages in the Sublime Text development environment.

Keywords: web-development, educational system, web-programming technologies.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Web-програмування, або ж web-розробка є популярною серед тих, хто бажає спробувати себе у сфері інформаційних технологій (IT), адже це доволі затребуваний напрямок. На сьогодні свою власну web-систему хочуть мати не лише великі компанії та установи, але й окремі інтернет-користувачі, як от блогери чи фрілансери, що робить цей напрямок ще більш перспективним. Крім того, у зв'язку з глобалізацією та стрімким розвитком технологій, все більш актуальною стає тема web-розробки.

Web-розробка – процес створення web-систем. Основними етапами створення систем є дизайн, верстка сторінок, програмування на стороні клієнта та на стороні сервера, а також конфігурування в web-сервері [1].

Однак для освоєння web-програмування необхідно опанувати принаймні базові технології, такі як HTML, CSS та JavaScript. Причиною, через яку потенційні новачки відмовляються від вивчення web-програмування є складність процесу самостійного знаходження справді важливої інформації та освоєння матеріалу. Вирішенням такої задачі є використання спеціалізованих навчальних систем, націлених на полегшення процесу вивчення технологій web-програмування.

Навчальні сайти або ж онлайн-курси є найбільш популярними серед користувачів, так як, на відміну від програмних додатків, не потребують встановлення на пристрій користувача, а завдяки адаптивності доступні на усіх пристроях. Саме тому для реалізації було обрано інструменти розробки програмних засобів спеціалізованої навчальної web-системи.

Наразі існує багато варіантів реалізацій навчальних web-додатків для вивчення програмування. Усі вони мають свої недоліки, серед яких надання користувачу доступу до неповного обсягу матеріалів, без преміум-підписки, або ж розрізненість курсів з виокремленням певної технології, що є незручним для

користувача, адже йому необхідно самостійно знаходити матеріали для подальшого освоєння своєї спеціалізації.

Тому розробка програмних засобів власної навчальної системи для вивчення web-програмування з поєднанням теорії та практичних прикладів усіх базових технологій, а також можливістю самоперевірки знань користувачем шляхом тестування, є досить актуальною задачею.

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

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є підвищення ефективності вивчення користувачем web-програмування шляхом об'єднання теоретичної інформації, практичних прикладів та тестування в межах одного інтернет-ресурсу, що дозволить забезпечити базові потреби користувача в рамках навчального процесу.

Відповідно до поставленої мети в бакалаврскій дипломній роботі потрібно вирішити такі **задачі**:

- розробити структуру створюваного web-додатку;
- розробити метод та модель роботи системи;
- налаштувати навігацію по web-додатку;
- створити блоки з теоретичним матеріалом;
- створити блоки з практичними прикладами для вивчення;
- розробити блоки «quiz» для тестування користувача;
- розробити блок для реєстрації користувача;
- розробити програмні компоненти web- додатку;
- провести тестування розробленої web-системи.

Об'єкт дослідження – процеси розробки програмних засобів для навчальних web-систем, призначених для вивчення web-програмування**.**

Предмет дослідження – методи та засоби розробки програмних засобів навчальної web-системи для вивчення web-програмування.

Методи дослідження. У процесі дослідження використовувались:

- методи комплексного аналізу для дослідження переваг і недоліків існуючих програмних реалізацій анлогів та формування вимог до розроблюваної системи;
- методи аналізу та синтезу для побудови ключових моделей автоматизованої системи;
- методи моделювання роботи web-системи та її окремих блоків у вигляді блок-схем для формалізації алгоритмів роботи системи;
- методи побудови компонентів системи та їхніх взаємозв'язків для створення загальної структури збереження та обробки інформаційних ресурсів;
- методи розробки Web-ресурсів для створення навчальної Web-системи;
- методи тестування для підтвердження працездатночті системи та її відповідності заданим вимогам.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод створення навчальної web-системи, який, на відміну від існуючих, реалізує систему вбудованих логічних зв'язків між компонентами для забезпечення повної підтримки навчального процесу з поєднанням блоків теоретичного матеріалу, практичних завдань і тестів з ідентифікацією рівня засвоєння матеріалу, що підвищує ефективність процесу вивчення основ web-програмування користувачем.

2. Подальшого розвитку отримала модель навчальної web-системи, яка, на відміну від існуючих, зосереджена на оптимізації навчального процесу шляхом реалізації комплексного функціоналу для вивчення основ web-програмування, що забезпечить реалізацію універсального підходу до розробки доступного навчального веб-ресурсу, спростить пошук потрібної інформації, забезпечить тренувальні потреби користувача й підвищить ефективність процесу навчання.

Розроблено алгоритм тестування виключно засобами програмування на стороні клієнта, що дозволило підвищити швидкість роботи навчальної web-системи.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає у створенні навчальної web-системи для вивчення основ web-програмування, що дозволить опановувати популярний IT-напрямок непідготовленим користувачам.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У наукових працях, опублікованих у співавторстві [2], автору належать такі результати: розробка навчальної веб-системи «StudyWEB»; розробка моделі та алгоритмів роботи системи, розробка блоку «quiz» для веб-системи.

Апробація результатів роботи. Результати роботи доповідалися на Міжнародній науково-практичній інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022", Секція - Інформаційні технології та комп’ютерна інженерія.

Публікації. Основні результати досліджень опубліковано в науковій праці [2] у збірнику матеріалів Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022".

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та використано 23 літературних джерела.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану навчальних web-систем для вивчення web-програмування

Для того, аби стати справді професійним web-розробником, необхідно опанувати велику кількість технологій, проте починати необхідно з основ.

HTML CSS та JavaScript – це фундаментальна основа web-програмування. Детальне ознайомлення з даними технологіями важливе для старту в web-розробці.

Самостійне онлайн навчання набуло широкої популярності у наш час, особливо у сфері IT, тому існує безліч курсів, додатків та відеороликів для вивчення web-програмування.

Однак курси та додатки здебільшого не є безкоштовними, що може відштовхнути потенційних новачків, а навчальні відео не систематизовані, що ускладнює процес освоєння нової спеціальності.

Станом на сьогоднішній день, коли доступ до мережі інтернет не є проблемою, завантажувані додатки не користуються популярністю серед користувачів, тобто, основною перевагою сайтів, у порівнянні із локальними додатками, є те, що користувачу не потрібно встановалювати їх на свій пристрій, достатньо мати доступ до мережі інтернет для отримання інформації.

Тому питання програмної реалізації навчальної web-системи залишається відкритим.

Web-система – сукупність web-сторінок під єдиним доменним ім’ям у мережі інтернет, об’єднаних між собою навігацією. За кожною web-системою закріплена адреса, що визначає її в мережі.

Для перегляду web-систем користувачу необхідна спеціальна програма – браузер. Фізично система розміщується на сервері, на одному або на кількох [3]. Приклад інтерфейсу навчальної web-системи наведено на рисунку 1.1.

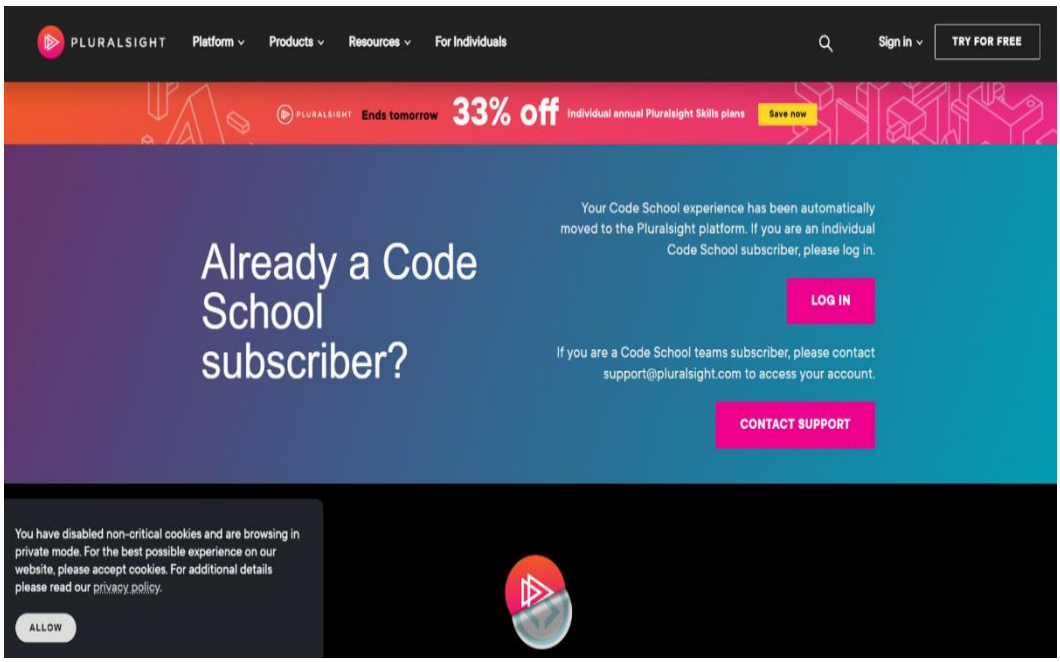

Рисунок 1.1 – Приклад інтерфейсу web-системи для вивчення програмування

Отже, самостійне онлайн навчання та навчальні сайти, що надають необхідний матеріал набувають у наш час все більшої популярності, особливо, якщо це стосується сфери IT, адже дозволяють користувачам опанувати нову спеціальність або необхідний навик.

Для доступу до web-систем необхідне лише підключення до мережі інтернет на будь-якому пристрої, що значно розширює список потенційних користувачів.

1.2 Порівняльний аналіз аналогів

Для аналізу сучасного стану питання теми дипломної роботи, було обрано кілька реалізацій web-систем, які надають користувачу можливість вивчати web-програмування.

Розглянемо деякі з них:

- W3schools;
- TeamtreeHouse;
- Codecademy;
- Css-tricks.

W3schools – web-система для вивчення програмування для початківців. Дана система надає користувачам матеріали для вивчення багатьох мов програмування, таких як JavaScript, Python чи PHP.

Основним недоліком є те, що повний обсяг матеріалу доступний лише при платній підписці «Pro». Також попри широкий вибір мов програмування для вивчення, система надає поверхневі курси у порівнянні з іншими аналогами.

Інтерфейс web-системи W3schools наведено на рисунку 1.2 [4].

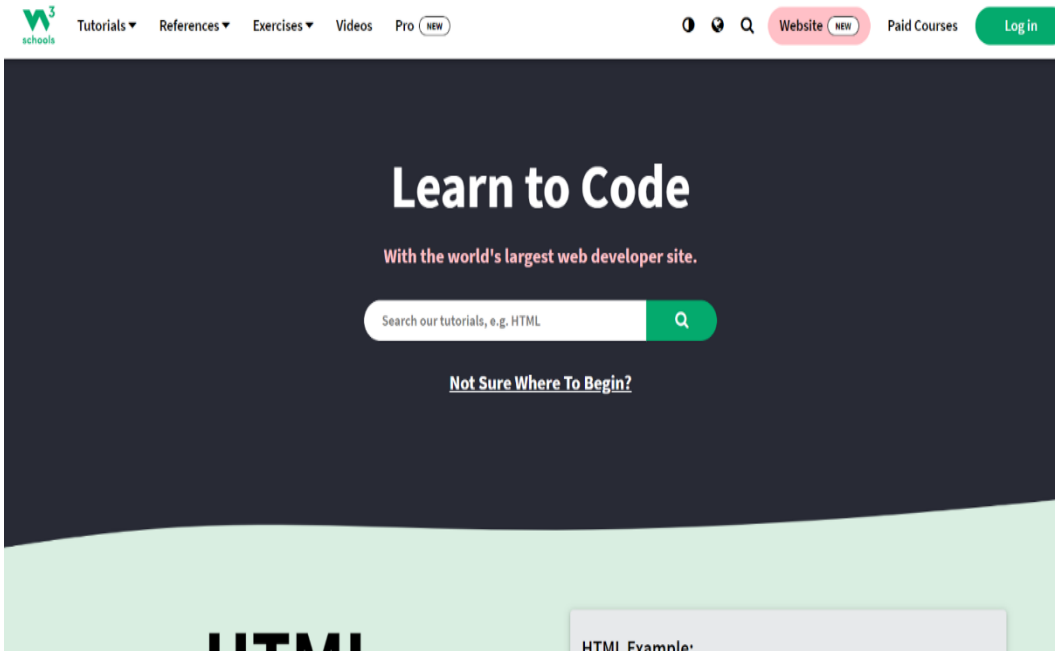

Рисунок 1.2 – Інтерфейс web-системи W3schools

TeamtreeHouse – web-система для вивчення web-програмування, що надає користувачу доступ до онлайн курсів для вивчення мови розмітки гіпертексту HTML, а також для вивчення каскадних таблиць стилів CSS.

Недоліками є відсутність матеріалів для вивчення мови програмування JavaScript, а також те, що безкоштовний доступ до повного обсягу матеріалу доступний лише на пробний період – 7 днів, далі за умови оформлення платної підписки.

Інтерфейс web-системи TeamtreeHouse наведено на рисунку 1.3 [5].

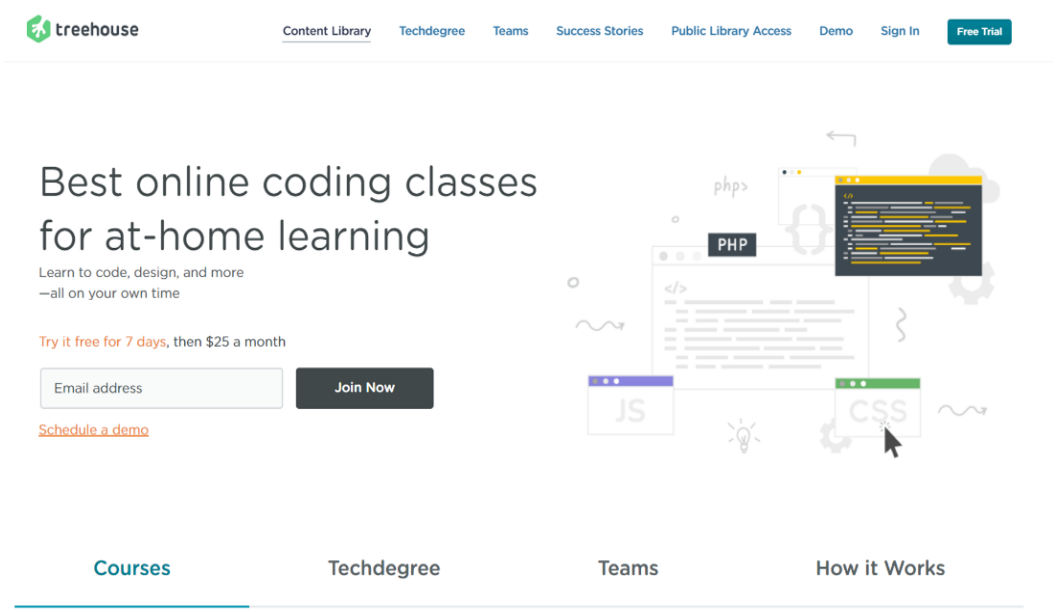

Рисунок 1.3 – Інтерфейс web-системи TeamtreeHouse

Codecademy – web-система для вивчення мови розмітки гіпертексту HTML. Цей web-сайт надає користувачу усі необхідні матеріали для вивчення мови HTML, а також дозволяє отримати сертифікат по завершенню курсу.

Недоліком є відсутність матеріалів для вивчення CSS та JavaScript. Інтерфейс web-системи Codecademy наведено на рисунку 1.4 [6].

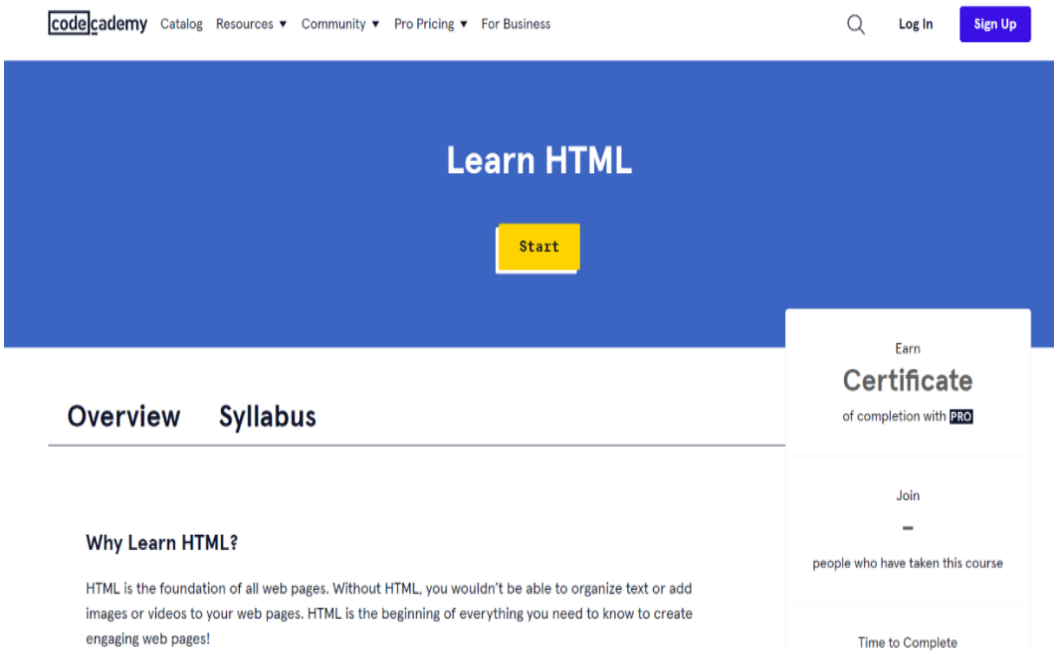

Рисунок 1.4 – Інтерфейс web-системи Codecademy

Css-tricks – web-система з практичними прикладами застосування каскадних таблиць стилю CSS. Користувачі сайту можуть ознайомитися з цікавими прикладами роботи досвідчених web-розробників.

Недоліком є відсутність матеріалів для вивчення мови програмування JavaScript.

Інтерфейс web-системи Css-tricks наведено на рисунку 1.5 [7].

Рисунок 1.4 – Інтерфейс web-системи Css-tricks

Після аналізу усіх аналогів визначено їхні переваги та недоліки та проведено порівняння із розроблюваною web-системою «StudyWEB». Результат порівняння зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльні характеристики навчальних web-систем

Критерій	W3schools	Teamtree House	Codec ademy	Css-tricks	StudyWEB
Наявність матеріалу для вивчення усіх базових технологій web-програмування	1	0	0	0	1
Тестування для самоперевірки	0	1	1	0	1

Продовження таблиці 1.1

Безкоштовність повного функціоналу	0	0	1	1	1
Адаптивність	1	1	1	1	1
Підсумковий результат	2	2	3	2	4

Розглянувши дані, наведені в таблиці 1.1, можна зробити висновок, що створення власної навчальної web-системи «StudyWEB» є актуальним. В результаті розробки отримаємо web-додаток, що враховує недоліки аналогів та надає користувачу необхідні матеріали для вивчення web-розробки.

1.3 Аналіз методів розв'язання задачі

Для реалізації поставленої задачі необхідно створити web-систему, що буде поєднувати в собі вивчення таких технологій як HTML, CSS та JavaScript. Також вона повинна надавати користувачу як теоретичний матеріал, так і практичні приклади, окрім цього необхідно реалізувати систему тестування, для перевірки засвоєного матеріалу.

Для цього web-систему розділено на сторінки:

- для вивчення загальної інформації про web-розробку;
- для вивчення мови розмітки гіпертексту HTML;
- для вивчення каскадних таблиць стилів CSS;
- для вивчення мови програмування JavaScript.

Для подачі матеріалу було прийнято рішення про розділення сторінок на блоки з інформацією, використовуючи методологію БЕМ, що також спростить задачу реалізації самої web-системи.

БЕМ (Блок, Елемент, Модифікатор) – компонентний підхід до web-розробки. Дана методологія заснована на принципі поділу інтерфейсу на незалежні блоки, що дозволяє повторно використовувати уже написаний код, там

де це необхідно. Такий підхід у web-програмуванні дозволяє швидко створювати сайти з гнучкою архітектурою.

Блок – функціонально незалежний компонент, що можна використати повторно. Атрибут class визначає блок та його назву. Назва блоку повинна описувати суть об’єкту.

Елемент – складова частина блоку, що не може використовуватись окремо. Назва елемента повинна описувати його суть. Синтаксис назви елемента включає в себе назву блоку, котрому належить елемент: ім’я-блоку__ім’я-елементу.

Модифікатор застосовується для визначення чи уточнення зовнішнього вигляду, стану чи поведінки блоку або елемента, якщо потрібно виділити певний компонент з числа ідентичних. Тому назва модифікатора повинна характеризувати стан або поведінку об’єкту [8].

Використовуючи компонентний підхід, створено блоки з теоретичним матеріалом, блоки з практичними прикладами та блоки з тестуванням, для надання користувачу можливості самоперевірки. Блоки реалізовано за допомогою технологій HTML5, SCC3 препроцесора SASS/SCSS та JS.

HTML (HyperText Markup Language) – стандартизована мова розмітки документів для перегляду web-сторінок у браузері. Мова HTML дозволяє визначити структуру електронного документа. Основою роботи з HTML є прості структури коду: теги та атрибути, для розмітки сторінок web-системи. Результуючий документ може містити різноманітні елементи: зображення, аудіо та відео фрагменти [9].

CSS (Cascading Style Sheets) – це мова візуального представлення вмісту сторінок, написаних HTML, XHTML та інших видів XML-документів. Це одна з базових технологій web -розробки.

Таблиці стилів дають змогу спростити процес створення сторінок а також поліпшити їхній зовнішній вигляд [10].

SASS/SCSS (англ. Syntactically Awesome Stylesheets) – це метамова створена на основі каскадних таблиць стилів CSS, для підвищення рівня абстракції коду.

SASS розширює CSS, надаючи нові можливості, такі як вкладеність класів, змінні, шаблони, міксини а також математичні оператори. Інтерпретатор SASS транслює SassScript у блоки правил CSS [11].

JavaScript – це об’єктно-орієнтована прототипна мова програмування. Вона являє собою реалізацію стандарту ECMAScript.

Найчастіше ця мова використовується для створення скриптів web-сторінок. Дана мова програмування дозволяє зробити web-сторінку інтерактивною, для взаємодії з користувачем [12].

1.4 Постановка задач розробки навчальної web-системи для вивчення web-програмування

Після проведення аналізу питання створення власної навчальної web-системи для вивчення web-розробки «StudyWEB», було виявлено недоліки існуючих реалізвцій та визначено їх переваги.

Опираючись на аналіз, визначили головні задачі для реалізації у бакалаврській дипломній роботі:

- розробити структуру створюваного web-додатку;
- розробити метод та модель роботи системи;
- налаштувати навігацію по web-додатку;
- створити блоки з теоретичним матеріалом;
- створити блоки з практичними прикладами для вивчення;
- розробити блоки «quiz» для тестування користувача;
- розробити блок для реєстрації користувача;
- розробити програмні компоненти web- додатку;
- провести тестування розробленої web-системи.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто стан питання існуючих реалізацій web-систем для вивчення web-програмування. Було проведено аналіз існуючих аналогів W3schools, TeamtreeHouse, Codecademy та Css-tricks, виявлено недоліки та переваги даних реалізацій. Шляхом порівняння створюваного web-додатку з аналогами було доведено доцільність розробки бакалаврської дипломної роботи. Також було проведено аналіз існуючих підходів до вирішення поставленої задачі та сформульовано основні завдання, які необхідно виконати для розробки навчальної web-системи.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ WEB-СИСТЕМИ

2.1 Аналіз навчального функціоналу web-системи

Основною задачею бакалаврської дипломній роботи є створення навчальної web-системи, що допоможе користувачам у вивченні основ web-програмування. Для виконання поставленої задачі необхідно приділити увагу саме навчальному аспекту системи, аби процес вивчення був максимально ефективним.

Для ефективного вивчення програмування новачкам рекомендується дотримуватися методу поділу матеріалу на частини. Не варто братися за вивчення всього і одразу, це не лише не ефективно, але й може відбити у новачка бажання навчатися напливом великого обсягу нового матеріалу.

Найкраще ділити матеріал для вивчення по темам, починаючи з основ і переходити до більш складних завдань. Важливо поєднувати теорію і практику – таке навчання більш ефективне. Розбір готових практичних прикладів є надзвичайно важливим аспектом вивчення програмування, так можна отримати новий досвід та знання, навчитися тонкощам, а також поліпшити загальне розуміння теми [13].

Ще одним ключовим аспектом навчання загалом, та процесу освоєння web-програмування зокрема, є перевірка засвоєного матеріалу та якості знань новачка. Для цього необхідно реалізувати метод тестування користувача. Надання відповідей на короткі запитання по заданій темі у тестовому форматі є найбільш ефективною формою перевірки засвоєного матеріалу та рівня знань користувача.

Увесь навчальний матеріал подано у вигляді web-системи, тому для покращення рівня ефективності процесу вивчення необхідно розробити зручний інтерфейс користувача, що допоможе йому орієнтуватися у створюваній системі.

Ще одним ключовим моментом є доступність навчальної web-системи. Для того щоб користувач міг отримати доступ до навчальних матеріалів з будь-якого пристрою, було прийнято рішення зробити створювану систему адаптивною.

Адаптивна верстка – це верстка, при якій сайт може відображатися на усіх можливих пристроях: комп’ютерах, планшетах та телефонах. Адаптивні web-системи стають дедалі більш популярними, адже частка мобільного трафіку від загального постійно зростає [14].

2.2 Розробка структури інтерфейсу web-системи

Інтерфейс створюваної web-системи повинен бути зрозумілим для нового користувача та простим у використанні.

При входжені до системи, на головній сторінці користувач може зареєструватися в системі за допомогою блоку реєстрації, що містить відповідну форму (рис. 2.1).

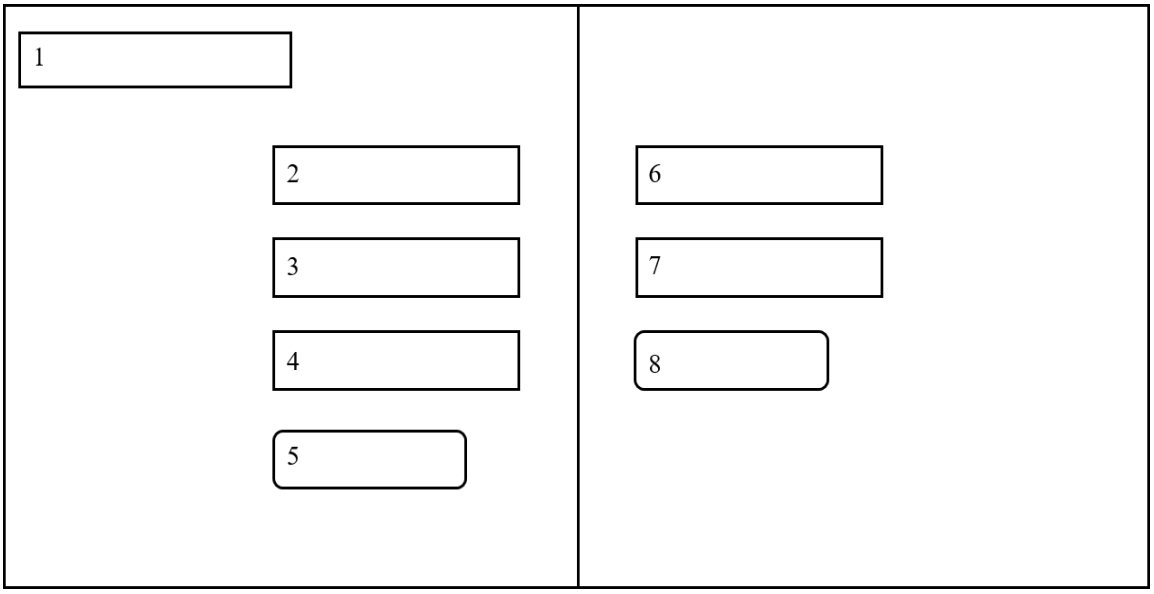

Рисунок 2.1 – Графічна схема блоку реєстрації

Елементи блоку реєстрації:

1. Заголовок блоку;
2. Поле «логін» для реєстрації;
3. Поле «ім’я» для реєстрації;
4. Поле «пароль» для реєстрації;
5. Кнопка для реєстрації;

6. Поле «логін» для авторизації;
7. Поле «пароль» для авторизації;
8. Кнопка для авторизації.

Для навігації по web-системі додано навігаційну панель (рис. 2.2), що містить посилання в межах конкретної сторінки, випадаюче меню (рис. 2.3) з посиланням на інші сторінки web-системи, а також назву web-додатку, що є посиланням на головну сторінку.

Рисунок 2.2 – Навігаційна панель

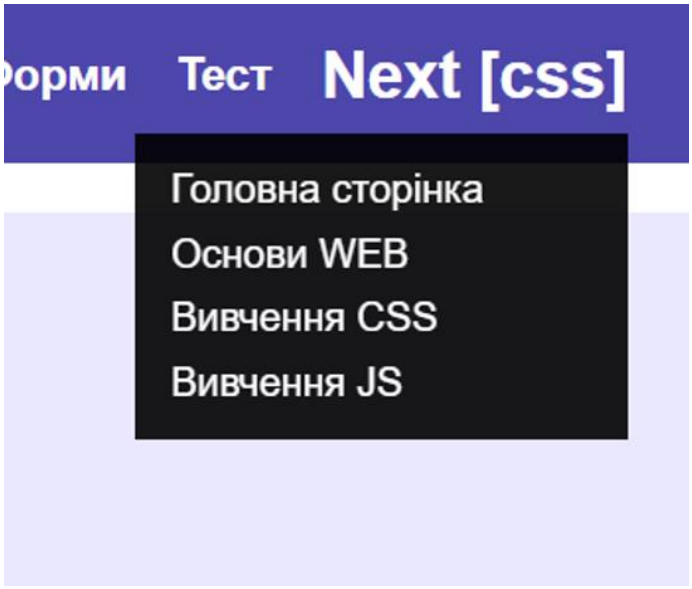

Рисунок 2.3 – Випадаюче меню

Для зручності навігації по навчальних матеріалах на відповідних сторінках додано зміст матеріалу (рис. 2.4).

Навчальний матеріал розділено на блоки (рис. 2.5) для зручності користувача.

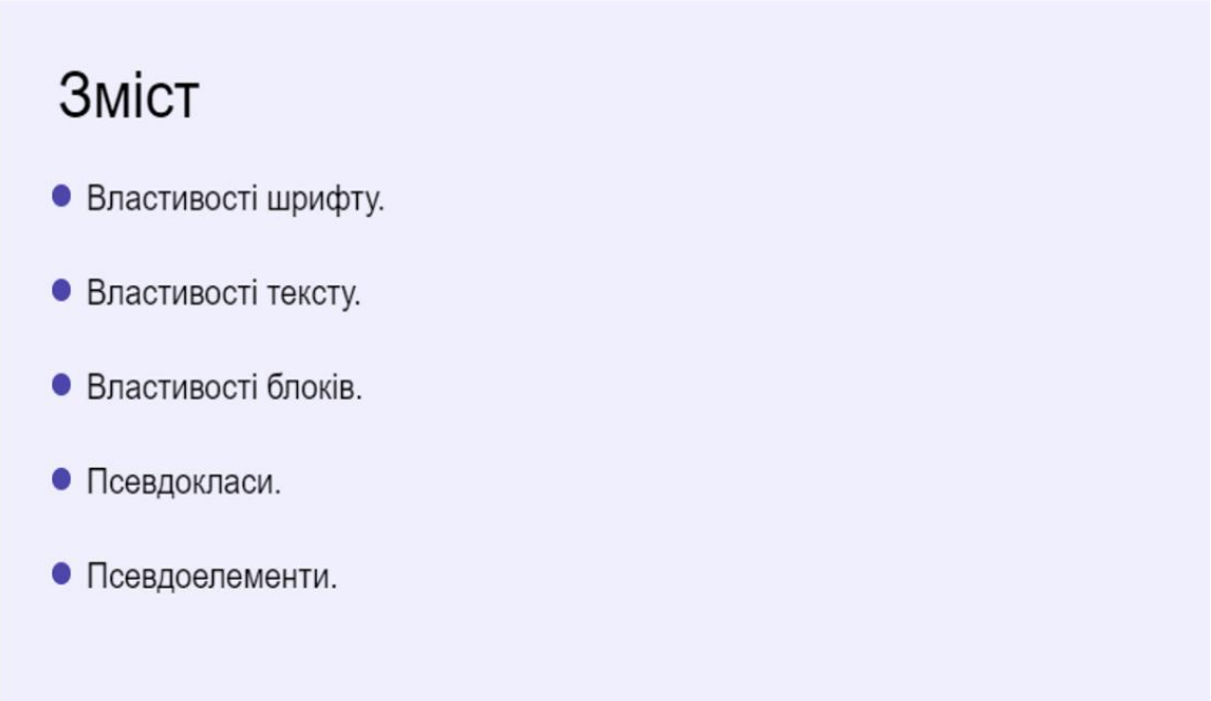

Рисунок 2.4 – Приклад змісту матеріалу

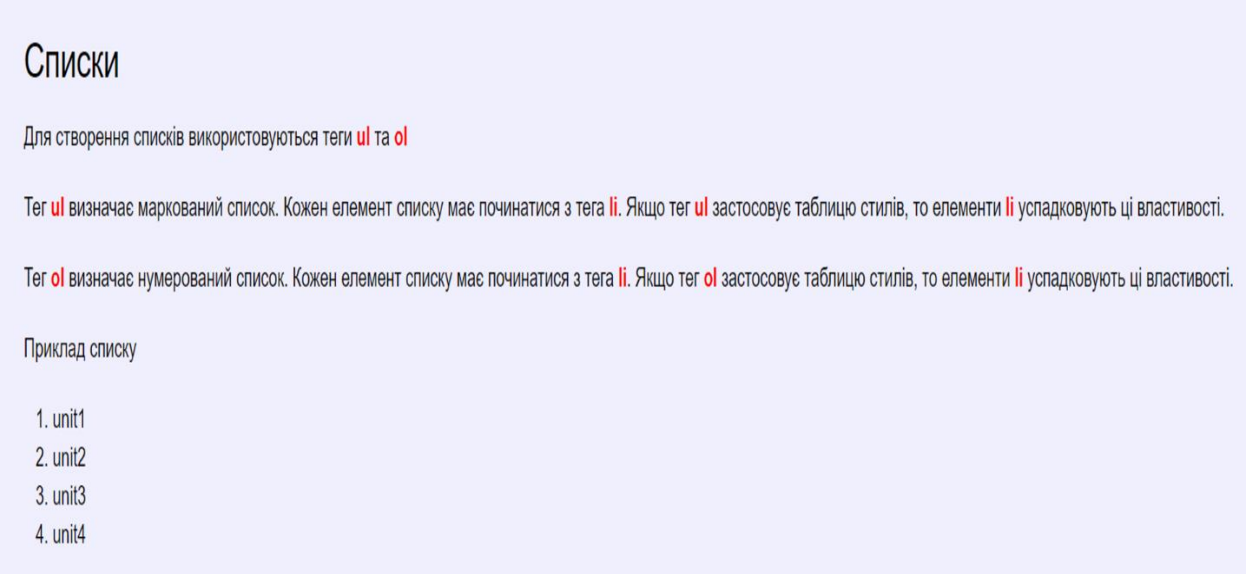

Рисунок 2.5 – Приклад блоку з інформацією

Блоки для тестування (рис. 2.6) містять перемикачі для вибору правильного, на думку користувача, варіанту відповіді, кнопку «Далі» для переходу до наступного питання та кнопку «Заново», для повторного проходження тесту, дана кнопка з’являється лише по завершенню тестування (рис. 2.7).

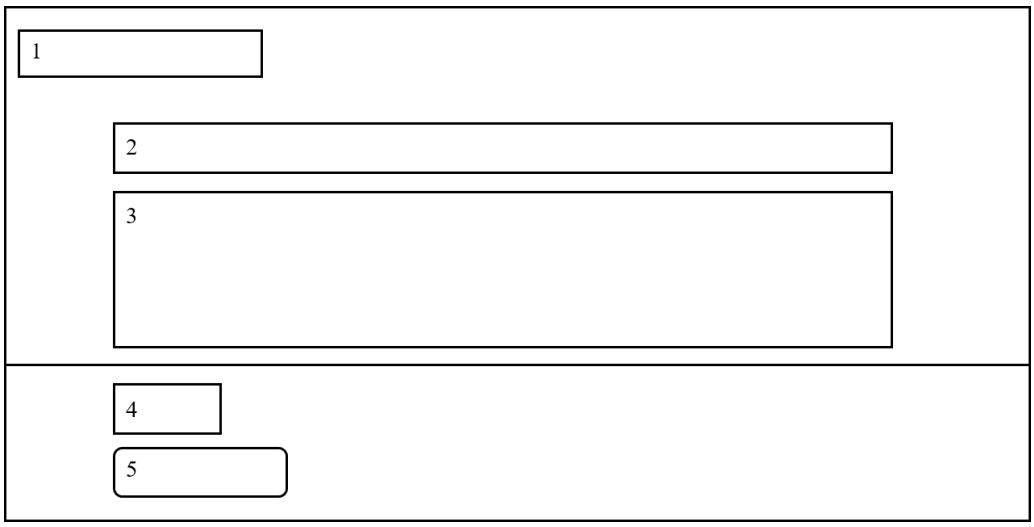

Рисунок 2.6 – Графічна схема блоку тестування

Елементи блоку тестування:

1. Заголовок блоку;
2. Питання;
3. Варіанти відповідей з перемикачами;
4. Індикатор номеру аитання;
5. Кнопка для переходу до наступного питання;

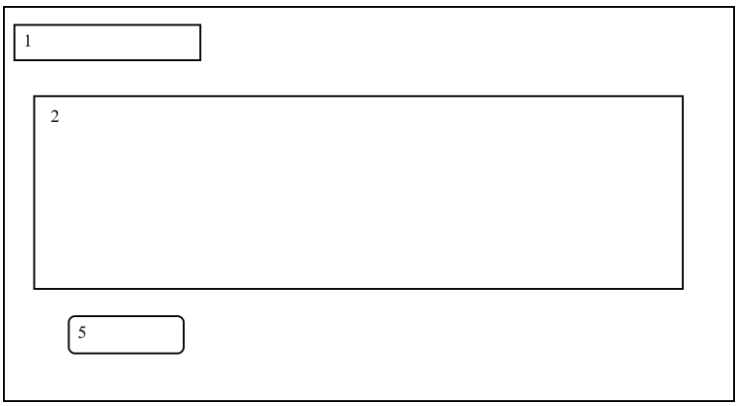

Рисунок 2.7 – Графічна схема блоку тестування при виведенні результатів

Елементи блоку тестування:

1. Заголовок блоку;
2. Питання та відповіді;
3. Кнопка для завершення тестування.

Розроблювана web-система повинна бути адаптивною, тому для коректної роботи на пристроях з тачскріном необхідно внести зміни до інтерфейсу. До основного посилання додано елемент «стрілка» для виклику випадаючого меню, що дозволить зберегти властивості посилання.

Для пристроїв з шириною екрану менше ніж 767 пікселів створено меню бургер (рис. 2.8), для коректного відображення елементів навігації на мобільних пристроях.

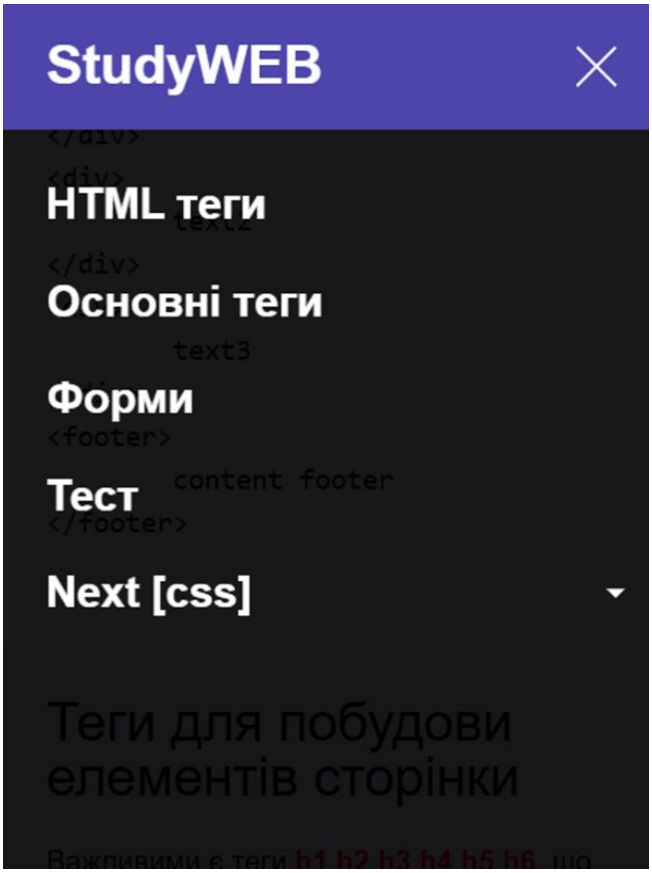

Рисунок 2.8 – Меню «Бургер»

Розробку графічного інтерфейсу web-системи проведено у середовищі розробки Sublime Text.

2.3 Розробка методу тестування для користувача

Для того, щоб надати користувачам навчальної web-системи можливість для перевірки своїх знань з конкретного аспекту web-програмування, було вирішено

розробити метод тестування для користувача (рис. 2.9). Даний метод передбачає створення блоків з тестуванням (рис. 2.10), що можуть містити задану кількість запитань та варіантів відповідей для кожного питання. Відповівши на усі питання користувач отримає свої результати. Також після проходження тесту користувач зможе пройти його повторно, будь-яку кількість разів, натиснувши кнопку «Заново». Питання та відповіді для тестування зберігаються у масиві в самому .js файлі, тобто повністю розміщені на клієнтській частині web-системи, не потребуючи серверної частини.

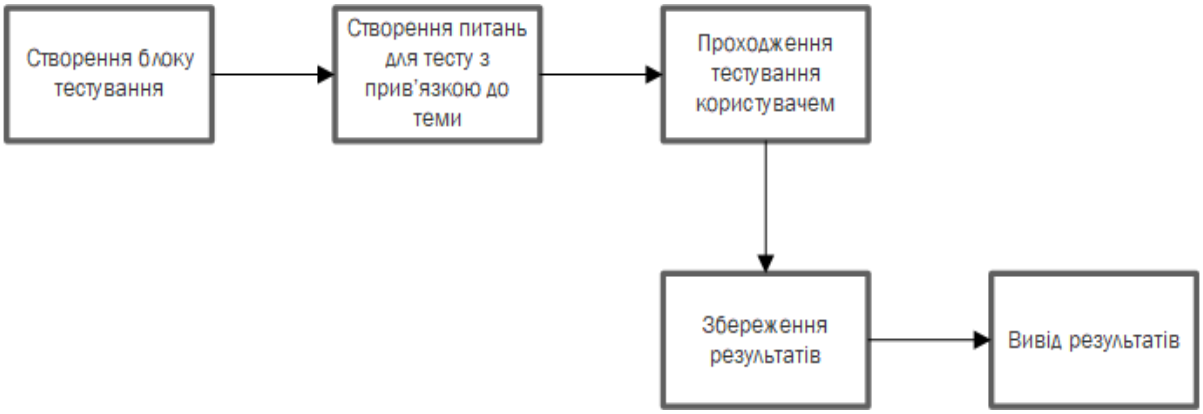

Рисунок 2.9 – Схема методу тестування для користувача

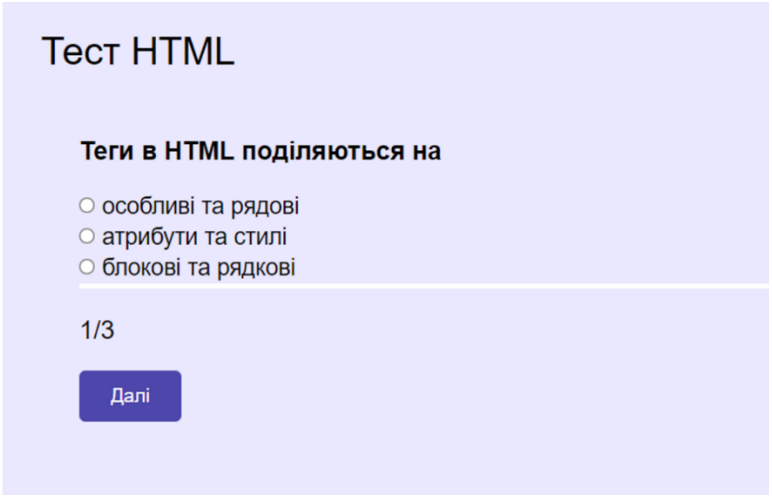

Рисунок 2.10 – Приклад блоку тестування

Розроблений метод тестування дозволяє користувачу перевірити свій рівень знань під час взаємодії з системою, та допомагає у виявленні прогалин в знаннях

користувача, для їх подальшого усунення. Метод тестування дозволяє підвищити ефективність навчання, та оптимізувати процес вивчення користувачем web-розробки.

Метод тестування користувача використовує заздалегідь створені масиви, для питань та відповідних їм варіантів відповідей, а також для збереження результатів поточного тестування. Використання масивів дозволяє проводити тестування користувача на клієнтській частині web-системи, без запитів до баз даних.

Також, після виводу результатів тестування на екран, масив поточних відповідей користувача обнуляється. Усе це дозволяє підвищити ефективність роботи системи.

Метод тестування містить таку послідовність операцій:

1. Авторизувавшись у системі, користувач отримує доступ до навчальних матеріалів, що включають в себе блоки тестування.

2. На початку процесу тестування блок уже містить перше питання та відповідні йому варіанти відповідей.

2.1. При цьому елемент кнопка «Далі» є недоступною для користувача, поки не буде обрано будь-який варіант відповіді з наявних.

2.2. При переході до кожного наступного питання індикатор номеру питання інкрементується.

3. У процесі тестування користувача надані ним відповіді зберігаються в масиві поточних результатів.

3.1. Надані користувачем відповіді зберігаються для їх подальшого виведення на екран.

3.2. По завершенню процесу тестування масив поточних результатів обнуляється.

4. Вивід результатів тестування на екран користувача.

2.4 Розробка моделі роботи web-системи

Модель роботи навчальної системи для вивчення web-програмування подано у вигляді UML діаграми діяльності (рис. 2.11).

UML діаграма діяльності – це візуальне представлення графу діяльності, що є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню цих дій [15].

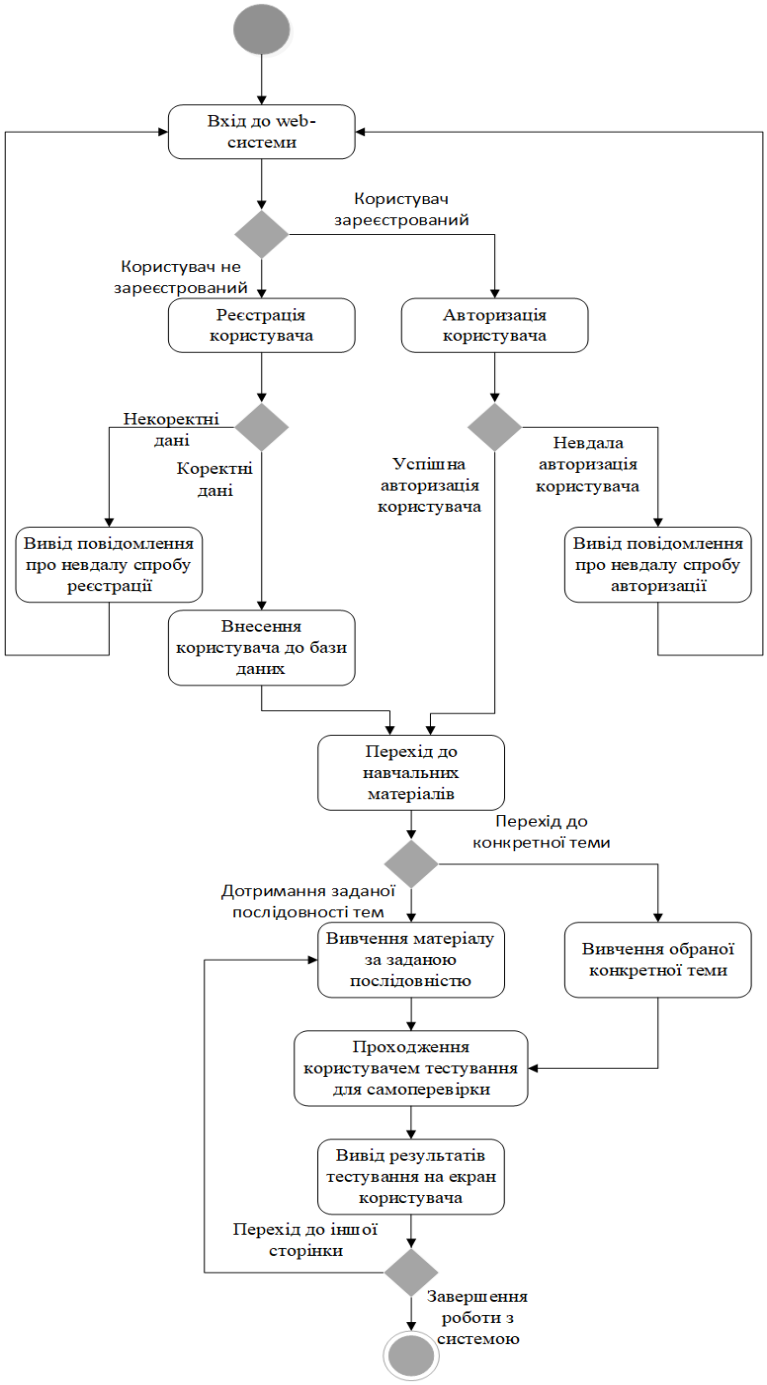

Рисунок 2.11 – Модель роботи навчальної web-системи

Як показано у моделі, при вході до web-системи користувачу потрібно зареєструватися або ж авторизуватися, якщо він уже має власний акаунт. При реєстрації нового акаунта дані користувача вносяться до бази даних.

При некоректному вводі даних для реєстрації система виводить спеціальне повідомлення з вказанням на недопустиме значення. Аналогічно, при невдалій авторизації система виводить відповідне повідомлення.

Після успішної реєстрації/авторизації користувач переходить до навчальних матеріалів, де також має можливість перевірити свої знання за допомогою тестування.

2.5 Розробка методу та алгоритмів роботи web-системи

Для досягнення мети бакалаврської дипломної роботи необхідно розробити загальний метод роботи web-системи.

Даний метод дозволить підвищити ефективність вивчення користувачем web-програмування, за рахунок об’єднання теоретичної інформації, практичних прикладів та тестування в межах однієї web-системи. Поділ навчальних матеріалів на блоки, та створення відповідної навігації дозволить забезпечити базові потреби користувача, у процесі освоєння основ web-розробки.

Метод роботи web-системи містить наступну послідовність операцій:

1. Авторизувавшись у системі, користувач отримує доступ до навчальних матеріалів. Блок реєстрації реалізовано засобами мови програмування PHP.

2. Після авторизації користувач переходить на сторінку Основи WEB (basic.html).

2.1. Ознайомившись із матеріалами заданої web-сторінки, користувач може перейти до наступної сторінки, користуючись посиланням на панелі навігації.

2.2. При переході за посиланням користувач буде поступово вивчати наданий матеріал за визначеним планом, однак, при наведені курсору на дане

посилання відкривається випадаюче меню, у якому користувач може самостійно обрати необхідний йому матеріал.

3. Інформацію на кожній web-сторінці розділено по блоках за тематикою.

3.1. Користувач буде поступово вичати наданий матеріал за рекомендованим планом, переходячи від одного блоку до іншого. Блоки теоретичного матеріалу чергуються з прикладами практичного використання для підвищення ефективності навчального процесу. Приклади практичного використання реалізовано засобами Css та мови програмування JavaScript.

3.2. Кожна web-сторінка містить внутрішні посилання на блоки з найбільш важливою інформацією, тому користвач може самостійно обрати тему для вивчення. Перехід за внутрішніми посиланнями реалізовано засобами мови програмування JavaScript, за допомогою методу .scrollTo() об’єкту window.

4. По завершенню процесу вивчення користувач може пройти тестування для самоперевірки у призначених для цього відповідних блоках. Блоки тестування реалізовано засобами мови програмування JavaScript та Css, для стилізації результатів.

Для створення навчальної web-системи для вивчення web-програмування необхідно розробити загальний алгоритм, відповідно до якого буде працювати система (рис. 2.12). За цим алгоритмом при входжені до системи користувач перш за все повинен зареєструватися, або ж авторизуватися, за наявності створеного раніше акауну. При реєстрації дані про нового користувача будуть внесені до бази даних.

Після успішного входу йому надається доступ до навчальних матеріалів web-системи. Матеріали у системі розподілені на окремі сторінки по технологіям а також розділені на блоки по темам і систематизовані. Користувач може проходити увесь курс від початку до кінця, або ж розглянути вибрану тему в

меню навігації. Для перевірки своїх знань по обраній темі користувач має можливість пройти тестування та одразу переглянути свої результати.

Для детального розбору роботи блоку тестування користувачів було розроблено відповідний алгоритм (рис. 2.13). Відповідно до даного алгоритму користувач розпочинає одразу з першого питання у блоці. Для розблокування кнопки «Далі» та переходу до наступного питання користувачу необхідно вказати відповідь на поточне питання. Надана відповідь зберігається на час проходження тестування для демонстрації результатів.

При переході до наступного питання відбувається рендеринг запитання та його варіантів відповідей. Разом з цим інкрементується індикатор питань, якщо дане питання є заключним, то після надання відповіді на нього користувач може переглянути результати свого тестування. Також для користувача відображається кнопка «Заново», що дозволяє пройти тестування повторно.

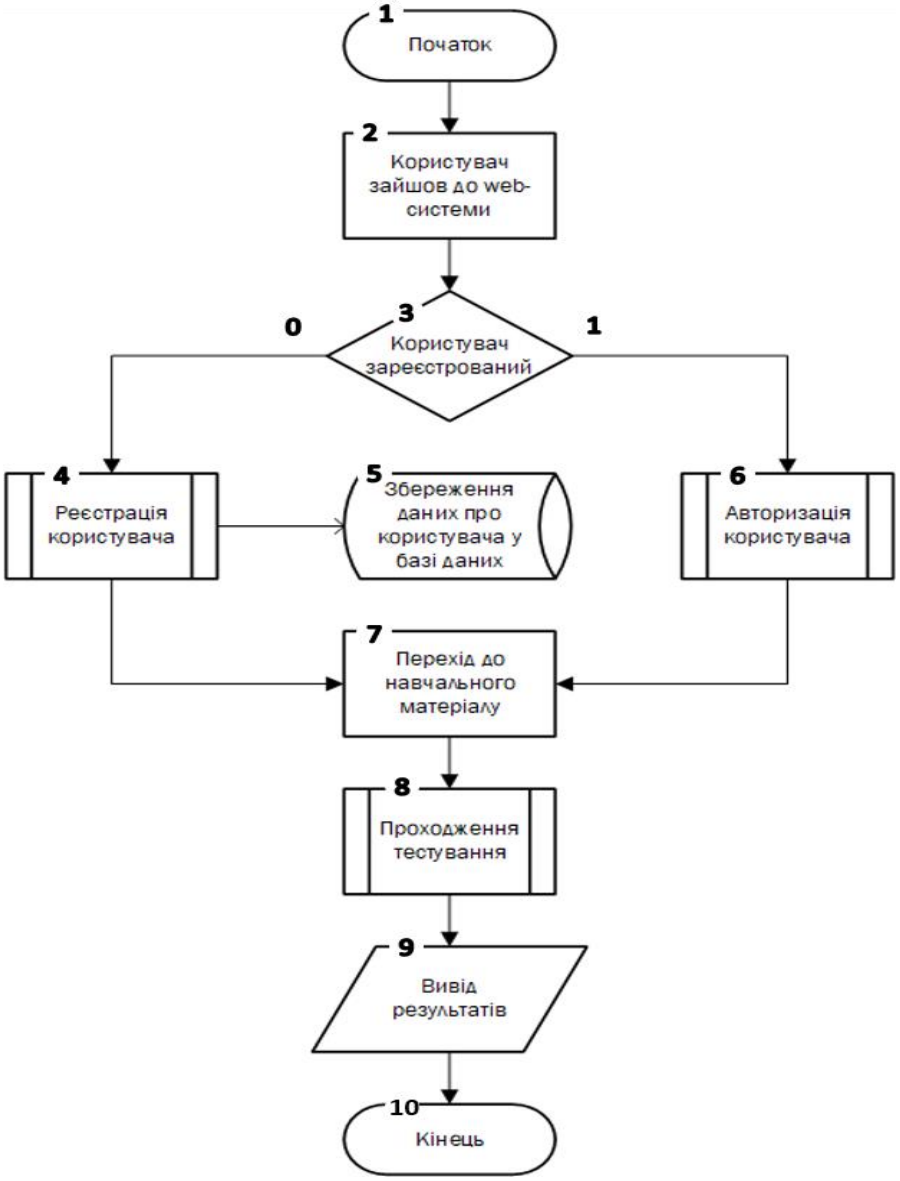

Рисунок 2.12 – Блок-схема загального алгоритму роботи web-системи

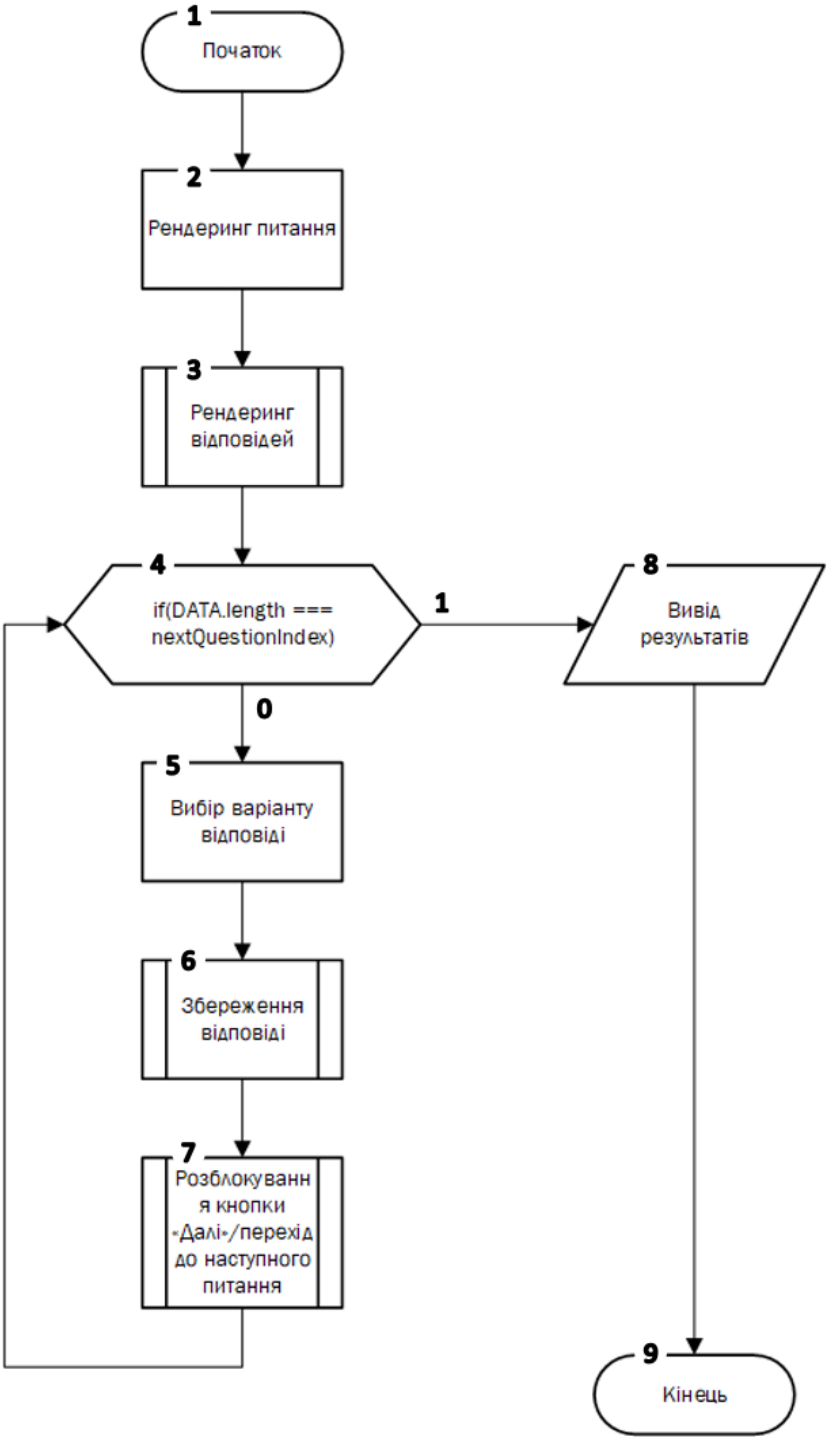

Рисунок 2.13 – Блок-схема алгоритму тестування користувача

Надані користувачем відповіді зберігаються в масиві поточних результатів. Це необхідно для подальшого виведення результатів тестування на екран користувача. По завершенню тестування масив обнуляється, для можливості наступного проходження тесту, що можливо реалізувати засобами мови програмування JavaScript.

2.6 Висновки

У другому розділі було проведено аналіз навчального аспекту розроблюваної web-системи. Було розроблено структуру інтерфейсу web-системи. Також було розроблено метод тестування, що допоможе користувачам у вивченні web-розробки, та модель роботи web-системи і алгоритм проведення тестування для перевірки знань користувача. Для наглядності, роботу web-системи було подано у вигляді UML діаграми діяльності.

3 РОЗРОБКА СИСТЕМИ ДЛЯ ВИВЧЕННЯ WEB-ПРОГРВМУВАННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для програмної реалізації web-системи

Для створення web-системи перш за все необхідно визначитися із технологіями для її реалізації. Основа системи побудована засобами мови розмітки гіпертексту HTML, каскадних таблиць стилів CSS та мови програмування JavaScript для реалізації функціоналу.

Однак створення даної web-системи потребує роботи не лише з клієнтською частиною, але і з серверною, для запису даних користувача при реєстрації. Для цього необхідно скористатися засобами мови програмування на серверній стороні. Станом на сьогодні існує широкий вибір мов програмування для створення скриптів на стороні сервера, найбільш використовуваними серед них є: Ruby, Python, та PHP.

Ruby – це мова програмування, що поєднує в собі Perl-подібний синтаксис, та об’єктно-орієнтований підхід мови програмування Smalltalk. У мові програмування Ruby кожен тип даних являється об’єктом, а кожна функція є методом.

Python – це об’єктно-орієнтована мова програмування високого рівня. Дана мова програмування підтримує модульність, що дозволяє повторно використовувати написаний код.

PHP – це широко відома скриптова мова програмування із відкритим вихідним кодом. PHP спеціально сконструйована мова для web-розробки, щоб генерувати HTML-сторінки на стороні сервера, код цієї мови програмування може впроваджуватися безпосередньо в HTML [16].

Для реалізації web-системи було обрано мову програмування PHP, перевагою якої є належність до інтерпретованих мов, тобто обробка скриптів відбувається швидше ніж при використанні аналогів. Ще однією перевагою даної

мови програмування є те, що вона підтримується більшістю хостинг-провайдерів, а також більшістю систем керування базами даних.

Для збереження даних користувачів системи, при реєстрації, було створено SQL базу даних методами додатку phpMyAdmin.

PhpMyAdmin – це web-додаток, що надає графічний інтерфейс для адміністрування бази даних MySQL або MariaDB. Можливості даного додатку дозволяють здійснювати адміністрування сервера MySQL, запускати запити SQL, а також переглядати чи редагувати вміст таблиць баз даних, без необхідності встановлення додаткового програмного забезпечення [17].

3.2 Аналіз та обґрунтування вибору середовища для розробки

Важливим етапом у процесі створення web-системи є вибір найбільш підходящого середовища для розробки.

Для роботи над реалізацією web-системи було обрано середовище для розробки Sublime Text (рис. 3.1). Даний редактор дозволяє працювати з усіма необхідними для реалізації теми бакалаврської дипломної роботи мовами програмування та технологіями. Sublime Text має високу ступінь персоналізації, а також надає можливість завантажувати додаткові плагіни для розробки.

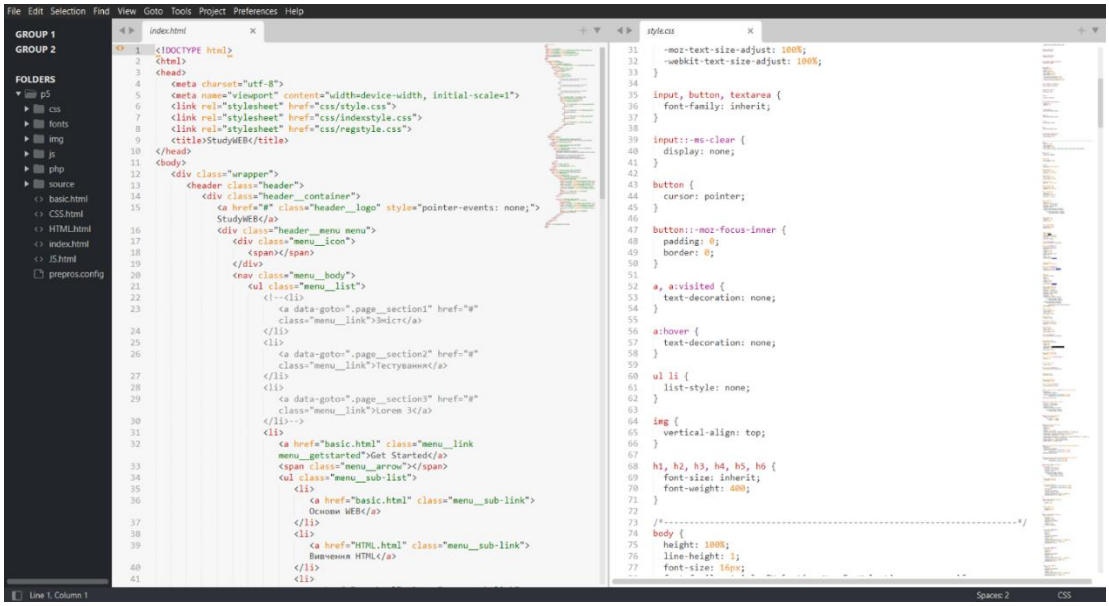

Рисунок 3.1 – Інтерфейс середовища для розробки Sublime Text

Редактор дозволяє застосовувати підсвічування синтаксису, відповідно до формату активного файлу, для зручності роботи. Sublime Text дозволяє розділяти робочий простір на вікна, для одночасної роботи з кількома файлами.

Також рдактор має вбудовану панель навігації по активному файлу та систему пошуку необхідних визначень символів чи рядків, що дозволяє швидко та легко орієнтуватися в написаному коді.

При створенні web-системи, для зручності та швидкості розробки було використано засоби CSS препроцесора SASS синтаксису SCSS, однак формат .scss не підтримується браузерами.

Для компіляції препроцесора, було вирішено скористатися програмою-збирачем Prepros (рис. 3.2). Вона дозволяє перетворювати формати препроцесорів у формат .css зрозумілий для браузерів, а також надає корисний функціонал для розробника.

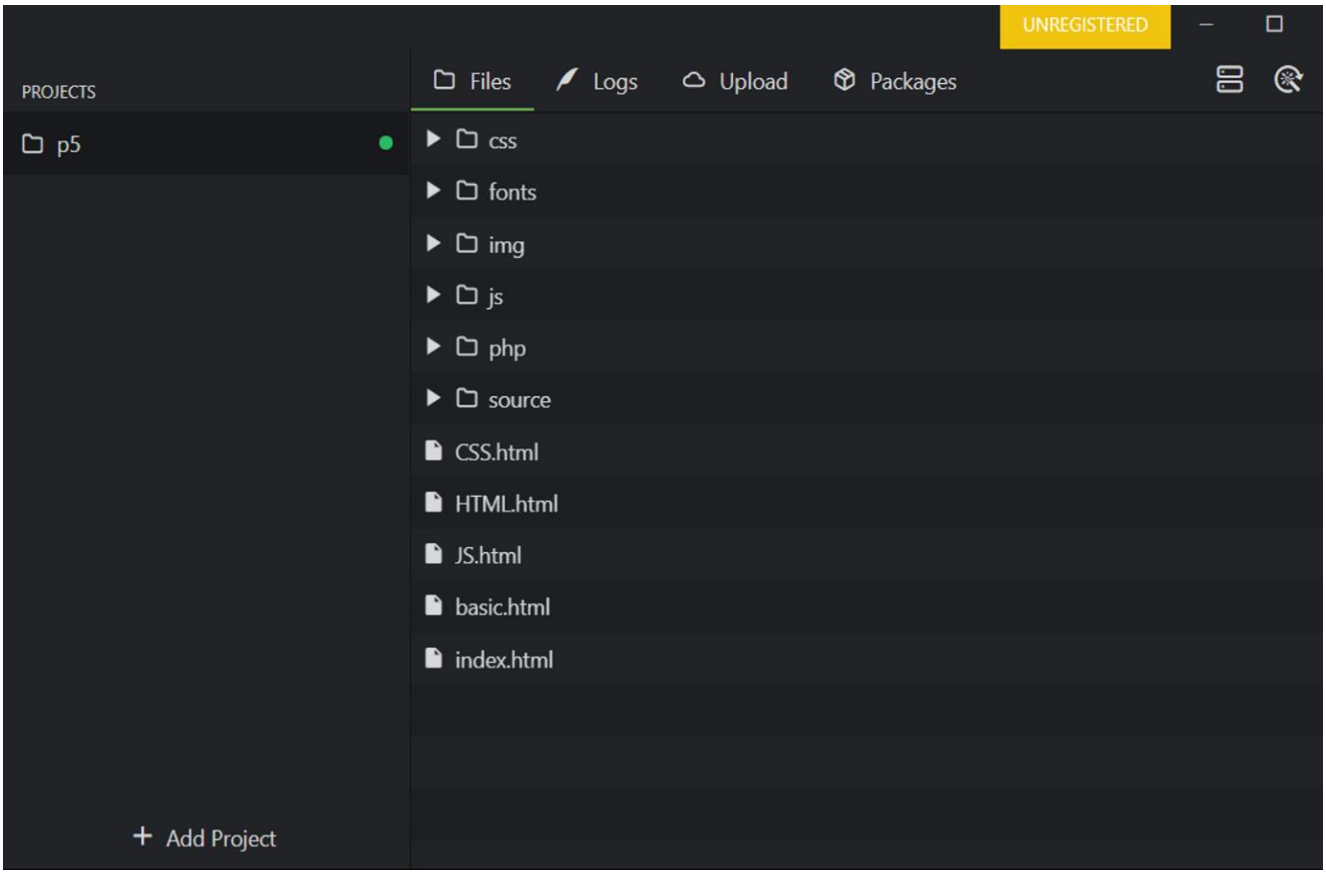

Рисунок 3.2 – Інтерфейс програми-збирача Prepros

З допомогою даної програми, розроблювану web-систему можна запустити в режимі попереднього перегляду, з автоматичним оновленням. Також Prepros містить у собі консоль, завдяки якій можна швидко виявити можливі помилки у написанні коду.

3.3 Розробка програмних засобів для впровадження адаптивної верстки

Станом на сьогодні адаптивна верстка є не просто популярним рішенням, а необхідністю для кожної web-системи, адже за останні десять років доля мобільного трафіку користувачів мережі інтернет, що відвідують web-системи виросла майже вдвічі, з шести до пятидесяти шести відсотків [18]. Все більше розробників притримуються принципу Mobile First.

Стратегія Mobile First полягає у проектування інтерфейсу web-системи, орієнтуючись перш за все на мобільні пристрої. За таким принципом спочатку створюється інтерфейс для мобільних додатків, котрий потім адаптується розробниками для десктопних пристроїв [19].

Існує і протилежний принцип, що є більш традиційним – Desktop First. Використання користувачем десктопних пристроїв забезпечить найбільш комфортні умови вивчення web-розробки, тому при виконанні бакалаврської дипломної роботи було прийнято рішення дотримуватися стратегії Desktop First.

Для впровадження адаптивної верстки web-системи, необхідно перш за все визначити тип пристрою користувача (рис. 3.3) (рис. 3.4).

```
1  "use strict"
2
3  const isMobile = {
4      Android: function () {
5          return navigator.userAgent.match(/Android/i);
6      },
7      BlackBerry: function () {
8          return navigator.userAgent.match(/BlackBerry/i);
9      },
```

Рисунок 3.3 – Визначення типу пристрою користувача

```
    iOS: function () {
        return navigator.userAgent.match(/iPhone|iPad|iPod/i);
    },
    Opera: function () {
        return navigator.userAgent.match(/Opera Mini/i);
    },
    Windows: function () {
        return navigator.userAgent.match(/IEMobile/i);
    },
    any: function () {
        return (
            isMobile.Android() ||
            isMobile.BlackBerry() ||
            isMobile.iOS() ||
            isMobile.Opera() ||
            isMobile.Windows());
    }
};
```

Рисунок 3.4 – Визначення типу пристрою користувача

Після визначення типу пристрою користувача, необхідно адаптувати інтерфейс web-системи під заданий тип.

Для мобільних пристроїв, користуючись методом classList.add, додається клас «_touch» (рис. 3.5).

```
if (isMobile.any()) {
    document.body.classList.add('_touch');
```

Рисунок 3.5 – Додання класу «_touch»

Даний клас визначено в каскадних таблицях стилів, він адаптує елементи інтерфейсу web-системи для пристроїв з тачскріном.

Наступним кроком є адаптація під мобільні пристрої елементів з випадаючим меню, необхідним для навігації (рис. 3.6).

```
let menuArrows = document.querySelectorAll('.menu__arrow');
if (menuArrows.length > 0){
   for (let index = 0; index < menuArrows.length; index++){
      const menuArrow = menuArrows[index];
      menuArrow.addEventListener("click", function(e){
         menuArrow.parentElement.classList.toggle('_active');
      });
   }
```

Рисунок 3.6 – Адаптація під мобільні пристрої елементів з випадаючим меню

Для цього для випадаючого меню було додано елемент «стрілка», для його виклику. За допомогою методу addEventListener(), було реалізовано додання або вилучення класу «_active» при умові кліку по даному елементу. Клас визначено в CSS, він необхідний для відображення та анімації випадаючого меню на пристроях з тачскріном.

Для десктопних пристроїв метод classList.add, додає клас «_pc». Даний клас необхідний для відображення відповідного інтерфейсу на персональних комп’ютерах та ноутбуках (рис. 3.7).

```
} else{
   document.body.classList.add('_pc');
}
```

Рисунок 3.7 – Додання класу «_pc»

Окрім того, розміри дисплею мобільних пристроїв не дозволяють коректно відобразити панель навігації. Для вирішення цієї проблеми було прийнято рішення розробити меню «Бургер» (рис. 3.8), для відображення панелі навігації на пристроях з тачскріном.

```
//Меню бургер

const iconMenu = document.querySelector('.menu__icon');
const menuBody = document.querySelector('.menu__body');
if(iconMenu){
   iconMenu.addEventListener("click", function(e){
      document.body.classList.toggle('_lock');
      iconMenu.classList.toggle('_active');
      menuBody.classList.toggle('_active');
   });
}
```

Рисунок 3.8 – Розробка меню «Бургер»

Класи «.menu__icon» та «.menu__body» задають стилі для іконки меню-бургера та самого меню відповідно. За допомогою методу addEventListener(), було реалізовано додання або вилучення класу «_lock», при умові кліку по елементу. Даний клас визначено в таблицях каскадних стилів, для блокування скролу web-системи, при відкритому меню навігації.

Також було додано клас «_active», для іконки меню та самого меню, що відповідає за їх анімацію.

Активацію відповідних класів, для різних типів пристроїв, реалізовано за допомогою медіазапитів. Усі необхідні функції реалізовано за допомогою засобів мови програмування JavaScript.

Окрім цього необхідно переконатися, що розроблювана система буде однаково відображатися, незалежно від браузера користувача. Різні браузери мають свої особливості відображення тих чи інших елементів.

Тому було вирішено прописати обнуляючі стилі, використовуючи засоби CSS. Відповідний .css файл буде підключено до усіх сторінок web-системи.

Для початку потрібно обнулити внутрішні та зовнішні відступи, а також границі для усіх елементів сторінок web-системи, звернувшись до селектора *, що позначає усі елементи (рис. 3.9).

```
1  *{
2      padding: 0;
3      margin: 0;
4      border: 0;
5  }
```

Рисунок 3.9 – Обнулення внутрішніх та зовнішніх відступів

Далі необхідно змінити метод підрахунку розмірів елементів (рис. 3.10), що можна реалізувати за допомогою border-box. Завдяки цьому, розміри елементів будуть одразу включати в себе товщину границі елементу, та внутрішні відступи, незалежно від браузера.

```
*,*:before,*:after{
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
}
```

Рисунок 3.10 – Зміна методу підрахунку розмірів елементів

Наступним кроком прибираються лінії обведення для активних елементів (рис. 3.11), що додаються деякими браузерами.

```
:focus,:active{outline: none;}
a:focus,a:active{outline: none;}
```

Рисунок 3.11 – Обнулення ліній обведення активних елементів

До вказаних HTML5 тегів застосовано параметр display зі значенням block (рис. 3.12), аби перетворити їх на блокові елементи.

```
nav,footer,header,aside{display: block;}
```

Рисунок 3.12 – Перетворення вказаних HTML5 тегів на блокові елементи

Звернувшись до тегів html та body, за допомогою вказаних властивостей урівнюється поведінка шрифтів та елементів, для різних браузерів (рис. 3.13).

```
html,body{
    height: 100%;
    width: 100%;
    font-size: 100%;
    line-height: 100%;
    font-size: 14px;
    -ms-text-size-adjust: 100%;
    -moz-text-size-adjust: 100%;
    -webkit-text-size-adjust: 100%;
}
```

Рисунок 3.13 – Урівнення поведінки шрифтів та елементів для різних браузерів

Для тегів input, button та textarea, що використовуються у формах, необхідно задати значення inherit параметру font-family, аби дані теги наслідували сімейства шрифтів (рис. 3.14).

```
25
26  input,button,textarea{font-family: inherit;}
27
```

Рисунок 3.14 – Наслідуваня сімейства шрифтів тегами форми

Наступним кроком було додано параметри для заданих тегів (рис. 3.15), аби прибрати особливості відображення тих чи інших елементів для відповідних браузерів, наприклад прибирається хрестик у полях введення, що по замовчуванню додається браузером Internet Explorer (рис. 3.16).

Також було обнулено параметри стандартного відображення списків, для їх подальшої стилізації засобами каскадних таблиць стилів.

```
28  input::-ms-clear{display: none;}
29  button{cursor: pointer;}
30  button::-moz-focus-inner{padding: 0;border: 0;}
31  a, a:visited{text-decoration: none;}
32  a:hover{text-decoration: none;}
33  ul li{list-style: none;}
34  img{vertical-align: top;}
35
```

Рисунок 3.15 – Обнулення стилізації елементів для різних браузерів

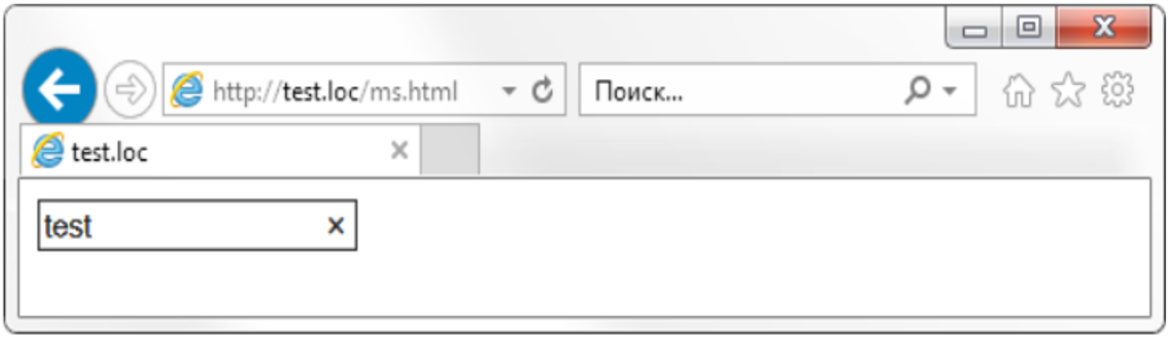

Рисунок 3.16 – Приклад відображення поля input у браузері Internet Explorer

Останній крок – обнулення стилів для заголовків усіх рівнів, на всіх сторінках web-системи (рис. 3.17).

```
35
36  h1,h2,h3,h4,h5,h6{font-size: inherit;font-weight: 400;}
37  /*------------------------------------------------------------------*/
```

Рисунок 3.17 – Обнулення стилів для заголовків

3.4 Розробка програмних засобів для навігації у web-системі

Для зручності користувача, при вивченні розміщених у системі матеріалів, необхідно забезпечити створювану web-систему відповідною навігацією.

Основа навігаційної системи реалізована за допомогою засобів мови розмітки гіпертексту HTML та каскадних таблиць стилів CSS. Але для зручності користувача, за допомогою засобів мови програмування JavaScript, було реалізовано плавний перехід між блоками при активації посилання в межах однієї сторінки.

Перехід реалізовано з використанням атрибуту «data-goto» (рис. 3.18).

```
57  //Прокрутка при натисканні
58
59  const menuLinks = document.querySelectorAll('.menu__link[data-goto]');
60  if (menuLinks.length > 0){
61      menuLinks.forEach(menuLink => {
62          menuLink.addEventListener("click", onMenuLinkClick);
63      });
```

Рисунок 3.18 – Перехід за посиланням в межах однієї сторінки

Для переходу прописано функції onMenuLinkClick (рис. 3.19), та onPageLinkClick (рис. 3.20), для посилань з панелі навігації та посилань зі змісту відповідно.

```
function onMenuLinkClick(e) {
   const menuLink = e.target;
   if(menuLink.dataset.goto && document.querySelector(menuLink.dataset.goto)){
      const gotoBlock = document.querySelector(menuLink.dataset.goto);
      const gotoBlockValue =gotoBlock.getBoundingClientRect().top +
      pageYOffset - document.querySelector('header').offsetHeight;

      if (iconMenu.classList.contains('_active')){
         document.body.classList.remove('_lock');
         iconMenu.classList.remove('_active');
         menuBody.classList.remove('_active');
      }

      window.scrollTo({
         top: gotoBlockValue,
         behavior: "smooth"
      });
      e.preventDefault();
   }
}
```

Рисунок 3.19 – функція onMenuLinkClick

Функція onMenuLinkClick додатково перевіряє елемент іконки меню, на наявність класу «_active», та за потреби виконує дії необхідні для того, аби перехід працював коректно, на пристроях з тачскріном, при відкритому меню навігації.

```
94
95      function onPageLinkClick(e) {
96          const pageLink = e.target;
97          if(pageLink.dataset.goto && document.querySelector(pageLink.dataset.goto)){
98              const gotoBlock = document.querySelector(pageLink.dataset.goto);
99              const gotoBlockValue = gotoBlock.getBoundingClientRect().top +
100             pageYOffset - document.querySelector('header').offsetHeight;
101
102             window.scrollTo({
103                 top: gotoBlockValue,
104                 behavior: "smooth"
105             });
106             e.preventDefault();
107         }
108     }
109 }
```

Рисунок 3.20 – функція onPageLinkClick

3.5 Розробка блоку реєстрації

Для отримання доступу до навчальних матеріалів web-системи, користувач повинен пройти реєстацію, або ж авторизуватися, якщо він уже має створений акаунт.

При реєстрації нового акаунта, дані користувача зберігаються в базі даних. Базою даних називається впорядкований набір даних, організованих відповідно до певної концепції, що характеризує ці даних та взаємозв’язки між їх елементами [20].

Інтерфейс блоку реєстрації реалізовано за допомогою засобів мови розмітки гіпертексту HTML та каскадних таблиць стилів CSS. Сама база даних створена засобами мови структурованих запитів SQL (рис. 3.21).

```
--
-- Table structure for table `users`
--

CREATE TABLE `users` (
  `id` int(11) UNSIGNED NOT NULL,
  `login` varchar(100) NOT NULL,
  `pass` varchar(32) NOT NULL,
```

```
  `name` varchar(50) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `users`
--
ALTER TABLE `users`
  ADD UNIQUE KEY `id` (`id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `users`
--
ALTER TABLE `users`
  MODIFY `id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;
```

Рисунок 3.21 – Створення бази даних

Алгоритми реєстрації та авторизації реалізовано за допомогою засобів мови програмування PHP.

Оголошено змінні «login», «name» та «pass», що приймають дані користувача в процесі реєстрації (рис. 3.22).

```
1  <?php
2      $login = filter_var(trim($_POST['login']),
3      FILTER_SANITIZE_STRING);
4      $name = filter_var(trim($_POST['name']),
5      FILTER_SANITIZE_STRING);
6      $pass = filter_var(trim($_POST['pass']),
7      FILTER_SANITIZE_STRING);
```

Рисунок 3.22 – Оголошення змінних «login», «name» та «pass»

Введені користувачем дані проходять перевірку на валідність. При виявлені некоректних даних, система виводить відповідне повідомлення за допомогою функції echo (рис. 3.23).

```
    if(mb_strlen($login) < 5 || mb_strlen($login) > 90) {
        echo "Недопустима довжина логіну";
        exit();
    } else if(mb_strlen($name) < 1 || mb_strlen($login) > 50) {
        echo "Недопустима довжина імені";
        exit();
    } else if(mb_strlen($pass) < 4 || mb_strlen($login) > 8) {
        echo "Недопустима довжина пароля (2-6 символів)";
        exit();
    }
```

Рисунок 3.23 – Перевірка на валідність введеніих користувачем даних

До паролю користувача додається визначений сталий набір символів, після чого утворений новий пароль хешується за допомогою функції md5 (рис. 3.24). Дана функція обчислює MD5-хеш рядку string, використовуючи алгоритм MD5 RSA Data Security, та повертає хеш [21].

```
    $pass = md5($pass."ckeuvichoistel");
```

Рисунок 3.24 – Застосування функції md5

Після хешування паролю, дані користувача зберігаються до бази даних (рис. 3.25), а користувач отримує доступ до навчальних матеріалів web-системи.

```
    require "connect.php";
    $mysql->query("INSERT INTO `users` (`login`, `pass`, `name`)
    VALUES('$login', '$pass', '$name')");

    $mysql->close();
```

Рисунок 3.25 – Збереження інформації до бази даних

Для користувачів, що вже мають зареєстрований у системі акаунт, для доступу до навчальних матеріалів, потрібно пройти авторизацію (рис. 3.26).

При авторизації система звіряє логін та пароль введені користувачем з даними що зберігаються в базі даних, попередньо обчислюючи хеш-код, з доданням сталого набору символів, через функцію md5.

```php
<?php
    $login = filter_var(trim($_POST['login']),
    FILTER_SANITIZE_STRING);
    $pass = filter_var(trim($_POST['pass']),
    FILTER_SANITIZE_STRING);

    $pass = md5($pass."ckeuvichoistel");
```

Рисунок 3.25 – Перевірка даних при авторизації

Якщо інформація введена користувачем не відповідає даним, збереженим у базі даних, то система виводить відповідне повідомлення (рис. 3.26) (рис. 3.27), після чого користувач повертається на головну сторінку, де зможе пройти авторизацію чи реєстрацію знову.

```php
require "connect.php";

$result = $mysql->query("SELECT * FROM `users` WHERE `login` = '$login'
   AND `pass` = '$pass'");
$user = $result->fetch_assoc();
if(count($user) == 0) {
```

Рисунок 3.26 – Вивід повідомлення про помилку при спробі авторизації

```php
require "connect.php";

$result = $mysql->query("SELECT * FROM `users` WHERE `login` = '$login'
   AND `pass` = '$pass'");
$user = $result->fetch_assoc();
if(count($user) == 0) {
   echo "Користувача не знайдено";
   exit();
}

$mysql->close();
```

Рисунок 3.27 – Вивід повідомлення про помилку при спробі авторизації

При вдалій авторизації користувач отримує доступ до навчальних матеріалів web-системи.

3.6 Розробка блоку тестування

Для підвищення рівня освоєння наданого матеріалу користувачем web-системи, було прийнято рішення розробити блоки з тестуванням для окремих тем. Інтерфейс боку розроблено за допомогою засобів мови розмітки гіпертексту HTML та каскадних таблиць стилів CSS. Тест створено за допомогою засобів мови програмування JavaScript, без серверної частини, питання та варіанти відповідей зберігаються в масиві (рис. 3.28).

```
const DATA = [
    {
        question: 'Питання 1: питання',
        answers: [
            {
                id: '1',
                value: 'Відповідь 1',
                correct: true,
            },
            {
                id: '2',
                value: 'Відповідь 2',
                correct: false,
            },
            {
                id: '3',
                value: 'Відповідь 3',
                correct: false,
            },
        ]
    },
```

Рисунок 3.28 – Приклад елементу масиву

Відповідні одному питанню варіанти відповідей, що теж формують окремий масив. Кожен варіант відповіді, окрім значення, також має унікальний ідентифікатор id.

Оголошення констант за допомогою методу getElementById (рис. 3.29).

```
const quiz = document.getElementById('quiz');
const questions = document.getElementById('questions');
const indicator = document.getElementById('indicator');
const results = document.getElementById('results');
const btnNext = document.getElementById('btn-next');
const btnRestart = document.getElementById('btn-restart');
```

Рисунок 3.29 – Оголошення констант за допомогою методу getElementById

Функція renderQuestion (рис. 3.30), відповідає за рендеринг питань.

```
const renderQuestions = (index) => {
   renderIndicator(index + 1);

   questions.dataset.currentStep = index;

```

Рисунок 3.30 – Функція renderQuestion

Функція renderAnswers відповідає за рендеринг відповідних, для поточного питання, варіантів відповідей (рис. 3.31) (рис. 3.32).

```
   const renderAnswers = () => DATA[index].answers
      .map((answer) => `
         <li>
            <label>
               <input class="answer-input" type="radio" name=${index}
               value=${answer.id}>
               ${answer.value}
            </label>
         </li>
```

Рисунок 3.31 – Функція renderAnswers

```
      `)
      .join('');

   questions.innerHTML = `
      <div class="quiz-questions-item">
        <div class="quiz-questions-item__question">${DATA[index].question}
        </div>
        <ul class="quiz-questions-item__answers">${renderAnswers()}</ul>
      </div>
   `;
};
```

Рисунок 3.32 – Функція renderAnswers

Функція renderResults (рис. 3.33) (рис. 3.34), використовується для демонстрації користувачу результатів тестування. За допомогою функції getClasname, система визначає який клас використовувати для кожного питання, при виведенні результатів на екран. Класи «answer--invalid», «answer--valid» та «answer--invalid--correct» визначені в каскадних таблицях стилів для стилізації вірних та не правильних відповідей користувача. Функція getAnswers дозволяє системі отримати варіанти відповідей для виводу на екран. Тоді як за допомогою методу forEach формується результат для виводу.

```
99  const renderResults = () => {
100     let content = '';
101
102     const getClassname = (answer, questionIndex) => {
103        let classname = '';
104
105        if (!answer.correct && answer.id === localResults[questionIndex]) {
106           classname = 'answer--invalid';
107        } else if (answer.correct && answer.id === localResults[questionIndex]
108        {
109           classname = 'answer--valid';
110        } else if (answer.correct ) {
111           classname = 'answer--invalid--correct';
112        }
113
114        return classname;
115     };
```

Рисунок 3.33 – Функція renderResults

```
116
117     const getAnswers = (questionIndex) => DATA[questionIndex].answers
118         .map((answer) => `<li class=${getClassname(answer, questionIndex)}>
119             ${answer.value}</li>`)
120         .join('');
121
122     DATA.forEach((question, index) => {
123         content += `
124             <div class="quiz-results-item">
125                 <div class="quiz-results-item__question">
126                 ${question.question}</div>
127                     <ul class="quiz-results-item__answers">
128                     ${getAnswers(index)}</ul>
129                 </div>
130         `;
131     });
132
133     results.innerHTML = content;
134 };
135
```

Рисунок 3.34 – Функція renderResults

Для рендерингу індикатора питань було реалізовано функцію renderIndicator (рис. 3.35).

```
136 const renderIndicator = (currentStep) => {
137     indicator.innerHTML = `${currentStep}/${DATA.length}`;
138 };
139
```

Рисунок 3.35 – Функція renderIndicator

Метод addEventListener реєструє зміни для блоку «quiz» (рис. 3.36), якщо користувач обирає певний варіант відповіді, то система зберігає його у тимчасових результатах, для подальшого виводу на екран. Також, після вибору варіанта відповіді, розблоковується елемент кнопка «Далі», що є недоступною по замовчуванню (рис. 3.37).

```
139
140 quiz.addEventListener('change', (event) =>{
141     //логіка відповіді
142     if(event.target.classList.contains('answer-input')) {
143         localResults[event.target.name] = event.target.value;
144         btnNext.disabled = false;
145     }
146 });
```

Рисунок 3.36 – Метод для реєстрації змін для блоку «quiz»

```
165
166         btnNext.disabled = true;
```

Рисунок 3.37 – блокування елементу кнопка «Далі»

Перед переходом до наступного питання користувач може вільно обрати інший варіант відповіді.

За допомогою методу addEventListener програма реєструє натискання в межах блоку «quiz» (рис. 3.38), якщо цільовий елемент містить клас «btn-next», то програма визначає індекс питання.

```
148 quiz.addEventListener('click', (event) =>{
149
150     if(event.target.classList.contains('btn-next')) {
151         const nextQuestionIndex = Number(questions.dataset.currentStep) + 1;
152
153         if(DATA.length === nextQuestionIndex) {
154             //перехід до результатів
155             questions.classList.add('questions--hidden');
156             indicator.classList.add('indicator--hidden');
157             results.classList.add('indicator--visible');
158             btnNext.classList.add('btn-next--hidden');
159             btnRestart.classList.add('btn-restart--visible');
160             renderResults();
```

Рисунок 3.38 – Метод для реєстрації натискань в межах блоку «quiz»

Якщо індекс рівний числу кількості елементів масиву питань, то система переходить до виводу результатів тестування користувача на екран. В іншому випадку відбувається перехід до наступного питання. Після переходу елемент кнопка «Далі» знову стає неактивною, допоки користувач не обере варіант відповіді для поточного питання.

Якщо цільовий елемент містить клас «btn-restart» (рис. 3.39), програма обнуляє масив збережених відповідей, та результати для виводу. Також за допомогою методу classList.remove, програма видаляє додані, під час процесу тестування, до елементів класи, а функція рендерингу питань повертається до першого питання. Усе це дозволяє користувачу пройти тестування заново будь-яку кількість разів.

```
161         } else {
162             //перехід до наступного питання
163             renderQuestions(nextQuestionIndex);
164         }
165 
166         btnNext.disabled = true;
167     }
168 
169     if(event.target.classList.contains('btn-restart')) {
170         localResults = {};
171         results.innerHTML = '';
172 
173         questions.classList.remove('questions--hidden');
174         indicator.classList.remove('indicator--hidden');
175         results.classList.remove('indicator--visible');
176         btnNext.classList.remove('btn-next--hidden');
177         btnRestart.classList.remove('btn-restart--visible');
178 
179         renderQuestions(0);
180     }
181 });
```

Рисунок 3.39 – Обнулення масиву збережених відповідей

Для виводу результатів на екран користувача, спочатку необхідно стилізувати їх засобами каскадних таблиць стилів. Класи «answer--invalid», «answer--valid» та «answer--invalid--correct» (рис. 3.40), додаються до відповідей засобами мови програмування JavaScript, за допомогою функції getClasname. Клас «answer--valid» створено для стилізації правильних відповідей наданих користувачем. Тоді як класи «answer--invalid» та «answer--invalid--correct» реалізовані для стилізації невірних відповідей наданих користувачем та правильних варіантів для цих питань відповідно.

```
80  .answer--invalid {
81      color: red;
82      font-weight: 600;
83      font-size: 24px;
84      line-height: 30px;
85      position: relative;
86
87      &::after{
88          position: absolute;
89          content: '';
90          width: 30px;
91          height: 30px;
92          //border: 1px solid #000;
93          background: url('../img/icons/icon_cross.png') no-repeat;
94      }
95  }
96
97  .answer--valid {
98      color: green;
99      font-weight: 600;
100     font-size: 28px;
101     line-height: 34px;
102
103     &::after{
104         position: absolute;
105         content: '';
106         width: 34px;
107         height: 34px;
108         //border: 1px solid #000;
109         background: url('../img/icons/icon_tick.png') no-repeat;
110     }
111 }
112
113 .answer--invalid--correct{
114     color: green;
115     font-weight: 600;
116     font-size: 28px;
117     line-height: 34px;
118 }
```

Рисунок 3.40 – Стилізація відповідей для виведення на екран

3.7 Розробка програмних засобів для перемикання теми web-системи

Для зручності користувача було прийнято рішення створити елемент перемикач (рис. 3.41) (рис. 3.42), що дозволить користувачу обипати між світлою та темною темами системи.

Рисунок 3.41 – елемент перемикач для активної світлої теми

Рисунок 3.42 – елемент перемикач для активної темної теми

Приклад відображення web-системи із зістосуванням темної теми (рис. 3.43).

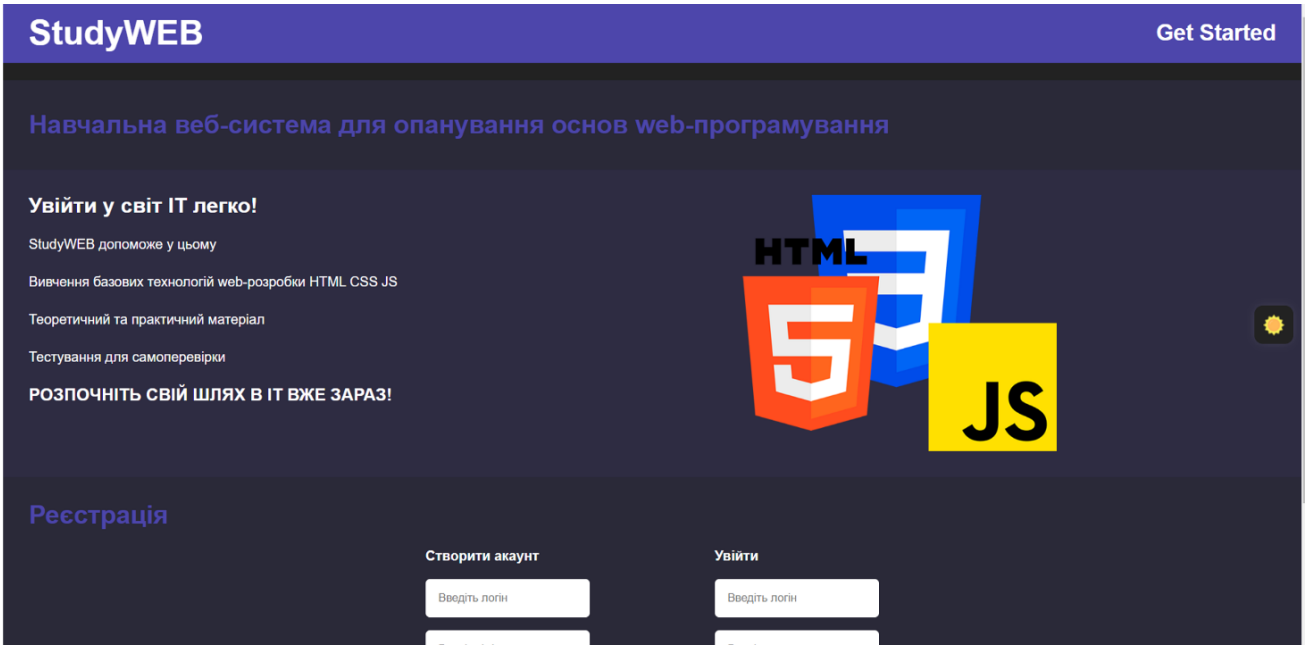

Рисунок 3.43 – Відображення головної сторінки для активної темної теми

Для реалізації перемикання було створено два .css файли, кожен з яких відповідав за визначену тему.

Перемикання між .css файлами реалізовано засобами мови програмування JavaScript.

В .html файлах прописано посилання для підключення відповідного .css файлу, в залежності від вибору користувача (рис. 3.44).

```
9    <link rel="stylesheet" title="theme" href="#">
```

Рисунок 3.44 – Посилання для підключення відповідного .css файлу

Спочатку елементи кнопок для перемикання теми було поміщено у змінну, за допомогою методу .querySelectorAll() (рис. 3.45).

```
1  let changeThemeButtons = document.querySelectorAll('.changeTheme');
```

Рисунок 3.45 – Метод .querySelectorAll()

Користуючись методом .addEventListener() було реалізовано виклик відповідної функції при натисканні на кнопку перемикання (рис. 3.46).

```
2
3  changeThemeButtons.forEach(button => {
4      button.addEventListener('click', function () {
5          let theme = this.dataset.theme;
6          applyTheme(theme);
7      });
8  });
9
```

Рисунок 3.46 – Виклик відповідної функції при натисканні на кнопку перемикання

Далі в змінну записується назва відповідної теми із атрибута «data-theme» визначеного для елементу перемикача у .html файлах. Функція що використовується для зміни тем приймає назву необхідної теми (рис. 3.47).

```
9
10  function applyTheme(themeName) {
11      document.querySelector('[title="theme"]').setAttribute('href', `css/theme-${themeName}.css`);
12      changeThemeButtons.forEach(button => {
13          button.style.display = 'block';
14      });
15      document.querySelector(`[data-theme="${themeName}"]`).style.display = 'none';
16      localStorage.setItem('theme', themeName);
17  }
```

Рисунок 3.47 – Запис назви відповідної теми із атрибута «data-theme»

Функція applyTheme() записує шлях до вфдповфдного файлу .css теми у визначений link. Функція відображає елементи кнопок перемикання, проте за допомогою методу .querySelector() приховує від користувача кнопку для активної теми.

Окрім цього необхідно реалізувати збереження обраної користувачем теми web-системи, при переході на інші сторінки системи, перезавантажені поточної сторінки, та за умови закриття браузера користувачем.

Для збереження поточної теми було використано локальне сховище браузера localStorage (рис. 3.48).

```
18
19  let activeTheme = localStorage.getItem('theme');
20
21  if(activeTheme === null || activeTheme === 'light') {
22      applyTheme('light');
23  } else if (activeTheme === 'dark') {
24      applyTheme('dark');
25  }
```

Рисунок 3.48 – Використання локального сховища браузера

Спочатку виконується перевірка локального сховища браузера, щоб виявити чи збережено значення використовуваної теми, дане значення записується у змінну.

Якщо значення не було записано раніше, або ж це значення «light», що відповідає світлій темі, то до системи застосовується світла тема відображення.

Якщо записано значення «dark», що відповідає темній темі, то до системи застосовується темна тема відображення.

Теми задають відповідну стилізацію для web-системи, визначену за допомогою засобів каскадних таблиць стилів CSS.

3.8 Висновки

У третьому розділі було проведено аналіз та обґрунтування вибору засобів для реалізації програмної компоненти web-системи, технологій, середовищ та додаткового програмного забезпечення для створення кінцевого продукту. Було розроблено програмні засоби для впровадження адаптивної верстки для web-системи за принципом Desktop First. Також було створено програмні засоби для навігації у web-системі. Розроблено програмні компоненти для блоку реєстрації та авторизації користувачів, а також для тестування знань користувача.

4 ТЕСТУВАННЯ WEB-СИСТЕМИ

4.1 Аналіз методів тестування web-систем

Перед повноцінним запуском web-системи, вона повинна пройти етап тестування, для виявлення та виправлення помилок. Цей етап є фінальним, та полягає в перевірці працездатності всього функціоналу системи та її відповідності поставленій задачі.

Основні види методів тестування web-систем:

- функціональне тестування;
- тестування зручності користування;
- конфігураційне тестування;
- тестування продуктивності;
- тестування безпеки.

Функціональне тестування – це найважливіший та наймасштабніший із видів тестування. Основною метою даного виду тестування є перевірка коректності роботи основного функціоналу web-системи. Для цього необхідно провести ряд перевірок:

- перевірка посилань, як внутрішніх так і вихідних;
- перевірка форм для взаємодії системи та користувача;
- перевірка системи реєстрації та авторизації;
- перевірка системи на наявність синтаксичних помилок.

Тестування зручності користування необхідне для оцінки дизайну та функціоналу системи з погляду користувача. Даний вид тестування призначений для перевірки структурованості web-системи. Також важливо виявити та усунути можливі помилки у граматиці та орфографії.

Конфігураційне тестування застосовується для перевірки web-системи на коректність відображення при різних розширеннях, пристроях та у різних

браузерах. Важливою складовою даного виду тестування є перевірка адаптивності системи.

Тестування продуктивності – вид перевірки, необхідний для оцінки стабільності роботи web-системи.

Тестування безпеки необхідне, щоб виявити слабкі місця системи, що можуть призвести до втрати даних або порушення її роботи, а також усунення імовірних загроз.

4.2 Тестування web-системи

Web-тестування – це процес тестування web-систем або -додатків на наявність потенційних помилок, що проводиться перед опублікуванням. Виконувати тестування потрібно аби переконатися, що web-система працює належним чином і може бути прийнята користувачами в режимі реального часу. Перевірка дизайну та функціональність інтерфейсу є головними етапами тестування web-систем [22].

При вході до головної сторінки (рис. 4.1) web-системи користувача зустрічає блок для реєстрації та авторизації користувачів (рис. 4.2).

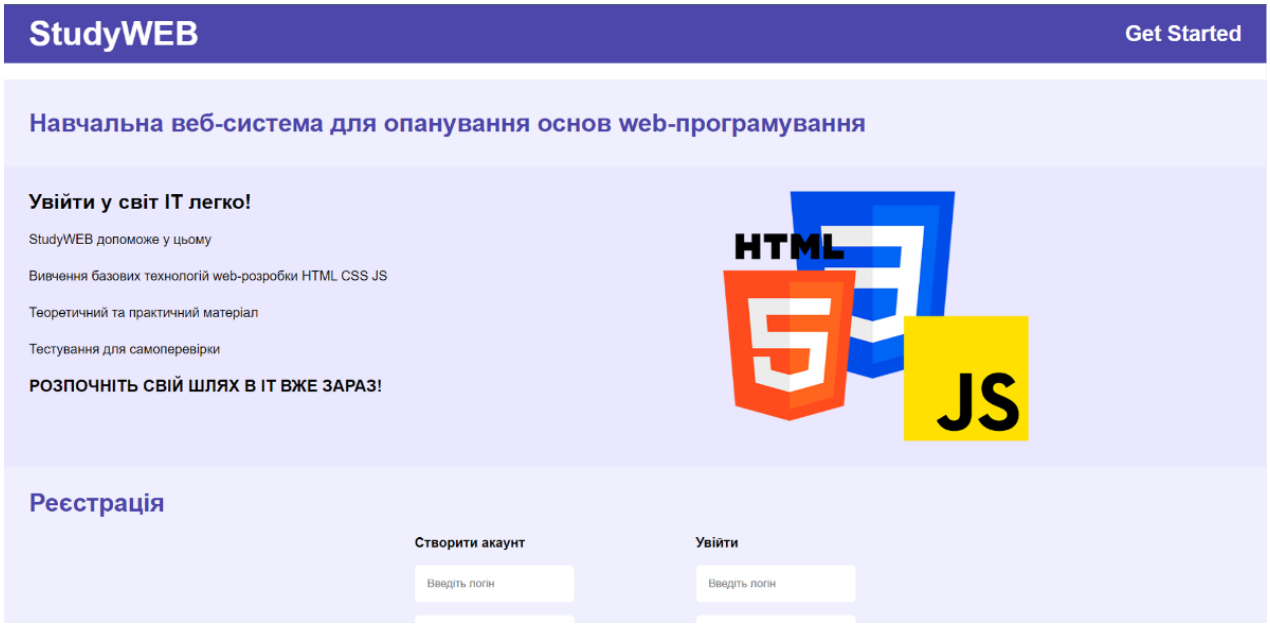

Рисунок 4.1 – Головна сторінка web-системи

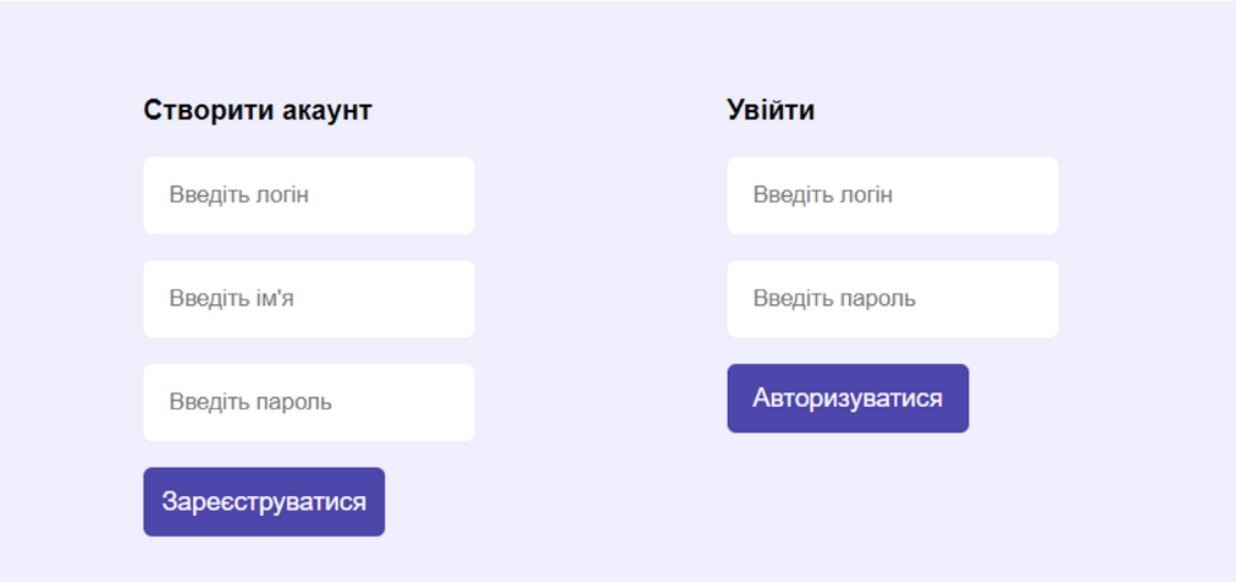

Рисунок 4.2 – Блок для реєстрації у web-системі

Таблиця 4.1 – Тестові випадки

Тестовий випадок	Результат виконання
Ввід некоректних даних у поля для реєстрації/авторизації	Система виводить користувачу відповідне повідомлення
Успішне проходження реєстрації	Система вносить дані користувача в базу даних
Реєстрація/авторизація	Перехід на сторінку basic.html

Після реєстрації у системі користувач переходить на сторінку з навчальним матеріалом для початку вивчення основ web-програмування. Для пареміщення в межах web-системи, було розроблено навігаційну панель (рис. 4.3).

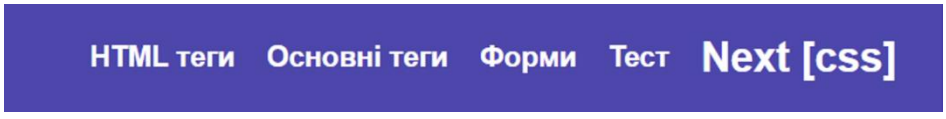

Рисунок 4.3 – Панель для навігації у web-системі

Панель містить назву сайту, що також є посиланням на головну сторінку, посилання на ключові блоки в межах даної сторінки, та посилання для переходу

до наступної сторінки. Крім того, дане посилання являє собою випадаюче меню, що містить у собі посилання на усі сторінки сайту (рис. 4.4).

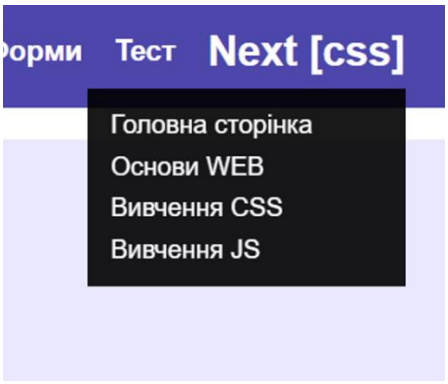

Рисунок 4.4 – Випадаюче меню панелі навігації

Таблиця 4.2 – Тестові випадки

Тестовий випадок	Результат виконання
Натискання на назву web-системи на панелі навігації	Перехід на головну сторінку (index.html)
Натискання на назву web-системи на панелі навігації головної сторінки	Нічого не відбувається
Натискання на посилання на панелі навігації	Перехід до необхідного блоку в межах даної сторінки
Наведення миші на посилання для переходу до наступної сторінки	Відкриття випадаючого меню
Натискання на посилання для переходу до наступної сторінки	Перехід на наступну сторінку
Натискання на посилання з випадаючого меню	Перехід на обрану сторінку
Натискання кнопку зміеи теми web-системи	Перемикання системи на відповідну тему

Для мобільних пристроїв панель навігації відображається відповідно до параметру ширини дисплею. Для виклику випадаючого меню поруч з посиланням додано елемент «стрілка» (рис. 4.5).

Рисунок 4.5 – Елемент «стрілка» для виклику меню

Для пристроїв з шириною дисплею меншою ніж 767 пікселів усі елементи навігаційної панелі, окрім назви системи, поміщаються у меню «бургер», що відкривається та закривається при натисканні на іконку меню (рис. 4.6) (рис. 4.7).

Рисунок 4.6 – Елемент іконка меню

Рисунок 4.7 – Елемент іконка меню при відкритому меню навігації

Для перевірки адаптивності системи було використано панель інструментів розробника браузера Google Chrome (рис. 4.8).

Панель інструментів розробника дозволяє web-розробникам тестувати та налагоджувати код web-сторінки. Дана панель містить інструменти, що використовуються для тестування як інтерфейсу, так і функціоналу web-системи.

У більшості популярних браузерів, таких як Google Chrome, Firefox, Internet Explorer, Safari, Microsoft Edge та Opera інструменти web-розробки заздалегідь вбудовані. Інструменти web-розробки дозволяють перевірити роботу web-системи для різних розширень, використовуючи один пристрій [23].

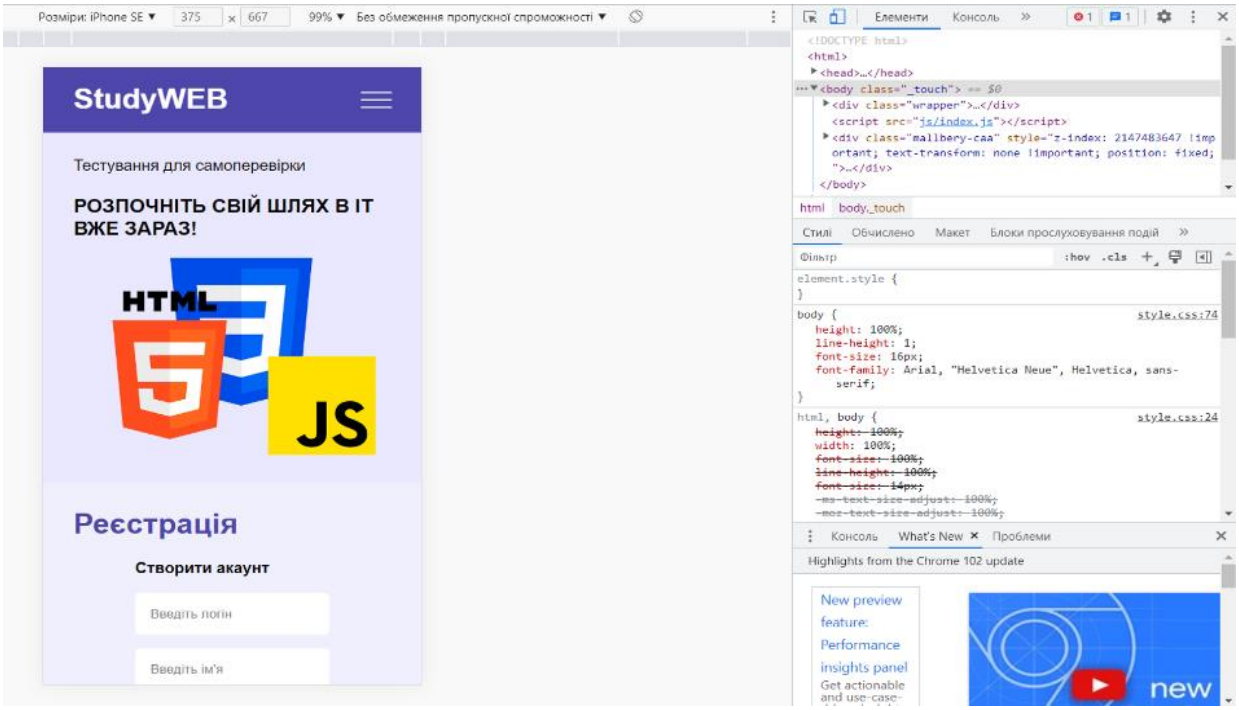

Рисунок 4.8 – Перевірка адаптивності web-системи

Графічний інтерфейс на мобільному пристрої відображається коректно. Тому можна зробити висновок, що web-систему адаптовано для відображення на мобільних пристроях.

Таблиця 4.3 – Тестові випадки

Тестовий випадок	Результат виконання
Натискання на елемент «стрілка»	Відкриття або закриття випадаючого меню
Натискання на елемент іконки меню	Відкриття меню навігації, анімована зміна форми іконки
Натискання на елемент іконки меню при відкритому меню навігації	Закриття меню навігації, анімована зміна форми іконки
Натискання на посилання в межах сторінки в меню навігації	Перехід до заданого блоку, закриття меню навігації

Продовження таблиці 4.3

Натискання на посилання для переходу до наступної сторінки в меню навігації	Перехід на наступну сторінку, закриття меню навігації
Натискання на елемент «стрілка» в меню навігації	Відкриття або закриття випадаючого меню
Натискання на посилання з випадаючого меню в меню навігації	Перехід на обрану сторінку, закриття меню навігації
Відкриття меню навігації	Блокування скролінгу контенту сторінки

Для зручності користувача в орієнтації по навчальним матеріалам, було вирішено додати блок зі змістом. Кожен пункт змісту (рис. 4.9) є посиланням на визначену тему.

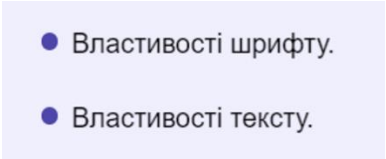

Рисунок 4.9 – Приклад елементів змісту

При наведенні мишкою на пункт змісту, текст посилання підкреслюється (рис. 4.10), що реалізовано в CSS за допомогою псевдокласу hover. При натисканні на посилання відбувається плавний перехід до визначеного блоку, в межах даної сторінки.

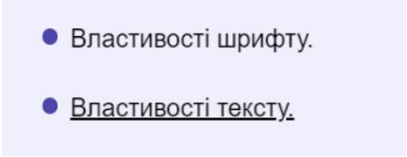

Рисунок 4.10 – Приклад застосування псевдокласу hover до елементу змісту

Для перевірки знань користувача було прийнято рішення реалізувати блоки з тестуванням по визначеним темам.

Кожен такий блок містить питання, варіанти відповідей з перемикачами для вибору (рис. 4.11), індикатор питання (рис. 4.12), а також кнопку «Далі» (рис. 4.13) для переходу до наступного питання та кнопку «Заново» (рис. 4.14) для повторного проходження тесту.

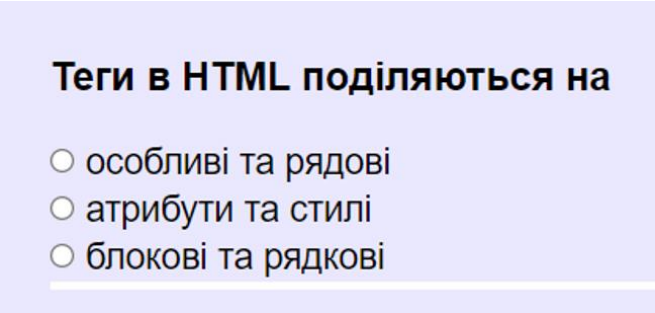

Рисунок 4.11 – Приклад питання та відповідей

Питання та відповідні їм варіанти відповідей зберігаються в масиві у відповідному .js файлі.

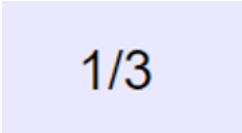

Рисунок 4.12 – Елемент індикатор питання блоку тестування

Індикатор питань показує користувачу кількість питань в даному блоці тестування, та номер поточного питання.

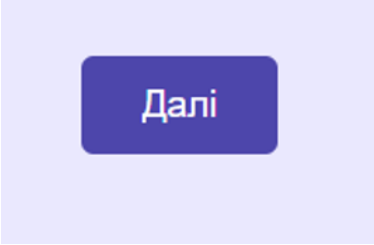

Рисунок 4.13 – Елемент кнопка «Далі» блоку тестування

Елемент кнопка «Далі» є недоступним для користувача, до моменту вибору варіанту відповіді поточного питання. Даний елемент приховується для користувача, при переході до результатів проходження тесту.

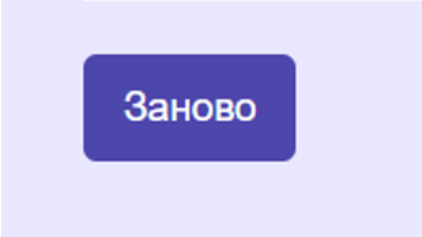

Рисунок 4.14 – Елемент кнопка «Заново» блоку тестування

Елемент кнопка «Заново» прихований від користувача, до моменту отримання та виведення на екран результатів тестування (рис. 4.15). Натиснувши на дану кнопку користувач завершує процес тестування.

Рисунок 4.15 – Приклад виведення результатів проходження тестування

При виведені результатів на екран, система виділяє правильні та неправильні відповіді користувача, стилізуючи їх засобами каскадних таблиць стилю CSS. Натискання кнопки «Заново» дозволяє пройти тестування необмежену кількість разів.

Таблиця 4.4 – Тестові випадки

Тестовий випадок	Результат виконання
Натискання на елемент кнопка «Далі»	Нічого не відбувається
Натискання на елемент варіанту відповіді	Даний варіант позначається обраним
Натискання на інший елемент варіанту відповіді	Даний варіант позначається обраним замість попереднього
Натискання на елемент кнопка «Далі», після обрання варіанту відповіді	Перехід до наступного питання, індикатор питання інкрементується, обраний варіант відповіді зберігається в поточних результатах
Натискання на елемент кнопка «Далі», після обрання варіанту відповіді для останнього питання	Перехід до результатів тестування, обраний варіант відповіді зберігається в поточних результатах
Натискання на елемент кнопка «Заново»	Очищення масиву поточних результатів, перехід до початку тестування
Вибір правильного варіанту відповіді	При виведені результатів, система стилізує дану відповідь як правильну
Вибір неправильного варіанту відповіді	При виведені результатів, система стилізує дану відповідь як неправильну, та позначає вірний варіант

Створена web-система містить приклади практичного застосування (рис. 4.16) розглянутих технологій розробки, для більшої ефективності процесу навчання користувача.

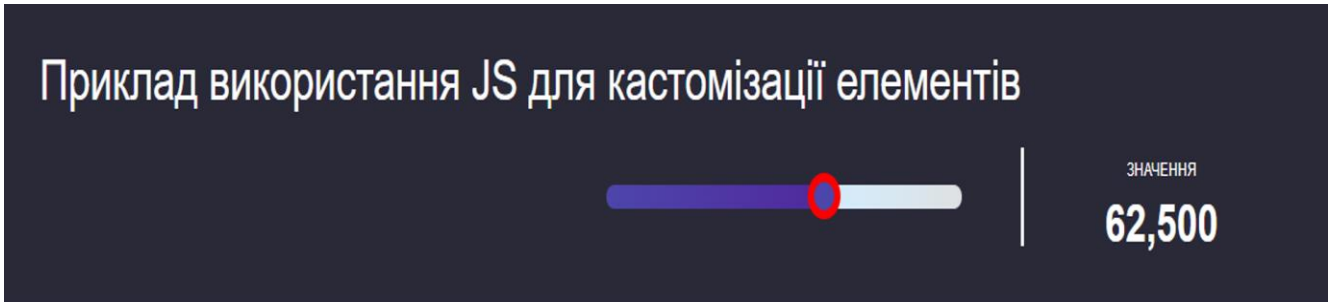

Рисунок 4.16 – Приклад практичного застосування мови програмування JavaScript

Таблиця 4.5 – Тестові випадки

Тестовий випадок	Результат виконання
Натискання на відповідні елементи, стилізовані засобами каскадних таблиць стилів	Відображення заданої анімації
Наведення курсору на відповідні елементи, стилізовані засобами каскадних таблиць стилів	Відображення заданої анімації
Зміщення повзунка елементу діапазону	Відповідна зміна числового еквіваленту
Натискання на елементи для перемикання зображень у слайдері	Перехід до відповідного зображення
Наведення курсору на відповідні елементи, стилізовані засобами мови програмування JavaScript	Відображення заданої анімації

Також було перевірено створені прикладні елементи на коректність відображення та виконання заданого функціоналу, на мобільних пристроях. Анімації що відображаються при наведені курсору, на пристроях з тачскріном викликаються натисканням на відповідний елемент.

4.3 Розробка інструкції користувача

Для доступу до створеної web-системи користувач потребує лише підключення до мережі інтернет та встановлений на пристрої браузер.

Різні браузери мають свої особливості відображення елементів web-сторінок, проте засобами каскадних таблиць стилів CSS дані особливості було усунено, тому розроблена web-система однаково відображається в усіх популярних браузерах.

У процесі розробки навчальної системи було використано HTML5 теги, що не підтримуються застарілими версіями браузерів. Для коректного відображення web-сторінок користувачу необхідний один із варіантів браузерів представлених у таблиці 4.6.

Таблиця 4.6 – Підтримка браузерами

Браузер	Підтримка
Internet Explorer	Версія 9.0+
Google Chrome	Версія 8.0+
Opera	Версія 9.2+
Safari	Версія 5.0+
Firefox	Версія 4.0+

Також елементи web-сторінок системи можуть не коректно відображатися на мобільних пристроях з застарілою версією операційної системи, для повної підтримки необхідно Android версії 2.1+, або ж iOS версії 3.0+.

При вході до web-системи, користувач потрапляє на головну сторінку, де йому необхідно пройти процес реєстрації або авторизації, для доступу до навчальних матеріалів.

Користувач може почати вивчення web-розробки згідно заданої послідовності, або самостійно обрати необхідну тему користуючись навігацією по системі.

4.4 Висновки

У четвертому розділі було проведено аналіз видів тестування web-систем, а також проведено тестування створеної web-системи для вивчення web-програмування. Було перевірено систему навігації, проведено валідацію полів введення інформації блоку реєстрації, та перевірено систему тестування користувача. Перевірено адаптивність системи. Також було розроблено інструкцію користувача.

ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи було створено web-систему для вивчення web-програмування. Розроблена система призначена для підвищення ефективності процесу вивчення web-програмування, шляхом об’єднання теоретичної інформації, практичних прикладів та тестування з базових технологій web-розробки в межах одного інтернет-ресурсу.

У бакалаврській дипломній роботі було проаналізовано стан навчальних web-систем для вивчення web-програмування на сьогоднішній день. Проведено порівняльний аналіз аналогів, на основі якого було доведено актуальність створення власної навчальної системи. Визначено основні задачі для виконання.

Виконуючи бакалаврську дипломну роботу було проведено аналіз навчального аспекту web-системи та підходи до подання матеріалу. Також було розроблено зрозумілу для нових користувачів структуру інтерфейсу web-системи.

Проведено розробку методу тестування для користувача, а також розробку методу і моделі загальної роботи web-системи. Проаналізовано та обґрунтовано вибір технологій для реалізації програмної компоненти web-системи, мови програмування PHP для забезпечення запису даних користувача до бази даних при реєстрації, сервісу PhpMyAdmin для адміністрування бази даних, а також вибору редактора Sublime Text як середовища для розробки та програми-збирача Prepros для компіляції препроцесора SASS.

У процесі виконання бакалаврської дипломної роботи було розроблено:

- програмні засоби для впровадження адаптивної верстки;
- програмні засоби для навігації у web-системі;
- блок реєстрації та авторизації у системі;
- блок тестування користувача.

Тестування web-системи довело її працездатність та відповідність поставленому технічному завданню. Також було створено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jennifer Campbell Web Design: Introductory / J. Campbell – Cengage Learning 2017. – 27 ст.
2. Бевз С. В. Розробка навчальної системи для вивчення веб-програмування / С. В. Бевз, Г. Б. Ракитянська, В.В. Войтко, С. М. Бурбело, Ю.С.Мазуренко Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022", Секція - Інформаційні технології та комп'ютерна інженерія. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/viewFile/16202/13644.
3. Website [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Website.
4. W3schools [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/.
5. TeamtreeHouse [Електронний ресурс] – Режим доступу до ресурсу: https://teamtreehouse.com/.
6. Codecademy [Електронний ресурс] – Режим доступу до ресурсу: https://www.codecademy.com/learn/learn-html.
7. Css-tricks [Електронний ресурс] – Режим доступу до ресурсу: https://css-tricks.com/.
8. Методологія БЕМ в дії [Електронний ресурс] – Режим доступу до ресурсу: https://avivi.pro/ua/blog/metodologiya-bem-v-deystvii/.
9. Dave Raggett Getting started with HTML / D. Raggett – W3C, may 2005.
10. CSS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS.
11. SASS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Sass.

12. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript.
13. AcademyUA [Електронний ресурс] – Режим доступу до ресурсу: https://academyua.com/ua/stati/78-vchimosya-programuvati-6-porad-novachkam.
14. Адаптивна верстка сайту [Електронний ресурс] – Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/adaptyvna-verstka-sajtu/.
15. Linda Heaton Unified Modeling Language Superstructure Specification, version 2.1.1. Document formal/2007-02-05. / L. Heaton – Object Management Group, February 2007. – 295 ст.
16. PHP [Електронний ресурс] – Режим доступу до ресурсу: http://programming.in.ua/web-design/allphp/30-about-php.html.
17. PhpMyAdmin [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/PhpMyAdmin.
18. BUSINESS SITE [Електронний ресурс] – Режим доступу до ресурсу: https://www.site2b.ua/web-blog/mobile-first-website-design.html.
19. Mobile First Design [Електронний ресурс] – Режим доступу до ресурсу: https://esputnik.com/uk/blog/mobile-first-design-pry-stvorenni-saitiv-shcho-vin-oznachaie-i-chomu-ie-nastilky-aktualnym.
20. Sam Lightstone Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more / Lightstone, S. Teorey T. Nadeau T. – Morgan Kaufmann Press 2007. – 7 ст.
21. Функція md5 [Електронний ресурс] – Режим доступу до ресурсу: https://php5.kiev.ua/php7/function.md5.html.
22. Види тестування сайтів [Електронний ресурс] – Режим доступу до ресурсу: https://webtune.com.ua/statti/web-rozrobka/vydy-testuvannya-sajtiv/.

23. Панель інструментів розробника [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/developer-tools-in-the-browser-console-tab/.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"_25_" _березня_2022 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка веб-системи для вивчення основ web-програмування»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доц. Г.Б. Ракитянська

"____"_____2022 р.

Виконав:

_____студент гр. 3ПІ-18б Ю.С. Мазуренко

"____"_____2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «розробка веб-системи для вивчення основ web-програмування».

Галузь застосування – веб-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 66 від « 24 » березня 2022 р.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності вивчення користувачем web-програмування шляхом об’єднання теоретичної інформації, практичних прикладів та тестування в межах одного інтернет-ресурсу, що дозволить забезпечити базові потреби користувача в рамках навчального процесу.

Призначення роботи – розробка та програмна реалізація навчальної web-системи для вивчення основ web-програмування.

4. Вихідні дані для проведення НДР

Бевз С. В. Розробка навчальної системи для вивчення веб-програмування / С. В. Бевз, Г. Б. Ракитянська, В.В. Войтко, С. М. Бурбело, Ю.С.Мазуренко Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022", Секція - Інформаційні технології та комп’ютерна інженерія. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/viewFile/16202/13644.

5. Технічні вимоги

Вхідні дані – інформація користувача, кількість вакансій; вихідні дані – графічний інтерфейс зі списком доступних вакансій.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка веб-системи для вивчення основ web-програмування»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ракитянська Г. Б.

Оригінальність	98,2%
Схожість	1,8%

Аналіз звіту подібності

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

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

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

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Мазуренко. Ю. С.

Керівник роботи _____ Ракитянська Г. Б.

Додаток В – Лістинг програми

Index.js

```
"use strict"

const isMobile = {
	Android: function () {
		return navigator.userAgent.match(/Android/i);
	},
	BlackBerry: function () {
		return navigator.userAgent.match(/BlackBerry/i);
	},
	iOS: function () {
		return navigator.userAgent.match(/iPhone|iPad|iPod/i);
	},
	Opera: function () {
		return navigator.userAgent.match(/Opera Mini/i);
	},
	Windows: function () {
		return navigator.userAgent.match(/IEMobile/i);
	},
	any: function () {
		return (
			isMobile.Android() ||
			isMobile.BlackBerry() ||
			isMobile.iOS() ||
			isMobile.Opera() ||
			isMobile.Windows());
	}
};

if (isMobile.any()) {
	document.body.classList.add('_touch');

	let menuArrows = document.querySelectorAll('.menu__arrow');
```

```
        if (menuArrows.length > 0){
                for (let index = 0; index < menuArrows.length; index++){
                        const menuArrow = menuArrows[index];
                        menuArrow.addEventListener("click", function(e){
                                menuArrow.parentElement.classList.toggle('_active');
                        });
                }
        }
} else{
        document.body.classList.add('_pc');
}

//Меню бургер

const iconMenu = document.querySelector('.menu__icon');
const menuBody = document.querySelector('.menu__body');
if(iconMenu){
        iconMenu.addEventListener("click", function(e){
                document.body.classList.toggle('_lock');
                iconMenu.classList.toggle('_active');
                menuBody.classList.toggle('_active');
        });
}

//Прокрутка при натисканні

const menuLinks = document.querySelectorAll('.menu__link[data-goto]');
if (menuLinks.length > 0){
        menuLinks.forEach(menuLink => {
                menuLink.addEventListener("click", onMenuLinkClick);
        });

        function onMenuLinkClick(e) {
                const menuLink = e.target;
```

```
                if(menuLink.dataset.goto && document.querySelector(menuLink.dataset.goto)){
                    const gotoBlock = document.querySelector(menuLink.dataset.goto);
                    const gotoBlockValue = gotoBlock.getBoundingClientRect().top + pageYOffset - document.querySelector('header').offsetHeight;

                    if (iconMenu.classList.contains('_active')){
                        document.body.classList.remove('_lock');
                        iconMenu.classList.remove('_active');
                        menuBody.classList.remove('_active');
                    }

                    window.scrollTo({
                        top: gotoBlockValue,
                        behavior: "smooth"
                    });
                    e.preventDefault();
                }
            }
        }

        //Прокрутка при натисканні

        const pageLinks = document.querySelectorAll('.page__link[data-goto]');
        if (pageLinks.length > 0){
            pageLinks.forEach(pageLink => {
                pageLink.addEventListener("click", onPageLinkClick);
            });

            function onPageLinkClick(e) {
                const pageLink = e.target;
                if(pageLink.dataset.goto && document.querySelector(pageLink.dataset.goto)){
                    const gotoBlock = document.querySelector(pageLink.dataset.goto);
```

```
                const gotoBlockValue = gotoBlock.getBoundingClientRect().top + pageYOffset - document.querySelector('header').offsetHeight;

                /*if (iconMenu.classList.contains('_active')){
                    document.body.classList.remove('_lock');
                    iconMenu.classList.remove('_active');
                    menuBody.classList.remove('_active');
                }*/

                window.scrollTo({
                    top: gotoBlockValue,
                    behavior: "smooth"
                });
                e.preventDefault();
            }
        }
    }
```

Quiz.js

```
const DATA = [
    {
        question: 'Приклад питання',
        answers: [
            {
                id: '1',
                value: 'приклад відповіді',
                correct: false,
            },
            {
                id: '2',
                value: 'приклад відповіді,
                correct: true,
            },
        ]
```

```
        },
];

let localResults = {};

const quiz = document.getElementById('quiz');
const questions = document.getElementById('questions');
const indicator = document.getElementById('indicator');
const results = document.getElementById('results');
const btnNext = document.getElementById('btn-next');
const btnRestart = document.getElementById('btn-restart');

const renderQuestions = (index) => {
        renderIndicator(index + 1);

        questions.dataset.currentStep = index;

        const renderAnswers = () => DATA[index].answers
                .map((answer) => `
                        <li>
        <label>
                <input class="answer-input" type="radio" name=${index} value=${answer.id}>
                ${answer.value}
        </label>
       </li>
                        `)
                .join('');

        questions.innerHTML = `
                <div class="quiz-questions-item">
      <div class="quiz-questions-item__question">${DATA[index].question}</div>
      <ul class="quiz-questions-item__answers">${renderAnswers()}</ul>
    </div>
        `;
```

```
};

const renderResults = () => {
    let content = '';

    const getClassname = (answer, questionIndex) => {
        let classname = '';

        if (!answer.correct && answer.id === localResults[questionIndex]) {
            classname = 'answer--invalid';
        } else if (answer.correct && answer.id === localResults[questionIndex]) {
            classname = 'answer--valid';
        } else if (answer.correct ) {
            classname = 'answer--invalid--correct';
        }

        return classname;
    };

    const getAnswers = (questionIndex) => DATA[questionIndex].answers
        .map((answer) => `<li class=${getClassname(answer, questionIndex)}>${answer.value}</li>`)
        .join('');

    DATA.forEach((question, index) => {
        content += `
            <div class="quiz-results-item">
        <div class="quiz-results-item__question">${question.question}</div>
            <ul class="quiz-results-item__answers">${getAnswers(index)}</ul>
        </div>
        `;
    });

    results.innerHTML = content;
```

```
};

const renderIndicator = (currentStep) => {
	indicator.innerHTML = `${currentStep}/${DATA.length}`;
};

quiz.addEventListener('change', (event) =>{
	//логіка відповіді
	if(event.target.classList.contains('answer-input')) {
		localResults[event.target.name] = event.target.value;
		btnNext.disabled = false;
	}
});

quiz.addEventListener('click', (event) =>{

	if(event.target.classList.contains('btn-next')) {
		const nextQuestionIndex = Number(questions.dataset.currentStep) + 1;

		if(DATA.length === nextQuestionIndex) {
			//перехід до результатів
			questions.classList.add('questions--hidden');
			indicator.classList.add('indicator--hidden');
			results.classList.add('indicator--visible');
			btnNext.classList.add('btn-next--hidden');
			btnRestart.classList.add('btn-restart--visible');
			renderResults();
		} else {
			//перехід до наступного питання
			renderQuestions(nextQuestionIndex);
		}

		btnNext.disabled = true;
	}
```

```
        if(event.target.classList.contains('btn-restart')) {
                localResults = {};
                results.innerHTML = '';

                questions.classList.remove('questions--hidden');
                indicator.classList.remove('indicator--hidden');
                results.classList.remove('indicator--visible');
                btnNext.classList.remove('btn-next--hidden');
                btnRestart.classList.remove('btn-restart--visible');

                renderQuestions(0);
        }
});

renderQuestions(0);
```

Theme.js

```
let changeThemeButtons = document.querySelectorAll('.changeTheme');

changeThemeButtons.forEach(button => {
   button.addEventListener('click', function () {
      let theme = this.dataset.theme;
      applyTheme(theme);
   });
});

function applyTheme(themeName) {
   document.querySelector('[title="theme"]').setAttribute('href',
`css/theme-${themeName}.css`);
   changeThemeButtons.forEach(button => {
      button.style.display = 'block';
   });
   document.querySelector(`[data-theme="${themeName}"]`).style.display = 'none';
```

```
    localStorage.setItem('theme', themeName);
}

let activeTheme = localStorage.getItem('theme');

if(activeTheme === null || activeTheme === 'light') {
    applyTheme('light');
} else if (activeTheme === 'dark') {
    applyTheme('dark');
}
```

Auth.php

```
<?php
        $login = filter_var(trim($_POST['login']),
        FILTER_SANITIZE_STRING);
        $pass = filter_var(trim($_POST['pass']),
        FILTER_SANITIZE_STRING);

        $pass = md5($pass."ckeuvichoistel");

        require "connect.php";

        $result = $mysql->query("SELECT * FROM `users` WHERE `login` = '$login' AND
`pass` = '$pass'");
        $user = $result->fetch_assoc();
        if(count($user) == 0) {
                echo "Користувача не знайдено";
                exit();
        }

        $mysql->close();

        header('Location: /');
?>
```

Check.php

```
<?php
	$login = filter_var(trim($_POST['login']),
	FILTER_SANITIZE_STRING);
	$name = filter_var(trim($_POST['name']),
	FILTER_SANITIZE_STRING);
	$pass = filter_var(trim($_POST['pass']),
	FILTER_SANITIZE_STRING);

	if(mb_strlen($login) < 5 || mb_strlen($login) > 90) {
		echo "Недопустима довжина логіну";
		exit();
	} else if(mb_strlen($name) < 1 || mb_strlen($login) > 50) {
		echo "Недопустима довжина імені";
		exit();
	} else if(mb_strlen($pass) < 4 || mb_strlen($login) > 8) {
		echo "Недопустима довжина пароля (2-6 символів)";
		exit();
	}

	$pass = md5($pass."ckeuvichoistel");

	require "connect.php";
	$mysql->query("INSERT INTO `users` (`login`, `pass`, `name`)
	VALUES('$login', '$pass', '$name')");

	$mysql->close();

	header('Location: /');
?>
```

Connect.php

```
<?php
```

```
        $mysql = new mysqli('localhost', 'root', 'root', 'reg-bd');
?>
```

Sicons.js

```
const icons = document.querySelectorAll('.icon');
icons.forEach (icon => {
  icon.addEventListener('click', (event) => {
    icon.classList.toggle("open");
  });
});
```

Sslider.js

```
const swiper = new Swiper('.swiper', {
      autoplay: {
        delay: 3000,
        disableOnInteraction: false,
      },
  loop: true,

  pagination: {
    el: '.swiper-pagination',
    clickable: true,
  },

  navigation: {
    nextEl: '.swiper-button-next',
    prevEl: '.swiper-button-prev',
  },
});
```

Srange.js

```
class Slider {
  constructor (rangeElement, valueElement, options) {
```

```
    this.rangeElement = rangeElement
    this.valueElement = valueElement
    this.options = options

    this.rangeElement.addEventListener('input', this.updateSlider.bind(this))
  }

  init() {
    this.rangeElement.setAttribute('min', options.min)
    this.rangeElement.setAttribute('max', options.max)
    this.rangeElement.value = options.cur

    this.updateSlider()
  }

  asVal(value) {
    return parseFloat(value)
      .toLocaleString('en-US', { maximumFractionDigits: 2 })
  }

  generateBackground(rangeElement) {
    if (this.rangeElement.value === this.options.min) {
      return
    }

    let percentage =  (this.rangeElement.value - this.options.min) / (this.options.max - this.options.min) * 100
    return 'background: linear-gradient(to right, #4D46AB, #50299c ' + percentage + '%, #d3edff ' + percentage + '%, #dee1e2 100%)'
  }

  updateSlider (newValue) {
    this.valueElement.innerHTML = this.asVal(this.rangeElement.value)
    this.rangeElement.style = this.generateBackground(this.rangeElement.value)
```

```
  }
}

let rangeElement = document.querySelector('.range [type="range"]')
let valueElement = document.querySelector('.range .range__value span')

let options = {
  min: 0,
  max: 100000,
  cur: 40000
}

if (rangeElement) {
  let slider = new Slider(rangeElement, valueElement, options)

  slider.init()
}
```

Sbutton.js

```
const LiquidButton = class LiquidButton {
  constructor(svg) {
    const options = svg.dataset;
    this.id = this.constructor.id || (this.constructor.id = 1);
    this.constructor.id++;
    this.xmlns = 'http://www.w3.org/2000/svg';
    this.tension = 0.4;
    this.width   = 200;
    this.height  = 50;
    this.margin  = 40;
    this.hoverFactor = -0.1;
    this.gap     = 5;
    this.debug   = options.debug;
    this.forceFactor = options.forceFactor;
    this.color1 = 'red';
```

```
        this.color2 = '#4D46AB';
        this.color3 = '#50299c';
        this.textColor = options.textColor || '#ffffff';
        this.text = options.text    || 'Button';
        this.svg = svg;
        this.layers = [{
          points: [],
          viscosity: 0.5,
          mouseForce: 100,
          forceLimit: 2,
        },{
          points: [],
          viscosity: 0.8,
          mouseForce: 150,
          forceLimit: 3,
        }];
        for (let layerIndex = 0; layerIndex < this.layers.length; layerIndex++) {
          const layer = this.layers[layerIndex];
          layer.viscosity = options['layer-' + (layerIndex + 1) + 'Viscosity'] * 1 || layer.viscosity;
          layer.mouseForce = options['layer-' + (layerIndex + 1) + 'MouseForce'] * 1 ||
layer.mouseForce;
          layer.forceLimit = options['layer-' + (layerIndex + 1) + 'ForceLimit'] * 1 || layer.forceLimit;
          layer.path = document.createElementNS(this.xmlns, 'path');
          this.svg.appendChild(layer.path);
        }
        this.wrapperElement = options.wrapperElement || document.body;
        if (!this.svg.parentElement) {
          this.wrapperElement.append(this.svg);
        }

        this.svgText = document.createElementNS(this.xmlns, 'text');
        this.svgText.setAttribute('x', '50%');
        this.svgText.setAttribute('y', '50%');
        this.svgText.setAttribute('dy', ~~(this.height / 8) + 'px');
```

```
  this.svgText.setAttribute('font-size', ~~(this.height / 3));
  this.svgText.style.fontFamily = 'sans-serif';
  this.svgText.setAttribute('text-anchor', 'middle');
  this.svgText.setAttribute('pointer-events', 'none');
  this.svg.appendChild(this.svgText);

  this.svgDefs = document.createElementNS(this.xmlns, 'defs')
  this.svg.appendChild(this.svgDefs);

  this.touches = [];
  this.noise = options.noise || 0;
  document.body.addEventListener('touchstart', this.touchHandler);
  document.body.addEventListener('touchmove', this.touchHandler);
  document.body.addEventListener('touchend', this.clearHandler);
  document.body.addEventListener('touchcancel', this.clearHandler);
  this.svg.addEventListener('mousemove', this.mouseHandler);
  this.svg.addEventListener('mouseout', this.clearHandler);
  this.initOrigins();
  this.animate();
}

get mouseHandler() {
  return (e) => {
    this.touches = [{
      x: e.offsetX,
      y: e.offsetY,
      force: 1,
    }];
  };
}

get touchHandler() {
  return (e) => {
    this.touches = [];
```

```
    const rect = this.svg.getBoundingClientRect();
    for (let touchIndex = 0; touchIndex < e.changedTouches.length; touchIndex++) {
      const touch = e.changedTouches[touchIndex];
      const x = touch.pageX - rect.left;
      const y = touch.pageY - rect.top;
      if (x > 0 && y > 0 && x < this.svgWidth && y < this.svgHeight) {
        this.touches.push({x, y, force: touch.force || 1});
      }
    }
    e.preventDefault();
  };
}
get clearHandler() {
  return (e) => {
    this.touches = [];
  };
}

get raf() {
  return this.__raf || (this.__raf = (
    window.requestAnimationFrame ||
    window.webkitRequestAnimationFrame ||
    window.mozRequestAnimationFrame ||
    function(callback){ setTimeout(callback, 10)}
  ).bind(window));
}

distance(p1, p2) {
  return Math.sqrt(Math.pow(p1.x - p2.x, 2) + Math.pow(p1.y - p2.y, 2));
}

update() {
  for (let layerIndex = 0; layerIndex < this.layers.length; layerIndex++) {
    const layer = this.layers[layerIndex];
```

```
        const points = layer.points;
        for (let pointIndex = 0; pointIndex < points.length; pointIndex++) {
          const point = points[pointIndex];
          const dx = point.ox - point.x + (Math.random() - 0.5) * this.noise;
          const dy = point.oy - point.y + (Math.random() - 0.5) * this.noise;
          const d = Math.sqrt(dx * dx + dy * dy);
          const f = d * this.forceFactor;
          point.vx += f * ((dx / d) || 0);
          point.vy += f * ((dy / d) || 0);
          for (let touchIndex = 0; touchIndex < this.touches.length; touchIndex++) {
            const touch = this.touches[touchIndex];
            let mouseForce = layer.mouseForce;
            if (
              touch.x > this.margin &&
              touch.x < this.margin + this.width &&
              touch.y > this.margin &&
              touch.y < this.margin + this.height
            ) {
              mouseForce *= -this.hoverFactor;
            }
            const mx = point.x - touch.x;
            const my = point.y - touch.y;
            const md = Math.sqrt(mx * mx + my * my);
            const mf = Math.max(-layer.forceLimit, Math.min(layer.forceLimit, (mouseForce * touch.force) / md));
            point.vx += mf * ((mx / md) || 0);
            point.vy += mf * ((my / md) || 0);
          }
          point.vx *= layer.viscosity;
          point.vy *= layer.viscosity;
          point.x += point.vx;
          point.y += point.vy;
        }
        for (let pointIndex = 0; pointIndex < points.length; pointIndex++) {
```

```
      const prev = points[(pointIndex + points.length - 1) % points.length];
      const point = points[pointIndex];
      const next = points[(pointIndex + points.length + 1) % points.length];
      const dPrev = this.distance(point, prev);
      const dNext = this.distance(point, next);

      const line = {
        x: next.x - prev.x,
        y: next.y - prev.y,
      };
      const dLine = Math.sqrt(line.x * line.x + line.y * line.y);

      point.cPrev = {
        x: point.x - (line.x / dLine) * dPrev * this.tension,
        y: point.y - (line.y / dLine) * dPrev * this.tension,
      };
      point.cNext = {
        x: point.x + (line.x / dLine) * dNext * this.tension,
        y: point.y + (line.y / dLine) * dNext * this.tension,
      };
    }
  }
}

animate() {
  this.raf(() => {
    this.update();
    this.draw();
    this.animate();
  });
}

get svgWidth() {
  return this.width + this.margin * 2;
```

```
}

get svgHeight() {
  return this.height + this.margin * 2;
}

draw() {
  for (let layerIndex = 0; layerIndex < this.layers.length; layerIndex++) {
    const layer = this.layers[layerIndex];
    if (layerIndex === 1) {
      if (this.touches.length > 0) {
        while (this.svgDefs.firstChild) {
          this.svgDefs.removeChild(this.svgDefs.firstChild);
        }
        for (let touchIndex = 0; touchIndex < this.touches.length; touchIndex++) {
          const touch = this.touches[touchIndex];
          const gradient = document.createElementNS(this.xmlns, 'radialGradient');
          gradient.id = 'liquid-gradient-' + this.id + '-' + touchIndex;
          const start = document.createElementNS(this.xmlns, 'stop');
          start.setAttribute('stop-color', this.color3);
          start.setAttribute('offset', '0%');
          const stop = document.createElementNS(this.xmlns, 'stop');
          stop.setAttribute('stop-color', this.color2);
          stop.setAttribute('offset', '100%');
          gradient.appendChild(start);
          gradient.appendChild(stop);
          this.svgDefs.appendChild(gradient);
          gradient.setAttribute('cx', touch.x / this.svgWidth);
          gradient.setAttribute('cy', touch.y / this.svgHeight);
          gradient.setAttribute('r', touch.force);
          layer.path.style.fill = 'url(#' + gradient.id + ')';
        }
      } else {
        layer.path.style.fill = this.color2;
```

```
      }
    } else {
      layer.path.style.fill = this.color1;
    }
    const points = layer.points;
    const commands = [];
    commands.push('M', points[0].x, points[0].y);
    for (let pointIndex = 1; pointIndex < points.length; pointIndex += 1) {
      commands.push('C',
        points[(pointIndex + 0) % points.length].cNext.x,
        points[(pointIndex + 0) % points.length].cNext.y,
        points[(pointIndex + 1) % points.length].cPrev.x,
        points[(pointIndex + 1) % points.length].cPrev.y,
        points[(pointIndex + 1) % points.length].x,
        points[(pointIndex + 1) % points.length].y
      );
    }
    commands.push('Z');
    layer.path.setAttribute('d', commands.join(' '));
  }
  this.svgText.textContent = this.text;
  this.svgText.style.fill = this.textColor;
}

createPoint(x, y) {
  return {
    x: x,
    y: y,
    ox: x,
    oy: y,
    vx: 0,
    vy: 0,
  };
}
```

```
initOrigins() {
  this.svg.setAttribute('width', this.svgWidth);
  this.svg.setAttribute('height', this.svgHeight);
  for (let layerIndex = 0; layerIndex < this.layers.length; layerIndex++) {
    const layer = this.layers[layerIndex];
    const points = [];
    for (let x = ~~(this.height / 2); x < this.width - ~~(this.height / 2); x += this.gap) {
      points.push(this.createPoint(
        x + this.margin,
        this.margin
      ));
    }
    for (let alpha = ~~(this.height * 1.25); alpha >= 0; alpha -= this.gap) {
      const angle = (Math.PI / ~~(this.height * 1.25)) * alpha;
      points.push(this.createPoint(
        Math.sin(angle) * this.height / 2 + this.margin + this.width - this.height / 2,
        Math.cos(angle) * this.height / 2 + this.margin + this.height / 2
      ));
    }
    for (let x = this.width - ~~(this.height / 2) - 1; x >= ~~(this.height / 2); x -= this.gap) {
      points.push(this.createPoint(
        x + this.margin,
        this.margin + this.height
      ));
    }
    for (let alpha = 0; alpha <= ~~(this.height * 1.25); alpha += this.gap) {
      const angle = (Math.PI / ~~(this.height * 1.25)) * alpha;
      points.push(this.createPoint(
        (this.height - Math.sin(angle) * this.height / 2) + this.margin - this.height / 2,
        Math.cos(angle) * this.height / 2 + this.margin + this.height / 2
      ));
    }
    layer.points = points;
```

```
    }
  }
}
const redraw = () => {
  button.initOrigins();
};

const buttons = document.getElementsByClassName('liquid-button');
for (let buttonIndex = 0; buttonIndex < buttons.length; buttonIndex++) {
  const button = buttons[buttonIndex];
  button.liquidButton = new LiquidButton(button);
}
```

Додаток Г – Графічна частина

ГРАФІЧНА ЧАСТИНА
РОЗРОБКА ВЕБ-СИСТЕМИ ДЛЯ ВИВЧЕННЯ ОСНОВ ВЕБ-ПРОГРАМУВАННЯ

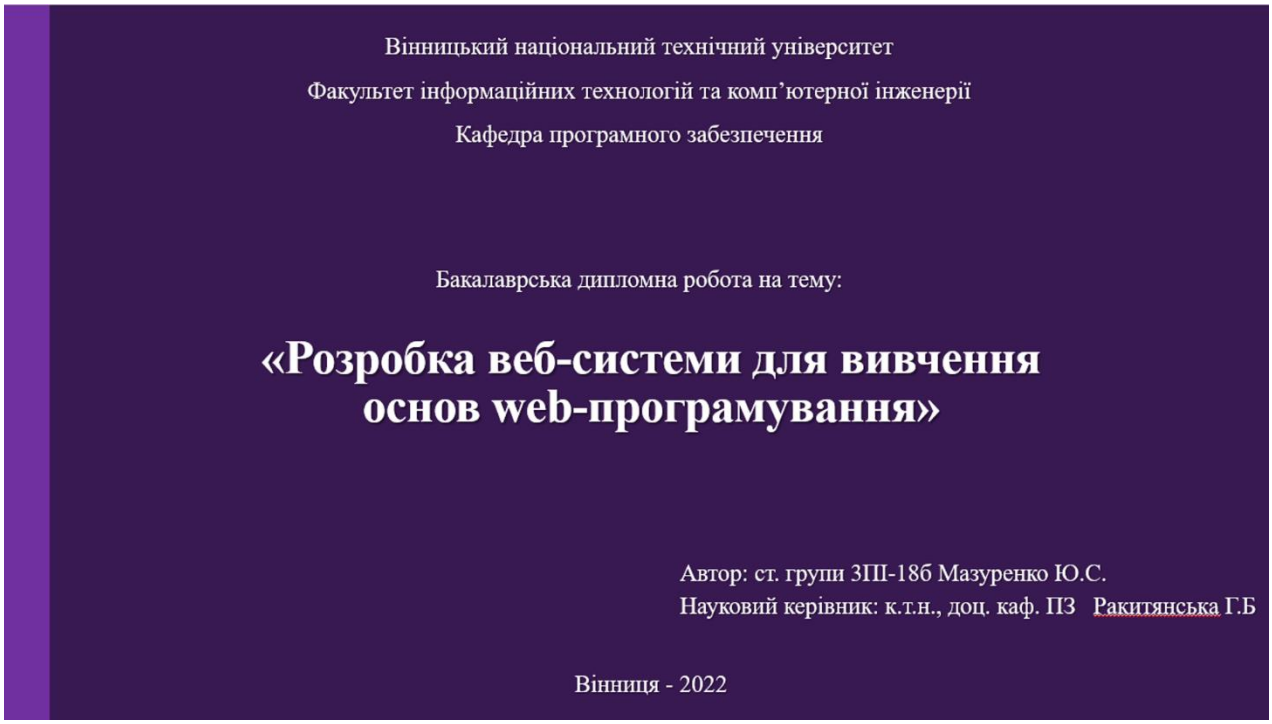

Рисунок Г.1 – Титульний слайд

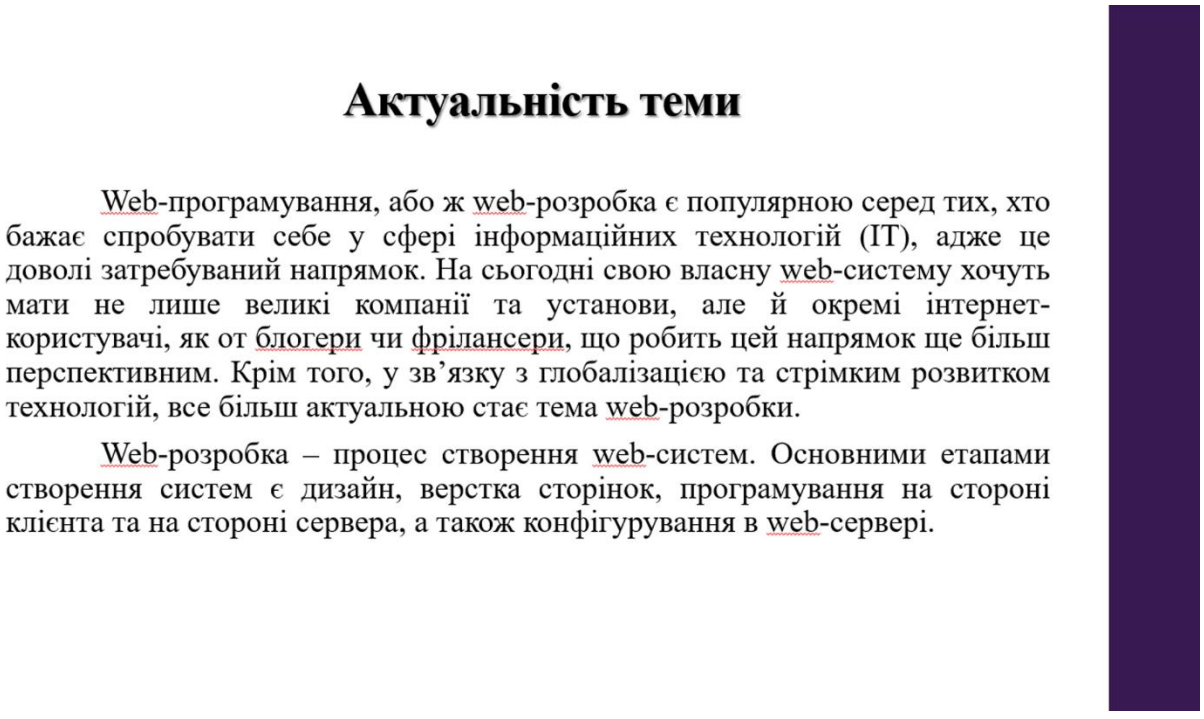

Рисунок Г.2 – Актуальність теми

Мета, об’єкт та предмет дослідження

- Метою бакалаврської дипломної роботи є підвищення ефективності вивчення користувачем web-програмування шляхом об’єднання теоретичної інформації, практичних прикладів та тестування в межах одного інтернет-ресурсу, що дозволить забезпечити базові потреби користувача в рамках навчального процесу.

- Об’єкт дослідження – процеси розробки програмних засобів для навчальних web-систем, призначених для вивчення web-програмування.

- Предмет дослідження – методи та засоби розробки програмних засобів навчальної web-системи для вивчення web-програмування.

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

Задачі дослідження

- розробити структуру створюваного web-додатку;
- розробити метод та модель роботи системи;
- налаштувати навігацію по web-додатку;
- створити блоки з теоретичним матеріалом;
- створити блоки з практичними прикладами для вивчення;
- розробити блоки «quiz» для тестування користувача;
- розробити блок для реєстрації користувача;
- розробити програмні компоненти web- додатку;
- провести тестування розробленої web-системи.

Рисунок Г.4 – Задачі дослідження

Новизна отриманих результатів

- Подальшого розвитку отримав метод створення навчальної web-системи, який, на відміну від існуючих, реалізує систему вбудованих логічних зв'язків між компонентами для забезпечення повної підтримки навчального процесу з поєднанням блоків теоретичного матеріалу, практичних завдань і тестів з ідентифікацією рівня засвоєння матеріалу, що підвищує ефективність процесу вивчення основ web-програмування користувачем.
- Подальшого розвитку отримала модель навчальної web-системи, яка, на відміну від існуючих, зосереджена на оптимізації навчального процесу шляхом реалізації комплексного функціоналу для вивчення основ web-програмування, що забезпечить реалізацію універсального підходу до розробки доступного навчального веб-ресурсу, спростить пошук потрібної інформації, забезпечить тренувальні потреби користувача й підвищить ефективність процесу навчання. Розроблено алгоритм тестування виключно засобами програмування на стороні клієнта, що дозволило підвищити швидкість роботи навчальної web-системи.

Рисунок Г.5 – Новизна отриманих результатів

Практична цінність отриманих результатів

Практична цінність одержаних результатів полягає у створенні навчальної web-системи для вивчення основ web-програмування, що дозволить опановувати популярний IT-напрямок непідготовленим користувачам.

Рисунок Г.6 – Практична цінність отриманих результатів

Порівняльний аналіз аналогів

Критерій	W3schools	TeamtreeHouse	Codecademy	Css-tricks	StudyWEB
Наявність матеріалу для вивчення усіх базових технологій web-програмування	1	0	0	0	1
Тестування для самоперевірки	0	1	1	0	1
Безкоштовність повного функціоналу	0	0	1	1	1
Адаптивність	1	1	1	1	1
Підсумковий результат	2	2	3	2	4

Рисунок Г.7 – Порівняльний аналіз аналогів

Блок-схема методу тестування для користувача

Рисунок Г.8 – Блок-схема методу тестування користувача

Модель роботи навчальної web-системи

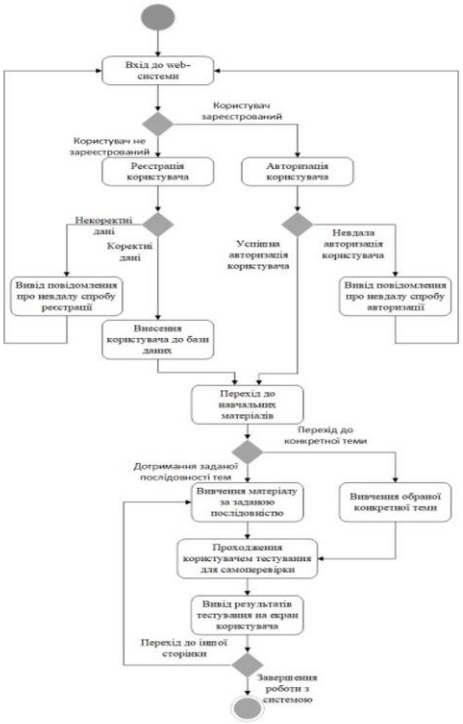

Рисунок Г.9 – Модель роботи навчальної web-системи

Загальний метод роботи web-системи

1. Авторизувавшись у системі, користувач отримує доступ до навчальних матеріалів. Блок реєстрації реалізовано засобами мови програмування PHP.

2. Після авторизації користувач переходить на сторінку Основи WEB (basic.html).

2.1. Ознайомившись із матеріалами заданої web-сторінки, користувач може перейти до наступної сторінки, користуючись посиланням на панелі навігації.

2.2. При переході за посиланням користувач буде поступово вивчати наданий матеріал за визначеним планом, однак, при наведені курсору на дане посилання відкривається випадаюче меню, у якому користувач може самостійно обрати необхідний йому матеріал.

Рисунок Г.10 – Загальний метод роботи web-системи

Загальний метод роботи web-системи

3. Інформацію на кожній web-сторінці розділено по блоках за тематикою.

3.1. Користувач буде поступово вивчати наданий матеріал за рекомендованим планом, переходячи від одного блоку до іншого. Блоки теоретичного матеріалу чергуються з прикладами практичного використання для підвищення ефективності навчального процесу. Приклади практичного використання реалізовано засобами Css та мови програмування JavaScript.

3.2. Кожна web-сторінка містить внутрішні посилання на блоки з найбільш важливою інформацією, тому користувач може самостійно обрати тему для вивчення Перехід за внутрішніми посиланнями реалізовано засобами мови програмування JavaScript, за допомогою методу .scrollTo() об'єкту window.

4. По завершенню процесу вивчення користувач може пройти тестування для самоперевірки у призначених для цього відповідних блоках. Блоки тестування реалізовано засобами мови програмування JavaScript та Css, для стилізації результатів.

Рисунок Г.11 – Загальний метод роботи web-системи (продовження)

Блок-схеми

Блок-схема загального алгоритму роботи web-системи

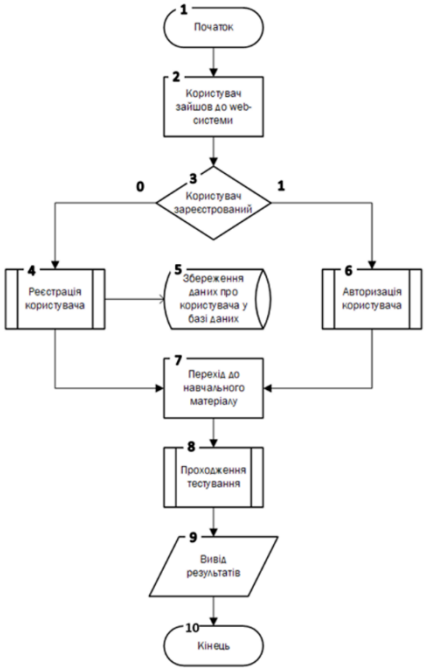

Рисунок Г.12 – Блок-схема загального алгоритму роботи web-системи

Блок-схема алгоритму тестування користувача

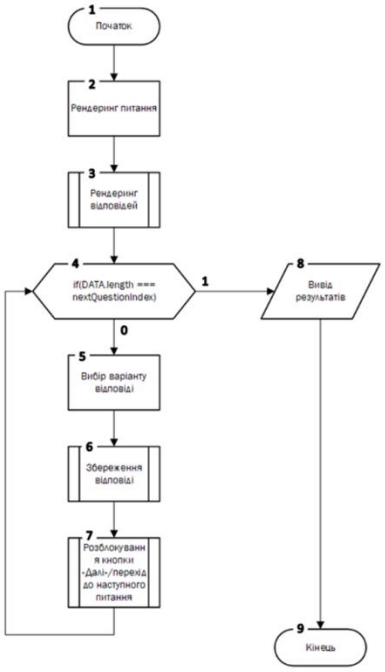

Рисунок Г.13 – Блок-схема алгоритму тестування користувача

Тестування web-системи

Головна сторінка web-системи

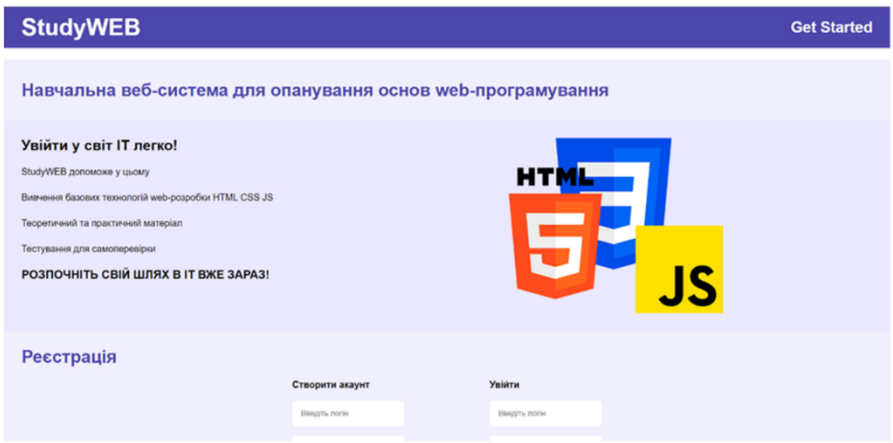

Рисунок Г.14 – Тестування web-системи: Головна сторінка

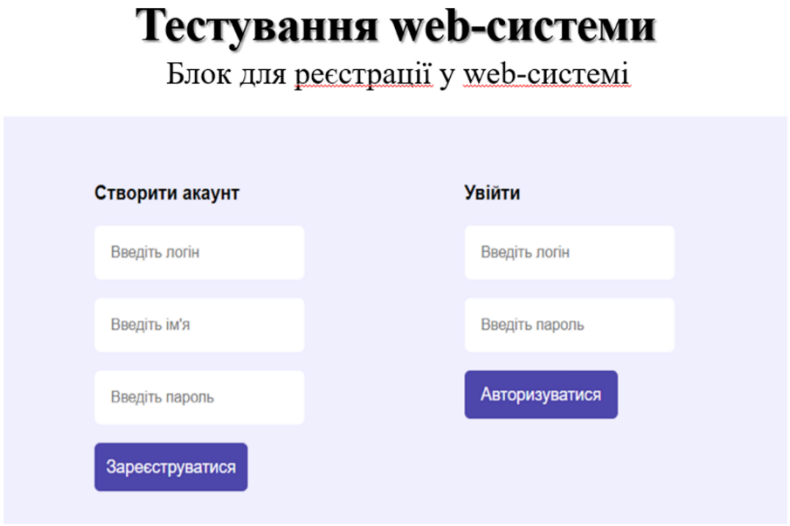

Рисунок Г.15 – Тестування web-системи: Блок реєстрації

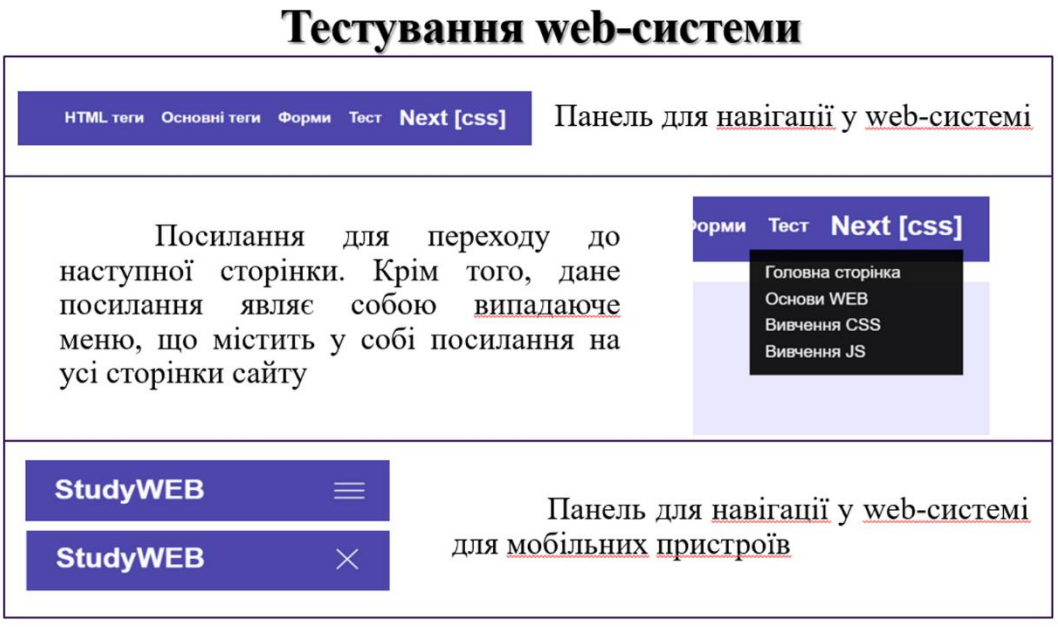

Рисунок Г.16 – Тестування web-системи: Навігація

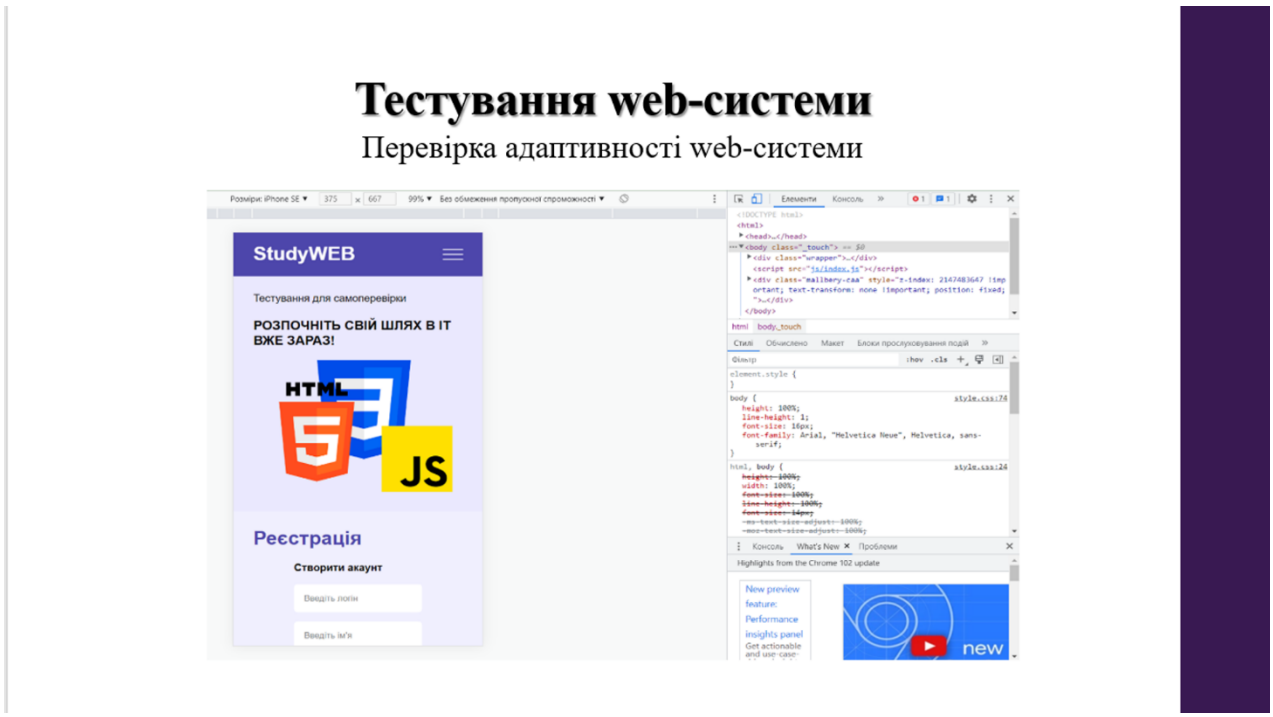

Рисунок Г.17 – Тестування web-системи: Перевірка адаптивності

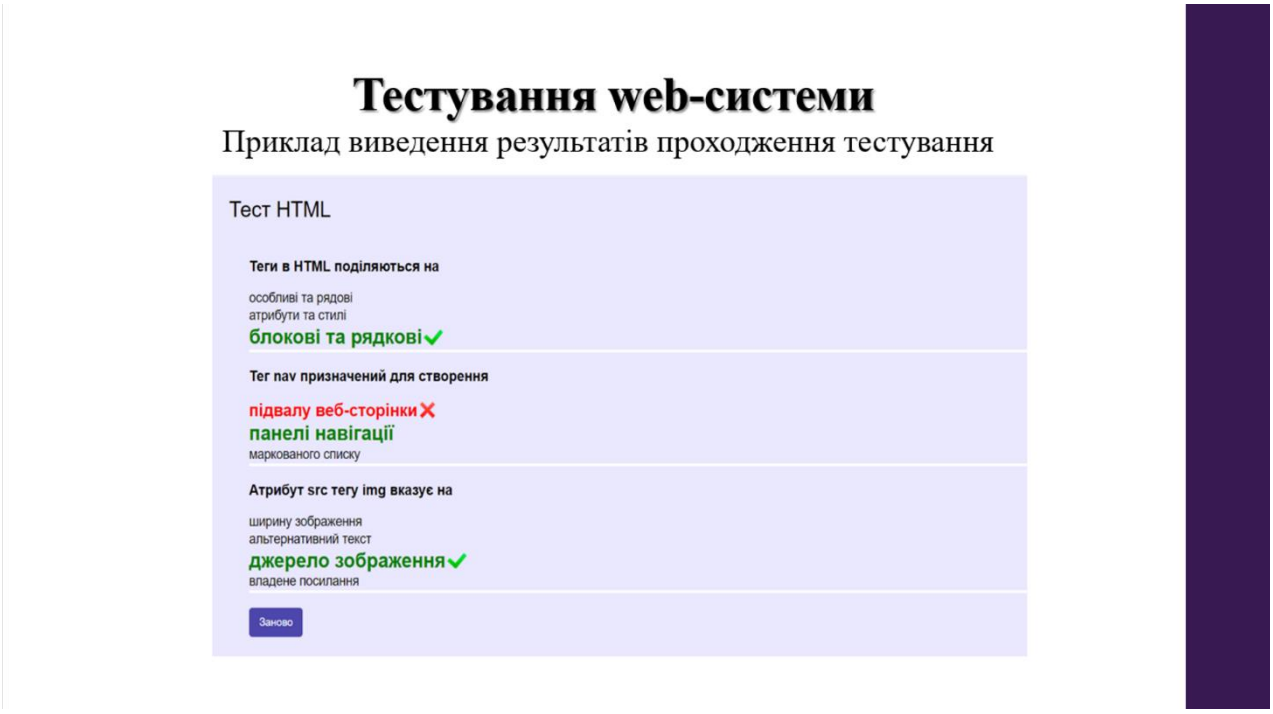

Рисунок Г.18 – Тестування web-системи: Вивід результатів тестування

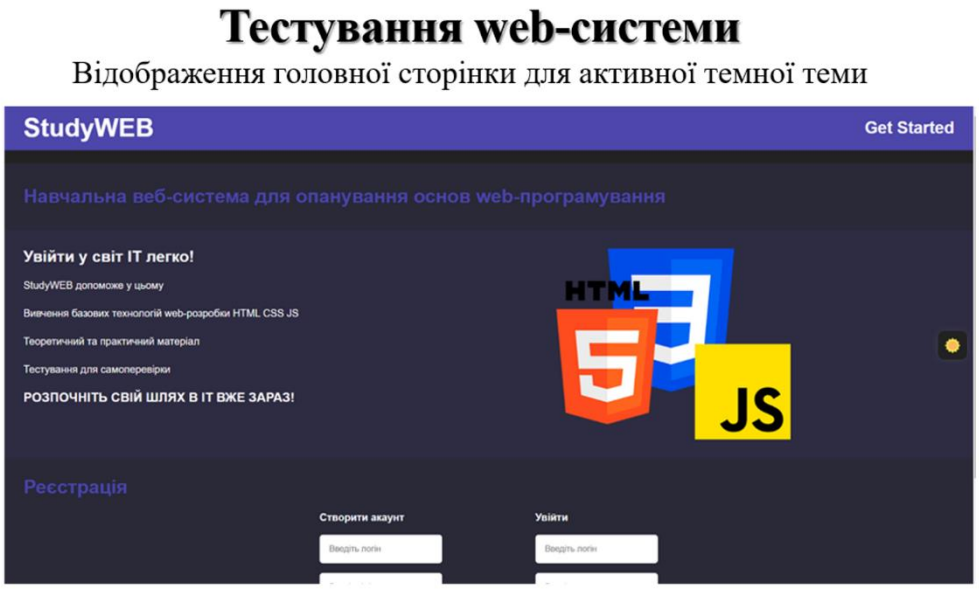

Рисунок Г.19 – Тестування web-системи: Перевірка зміни колірної теми

Рисунок Г.20 – Тестування web-системи: Практичні приклади

Рисунок Г.21 – Тестування web-системи: Практичні приклади

Апробація та публікації результатів роботи

Результати роботи доповідалися на Всеукраїнській науково-практичній інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022", Секція - Інформаційні технології та комп'ютерна інженерія.

Бевз С. В. Розробка навчальної системи для вивчення веб-програмування / С. В. Бевз, Г. Б. Ракитянська, В.В. Войтко, С. М. Бурбело, Ю.С.Мазуренко Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи - 2022", Секція - Інформаційні технології та комп'ютерна інженерія. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/viewFile/16202/13644

Рисунок Г.22 – Апробація та публікації результатів роботи

Висновки

У ході виконання бакалаврської дипломної роботи було створено web-систему для вивчення web-програмування. Розроблена система призначена для підвищення ефективності процесу вивчення web-програмування, шляхом об’єднання теоретичної інформації, практичних прикладів та тестування з базових технологій web-розробки в межах одного інтернет-ресурсу.

У процесі виконання бакалаврської дипломної роботи було розроблено:

- програмні засоби для впровадження адаптивної верстки;
- програмні засоби для навігації у web-системі;
- блок реєстрації та авторизації у системі;
- блок тестування користувача.

Тестування web-системи довело її працездатність та відповідність поставленому технічному завданню.

Рисунок Г.23 – Висновки