Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка веб-системи для вивчення мови програмування JavaScript»

Виконав: студент IV курсу
групи 2ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Давиденко Ілля Сергійович
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Бабюк Н.П.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН Крилик Л.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

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

Давиденку Іллі Сергійовичу

1. Тема роботи: Розробка веб-системи для вивчення мови програмування JavaScript.

Керівник роботи: к.т.н., доц. кафедри ПЗ Бабюк Н. П. 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 – Visual Studio Code, Мови розробки – HTHL, CSS, JavaScript, Операційна система – Windows 10.

4. Зміст розрахунково-пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка структури та алгоритмів програмного продукту; розробка модулів програмного продукту; тестування програми, висновки; перелік посилань; додатки; графічна частина.

5. Перелік графічного матеріалу: титульний слайд; мета, предмет і об'єкт дослідження; новизна та практична цінність; актуальність розробки; задачі дослідження; розробка структури інтерфейсу веб-системи; блок-схема алгоритму реєстрації та авторизації; використані технології; навчальна програма; перевірка завдань; тестування програми; апробація та публікація роботи.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–4	к.т.н Бабюк Н. П., доц. кафедри ПЗ		

7. Дата видачі завдання 25 березня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач дослідження		Виконано
2	Розробка структури та алгоритмів програмного продукту		Виконано
3	Розробка модулів програмного продукту		Виконано
4	Тестування програми		Виконано
5	Оформлення матеріалів до захисту БДР		Виконано

Студент _____ **Давиденко І.С.**
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ **Бабюк Н. П.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 94 сторінок формату А4, на яких є 35 рисунків, 2 таблиці, 4 додатки список використаних джерел містить 16 найменувань.

У бакалаврській дипломній роботі було проведено детальний аналіз сфери вивчення мови програмування JavaScript. Описано об’єкт, предмет, завдання та методи дослідження. Сформульовано мету дослідження – створення веб-системи для вивчення мови програмування JavaScript. Проаналізовано аналоги та зроблено висновок, що дана розробка є актуальною. Веб-система буде надавати користувачам знання для вивчення мови програмування JavaScript.

Було розроблено алгоритми та блок-схеми для критично важливих операцій, а також розроблено структуру веб-системи.

Веб-додаток було написано на мові програмування JavaScript та її платформі Node.js, а також мові розмітки HTML та каскадних таблицях стилів CSS. У якості середовища розробки було обрано Visual Studio Code. База даних була створена за допомогою технологій MySQL Workbench Кінцевий програмний продукт відповідає завданням та меті. У підсумку було отримано веб-додаток, який представляє собою веб-систему для вивчення мови програмування JavaScript.

Ключові слова: JavaScript, мови програмування, веб-система.

ANNOTATION

The bachelor's thesis consists of 94 A4 pages, which have 35 figures, 2 tables, 4 additions the list of sources used contains 16 items.

In the bachelor's thesis, a detailed analysis was performed of the field of JavaScript programming langage learning. The object, subject, tasks and methods of research are established. The aim of the study is creating a web system for the JavaScript programming language learning. The analysis of analogues is carried out and the actuality of development is proved. The system will provide user with knowledge needed to learn a JavaScript programming language.

Algorithms, block diagrams, have been developed for critical operations, as well as a web system structure.

The web system was written in JavaScript and its platform Node.js, markup language HTML and CSS. For the development environment Visual Studio Code was chosen. The database was created by using MySQL Workbench technologies. The final software product meets the objectives and goals. As a result, we received a web application, which is a web system for learning the JavaScript programming naguage.

Keywords: JavaScript, programming languages, web system.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сфера інформаційних технологій з кожним днем стає все актуальнішою і все більше людей бажають спробувати себе в ній, тому що як кажуть всюди, ви можете навчитися цьому в вільний час в себе вдома. Коли діло доходить до вивчення, початківці дізнаються, що інформації занадто багато, і часто просто не розуміють з чого почати.

Найскладнішим в початку вивчення програмування є вибір мови. Людині, яка зовсім не знайома з сферою інформаційних технологій дуже важко взнати, яку мову вивчити легше, яка найбільш актуальна або яка потрібна для того чим вона хоче займатися. І хоча всі мови потрібні по своєму, є мова яка буде найбільш легкою і потрібною для вивчення початківцям, і ця мова – JavaScript.

Основною сферою використання цієї мови є веб-розробка, але завдяки великої кількості створених бібліотек та фреймворків вона може використовуватися будь-де. В порівнянні з іншими мовами програмування, JavaScript виділяється, в першу чергу, своєю універсальністю. Більшість мов виконують лише свою роль, для якої вони були створені, а JavaScript може виконувати роль інших мов, хоч і не так досконально як вони. Завдяки тому, що він використовується в багатьох сферах, з ним можна буде працювати з тим, чим хоче займатися його користувач, а не вивчати для цього нову мову і саме тому його можна вважати найкращою мовою для початківців [1].

Велика частина програмістів стали ними завдяки вивчення програмування самотужки за допомогою доступних в інтернеті ресурсів. Саме по цій причині актуальність створення таких ресурсів є дуже високою в наш час.

Зазвичай на ресурсах для вивчання мов програмування доступна лише їх документація з поясненнями того як вони працюють, але без чіткого шляху як саме можна їх вивчити, а через це новачки можуть загубитися в інформації та можливо навіть залишити ідею навчання через думку, що це занадто складно.

Отже, розробка веб-системи для вивчення мови програмування JavaScript є актуальною, бо все більше людей бажають розпочати свій шлях в сфері інформаційних технологій, а ця мова є однією з найкращих для початківців.

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

Мета і завдання дослідження. Метою дослідження є підвищення продуктивності вивчення мови програмування JavaScript шляхом створення веб-системи для самостійного навчання.

Основними задачами дослідження є:

− аналіз стану галузі вивчення мови програмування JavaScript;

− порівняльний аналіз аналогів;

− аналіз методів розв’язання задачі;

− постановка задач;

− розробку структури і алгоритмів програмного продукту;

− варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу;

− розробка веб-системи для вивчення мови програмування JavaScript;

− тестування програми.

Об’єкт дослідження – процеси отримання знань користувачем під час користування веб-системою для вивчення мови програмування JavaScript.

Предмет дослідження – методи та засоби реалізації способів вивчення мови програмування JavaScript.

Методи дослідження. Протягом дослідження було використано методи розробки клієнтського інтерфейсу; методи навчання; методи створення бази даних; методи семантичної розмітки; методи задання стилів; методи оптимізації веб-системи та її клієнтського інтерфейсу; методи алгоритмізації модулів програмної системи.

Новизна отриманих результатів.

1. Було запропоновано збереження та відображення прогресу навчання, що відрізняє розроблюваний додаток від розглянутих аналогів, що дозволить користувачам відслідковувати свій прогрес в вивченні і тим самим прискорити його завдяки появі мотивації.

2. Було запропоновано покращену структуризацію контенту порівняно з досліджуваними ресурсами, що дозволить користувачам швидше знаходити потрібну їм інформацію та допоможе веб-сайту з'являтися на перших сторінках пошуку в пошукових системах.

Практична цінність одержаних результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень розроблено алгоритми та програмні засоби веб-системи для вивчення мови програмування JavaScript.

Особистий внесок. Усі наукові результати отримано здобувачем самостійно.

Апробація роботи матеріали бакалаврської кваліфікаційної роботи доповідались на науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії “LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії” (2022 р., м. Вінниця).

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференцій.

Технічне завдання наведено в додатку А.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану галузі вивчення мови програмування JavaScript

Завдяки стрімкому розвитку IT-сфери на сьогоднішній день існує чимало сервісів для вивчення мов програмування для різних рівнів підготовки. Аби отримати нові знання, потрібно лише мати час та бажання. Але з розвитком постають і нові проблеми: неспроможність обрати саме ту програму, яка потрібна; багато різної інформації, але далеко не вся є достовірною та якісною; чимало вузькоспеціальної лексики, яку новачкам доводиться шукати окремо в мережі Інтернет.

Проблема розробки програми під себе, незнання того, що за чим потрібно вчити є доволі поширеною, адже мережа Інтернет – це величезна купа корисної та навпаки непотрібної і шкідливої інформації. Через це багато людей заробляють на таких новачках, розробляючи програму навчання, даючи вже підготовлену інформацію та просячи за те саме, що є у вільному доступі, гроші.

Головним завданням людини, яка бажає почати свій шлях в сфері інформаційних технологій є вибір мови програмування. Від цього вибору може залежати чи сподобається людині ця галузь і чи продовжить вона свій в ній шлях. Однією з найкращих мов для початківців є мова JavaScript. JavaScript є універсальною мовою програмування. Головна галузь її використання це фронт енд розробка веб сторінок, тобто частина сайту яку бачить користувач, проте для цієї мови було створено велику кількість бібліотек та фреймворків які дозволяють використовувати її також для серверної частини сайту або навіть поза веб розробкою. Вона підтримується всюди, будь-який пристрій, операційна система та браузер будуть підтримувати JavaScript, що дозволяє не турбуватися про те, що додаток чи сайт може не підтримуватися в якогось користувача. Підтримка в будь якій сфері відкриває двері для того, щоб цю мову можна було використовувати для

майже будь-якої місії, а тому якщо початківець не захоче займатися веб розробкою, з знанням лише мови JavaScript він зможе перейти до іншої сфери розробки [1].

Сучасний JavaScript - це "безпечна" мова програмування. Він не надає низькорівневий доступ до пам'яті або процесора, тому що спочатку був створений для браузерів, які цього не вимагають.

Можливості JavaScript залежать від оточення, в якому він працює. Наприклад, Node.JS підтримує функції читання/запису довільних файлів, виконання мережевих запитів тощо.

У браузері для JavaScript доступне все, що пов'язане з маніпулюванням веб-сторінками, взаємодією з користувачем та веб-сервером.

Наприклад, у браузері JavaScript може:

− додавати новий HTML-код на сторінку, змінювати наявний вміст, модифікувати стилі;

− реагувати на дії користувача, клацання миші, переміщення вказівника, натискання клавіш;

− надсилати мережеві запити на віддалені сервери, завантажувати та завантажувати файли (технології AJAX та COMET);

− отримувати та встановлювати куки, ставити запитання відвідувачу, показувати повідомлення;

− запам'ятовувати дані на стороні клієнта (local storage).

Через те, що у неосяжній мережі Інтернет занадто багато інформації, часто просто блукаєш веб-сторінками і не знаєш, що саме тобі потрібно. Навіть якщо і обираєш для себе конкретний сервіс, потім розумієш, що це не зовсім те, що тобі потрібно. Через цю проблему багато людей можуть бути дезінформованими і, думаючи, що вони праві, поширювати неправду серед інших, породжуючи багато плутанини.

Проблема обирання інформації, розрахованої на іншу аудиторію, є не менш

поширеною, ніж попередні, але часто люди встигають зрозуміти, що дане скупчення знань не для їх рівня. Усе ж таки бувають моменти, коли на початку начебто все зрозуміло, але чим дальше – тим більш заплутано та незрозуміло.

Отже, враховуючи усе вищесказане, зроблено висновок, що потрібно розробити безкоштовну веб-систему для вивчення мови програмування JavaScript, у якій буде зібраний якісний матеріал, для поглинання якого не потрібно буде звертатися до сторонніх ресурсів.

1.2 Порівняльний аналіз аналогів розроблюваної системи

Найкращим шляхом визначення актуальності розробки веб-системи є проведення аналізу її аналогів та їх порівняння за найголовнішими для навчальних ресурсів критеріями оцінювання.

Оскільки мова JavaScript є однією з найпоширеніших, існує багато ресурсів, які допомагають з її вивченням. Майже кожен з них виконує свою окрему роль, але більшість являють собою документацію до того як працювати з цією мовою. Документація не є гарним способом вивчення, тому що інформація там подана в не структурованому виді, і якщо початківець захоче дізнатися щось нове, він може наткнутися на тему для розуміння якої йому потрібно вже мати немалий багаж знань.

Розглянемо найпопулярніші ресурси для вивчення мови програмування JavaScript.

MDN Web Docs (див. рис. 1.1) – це ресурс для вивчення веб розробки від Mozilla. В ньому збережена документація для майже всього, що необхідно знати в розробці сайтів, в тому числі і до мови JavaScript, але подана вона в такому виді, який буде важко зрозуміти початківцям. Приклади застосування того чи іншого аспекту мови показані вже для досвідчених користувачів, а для новачків це може бути зовсім незрозумілим [2].

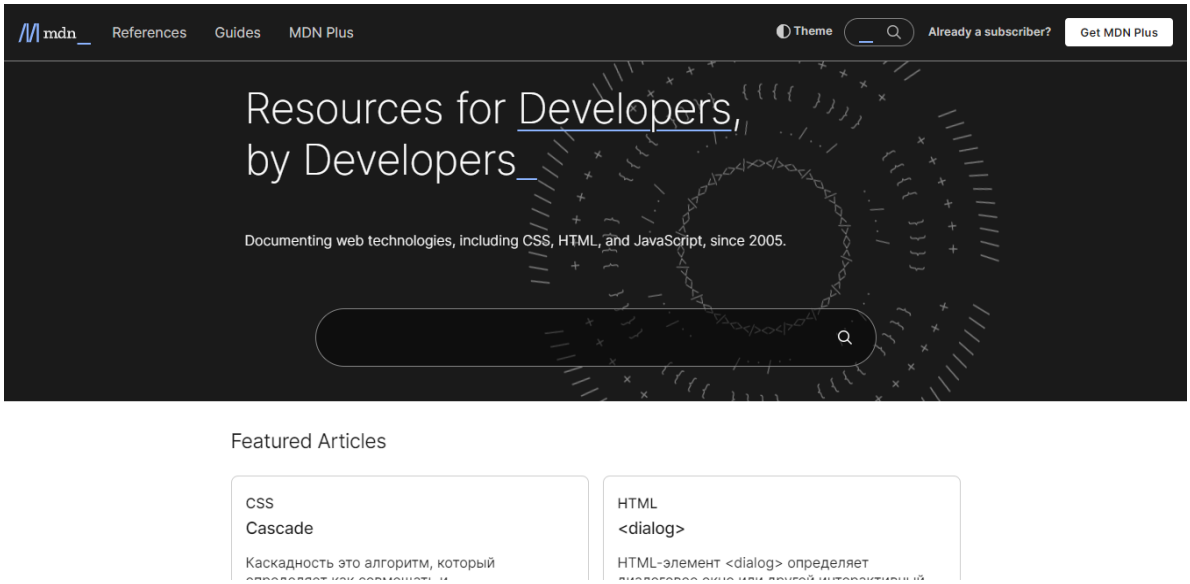


Рисунок 1.1 – Ресурс MDN Web Docs

W3Schools (див. рис. 1.2) – це ресурс для вивчення повного стеку технологій для розробки сайтів, і бекенд і фронтенд. Він містить документацію до мови JavaScript та задачі для перевірки своїх знань. Окрім документації там доступні платні курси та навчальні відео для прискорення навчання. Там наведено багато прикладів взаємодії JavaScript з іншими мовами програмування, що допомагає початківцям краще зрозуміти сфери його використання [3].

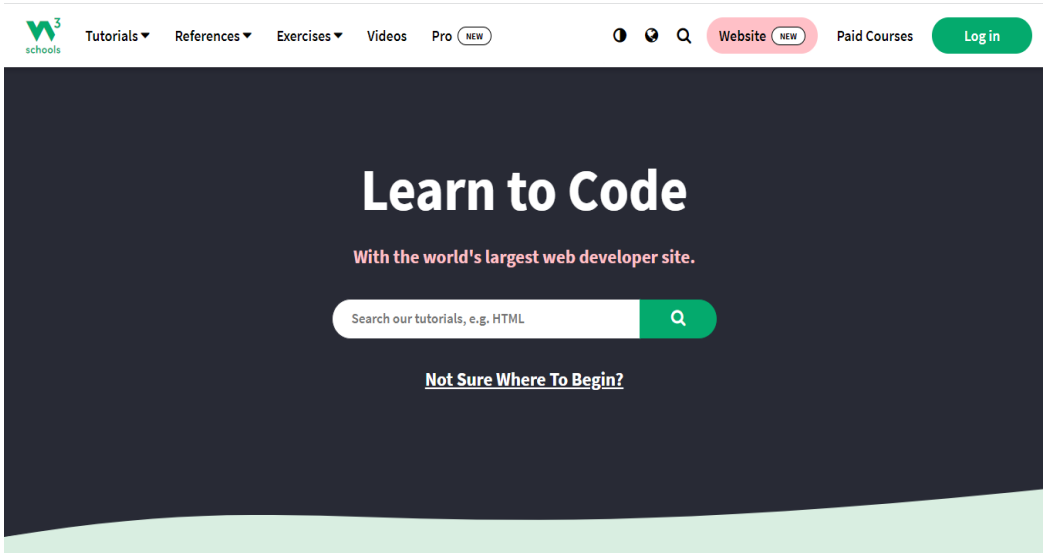


Рисунок 1.2 – Ресурс W3Schools

JavaScript.info (див. рис. 1.3) – веб-ресурс для вивчення мови програмування JavaScript який містить в собі документацію та практичні задачі для закріплення матеріалу. На цьому сайті інформація про мову чудово структурована і подана в вигляді курсу від найнижчого рівня знань до досвідчених користувачів, що набагато спрощує її вивчення. На сайті розглянуто найважливіші теми з поясненнями для людей різних рівнів розуміння, а до практичних задач в кінці розділів завжди є відповіді з поясненнями [4].

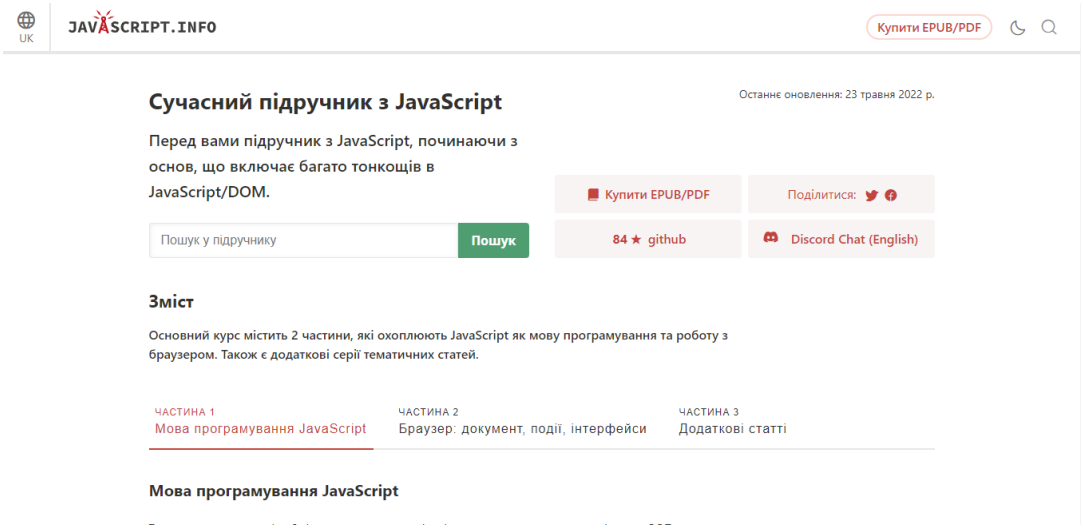


Рисунок 1.3 – Ресурс JavaScript.info

Результати порівняння аналогів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів

Критерії оцінювання	MDN Web Docs	W3Schools	JavaScript.info	StudyJS
Можливість створення аккаунта	-	+	+	+
Наявність тестувань	-	+	+	+
Збереження прогресу навчання	-	-	-	+
Структуроване навчання	-	+	+	+

З розробленою веб-системою під назвою “StudyJS” було порівняно її аналоги за такими критеріями:

1. Можливість створення аккаунта – якщо в користувача є можливість авторизуватися на сайті, в нього є можливість зберігати, наприклад, свій прогрес або підписуватися на якісь теми та їх оновлення якщо це доступно.

2. Наявність тестувань – можливість перевірки набутих знань. При перевірці своїх знань набагато легше дізнатися, які теми варто підтянути.

3. Збереження прогресу навчання – за наявності на сайті можливості запам’ятати де саме ви зупинилися можна зберегти багато часу через відсутність необхідності згадувати, що було пройдено, а що ні.

4. Структуроване навчання – на багатьох навчальних ресурсах інформація представлена без чуткої структури і початківцю може бути незрозуміло, які теми йому потрібно вчити на початку, а які необхідні для досвідчених користувачів.

Виходячи з результатів аналізу аналогів, можна зробити висновок, що веб-система для вивчення мови програмування JavaScript є актуальною на сьогоднішній день та має переваги, які роблять її конкурентоспроможною.

1.3 Аналіз методів розв’язання поставленої задачі

Одним з перший етапів розробки є аналіз методів розв’язання поставленої задачі. Від цього залежить майбутнє продукту, тому що якщо буде обрано неправильній метод то в найгіршому випадку необхідно буде повністю переписувати задачу з початку, бо переробити щось коли воно неправильно розпочато буде дуже важко.

Одним з найпростіших варіантів є використання конструктора сайтів для розробки зовнішнього вигляду сайту і готові бібліотеки з рішеннями для функціоналу. Для бази даних можна використовувати доступні бази даних з сторонніх ресурсів. Цей метод не дивлячись на свою простоту має багато недоліків. Головним мінусом є залежність від ресурсів над якими ви не маєте

контролю, через це сайт може змінитися в будь-який момент або взагалі перестати працювати. Такий сайт буде дуже важко розширювати, а можливості для цього будуть дуже обмежені.

Другим методом буде взяти готовий схожий проект і переробити його під свої потреби. В вільному доступі є багато ресурсів де можна знайти сайт, або ж можна використовувати свої старі свої старі роботи. Перевагою цього методу є те, що більша частина роботи вже буде виконана і потрібно буде лише переробити або розробити базу даних і дещо змінити клієнтську частину. Недоліком є те, що такий метод забов'язує вас залежати від не зв'язаного з вашим проектом коду. Також буде необхідно провести дуже великий рефакторінг, тобто переписування коду щоб він відповідав темі проекту, а це означає, що кількість роботи особливо не зменшується, а її складність тільки збільшується.

Третім, і єдиним правильним методом буде розробка продукту з самого початку. Вся архітектура проекту буде складена вами, а тому будь-яке розширення або зміни будуть проводитися дуже легко, так як вам буде відомий будь-який аспект проекту. Завдяки цьому методу ви будете незалежні або мінімально залежні від сторонніх ресурсів. В такому випадку, весь продукт буде залежати лише від того, як ви вирішите його написати, а саме головне, структура бази даних, буде саме такою яка вам необхідна, а тому доступ до даних буде максимально простим.

Отже, серед розглянутих методів розв'язання поставленої задачі було обрано використовувати метод, в якому весь продукт розробляється з самого початку, тому що так буде повна свобода в розробці і підтримувати продукт буде дуже легко.

1.4 Постановка задач дослідження

Проаналізувавши стан галузі вивчення програмування JavaScript та дослідивши методи розробки продукту для розв'язання теми «Розробка веб-системи для вивчення мови програмування JavaScript», було виділено наступні

задачі, які будуть забезпечувати високу якість продукту і його модулів:

− визначити найбільш ефективний метод вивчення мови програмування Javscript;

− розробити структуру веб-сайту;

− розробити клієнтську частину в якій користувачам буде зручно оперувати;

− розробити алгоритм перевірки задач користувачів після проходження тем;

− розробити базу даних, яка буде містити інформацію про користувачів та їх прогрес в навчанні;

− розробити модуль реєстрації та авторизації користувачів у веб-систему;

− здійснити тестування програмного модуля.

Отже, було поставлено задачі дослідження на тему «Розробка веб-системи для вивчення мови програмування JavaScript».

1.5 Висновки

У першому розділі було проаналізовано сучасний стан галузі вивчення мови програмування JavaScript.

Провівши порівняльний аналіз аналогів, було зроблено висновок, що дана розробка є актуальною та задовільнить потреби користувачів, які бажають увійти в сферу інформаційних технологій або бажають покращити свої знання.

Обрано метод розробки веб-системи власноручно з самого початку через його суттєві переваги.

Було визначено задачі дослідження, які потрібно виконати для успішної розробки на тему «Розробка веб-системи для вивчення мови програмування JavaScript».

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз структурування навчальних порталів

Одним з найголовніших аспектів навчальних порталів є організація їх контенту. При першому переході на сторінку користувач повинен знати як йому знайти все, що він бажає бо його головною ціллю є навчання, а не пошук важливих частин сайту.

Якщо ресурс являє собою лише, наприклад, документацію до мови програмування JavaScript, то його структура повинна бути такою, що до будь якого розділу можна потрапити з головної сторінки. Користувачам не потрібні новини, соціальні активності чи щось інше коли вони прийшли прочитати про якусь особливість мови програмування. Також на таких порталах теми повинні мати вкладеність інших тем всередині себе. Якщо розглядати документацію чи портал з курсами по програмуванню, то там є дуже велика кількість інформації, яку необхідно групувати, і якщо цього не робити то сайт стане важким для читання.

Важливим аспектом структури сайту є його навігація. Вона повинна бути на видному місці і мати декілька головних частин сайту з яких, в свою чергу, можна перейти до інших. При пересуванні по веб-сайту користувач повинен знати де він зараз знаходиться і як йому легко перейти до іншого розділу. Для цього в навігації можна підсвічувати розділ на якому зараз знаходиться юзер і завжди тримати навігацію в нього перед очима.

Найкращим способом показати структуру сайту користувачеві є використання сторінки з мапою сайту. Там присутні всі розділі і підрозділи продукту і з неї можна потрапити будь-куди. Її використання особливо поширене в інтернет магазинах, де користувачі можуть побачити які, наприклад, розділи одягу чи взуття присутні в магазині. На навчальних ресурсах вона може бути дуже зручною, тому що користувачу не потрібно переходити по розділам щоб знайти їх

підрозділи які йому необхідні, він може просто відкрити мапу сайту та знайти там те, що йому необхідно.

До структури веб-сайту можна віднести також окремо структуру його сторінок. Якщо існують розділи, головний сенс яких повторюється, то вони мають мати схожий зовнішній вигляд і організацію контенту. Наприклад, якщо користувач розглядає якийсь розділ з певною темою мови програмування, і на сторінці присутні: визначення теми, особливості, використання на практиці, висновок, практичні завдання; то в інших темах послідовність і структура контенту повинні бути такими ж. Якщо ж на окремій темі відсутні деякі розділи, то вони можуть бути не відображеними, але послідовність завжди повинна бути однаковою.

Головною ознакою гарної структури веб-сайту, є можливість користувача пересуватися по головним його розділам якнайшвидше. Щоб виявити, якими розділами користувачі цікавляться більше, можна аналізувати їх активністьза допомогою різних додатків, і тим самим дізнаватися, до яких розділів потрібно спростити доступ, а які можна прибрати, щоб вони не займали місце на сторінці.

Отже, було розглянуто важливість та основні можливості структурування продукту.

2.2 Розробка структури інтерфейсу веб-системи для вивчення мови програмування JavaScript

Розробка структури веб-сайту – це планування та реалізація навігації по сторінках чи блоках веб-сайту. Її метою є спрощення використання сайту користувачем і забезпечення швидкого та зрозумілого переміщення по ньому. Головними вимогами до структури веб-сайту є: назва посилань повинна відповідати сторінкам, на які вони посилаються, блоки повинні бути розташовані в логічній послідовності, зв’язок сторінок між собою має бути зрозумілим, доступ до головних сторінок має бути легким.

Якщо структура сайту є логічною та зрозумілою, користувач з більшою вірогідністю продовжить використовувати його в майбутньому, а якщо ні, то він просто почне шукати новий, іноді навіть незважаючи на те, що інші можуть мати в собі меншу кількість контенту. Чим більше той, хто використовує продукт, не розуміє, тим швидше росте його незадоволеність, а саме це відбувається якщо структура веб-додатку виконана погано.

Структуру веб-сайту поділяють на 3 види [4]:

1. Лінійна. В цій структурі передбачений перехід між сторінками одна за одною. Відсутність підрозділів та зрозумілість кожного переходу робить цей вид чудовим вибором для малих сайтів, або тих, де інформація йде послідовно, без необхідності створювати додаткові розділи з сторінками.

2. Деревоподібна. Така структура передбачає, що доступ до сторінок здійснюється на різних «гілках», тобто до деяких сторінок користувач може отримати доступ лише якщо він перед цим знаходився на сторінці цієї ж «гілки».

3. Довільна. Доступ до сторінок відбувається в довільному порядку, тобто з головної, користувач може потрапити до найвіддаленішої без необхідності відвідувати перед цим сторінки які логічно зв’язані з нею. Такий підхід зручний, якщо всі сторінки не зв’язані між собою напряму або якщо на сайті міститься багато інформації по різним темам і користувач обирає лише одну, яка йому необхідна.

Враховуючи специфіку веб-системи, було вирішено використовувати вільну структуру. З головної сторінки буде доступ до будь-якого розділу сайту, тому що користувачу зручніше одразу перейти до цікавлячої його теми по мові програмування JavaScript чим переходити по розділам в її пошуках. Це було досягнуто завдяки створенню мапи сайту на головній сторінці, тобто всі теми відображені одразу на головній сторінці .

Було розроблено структуру головної сторінки веб-системи і відображено її у вигляді схеми (див. рис. 2.1).

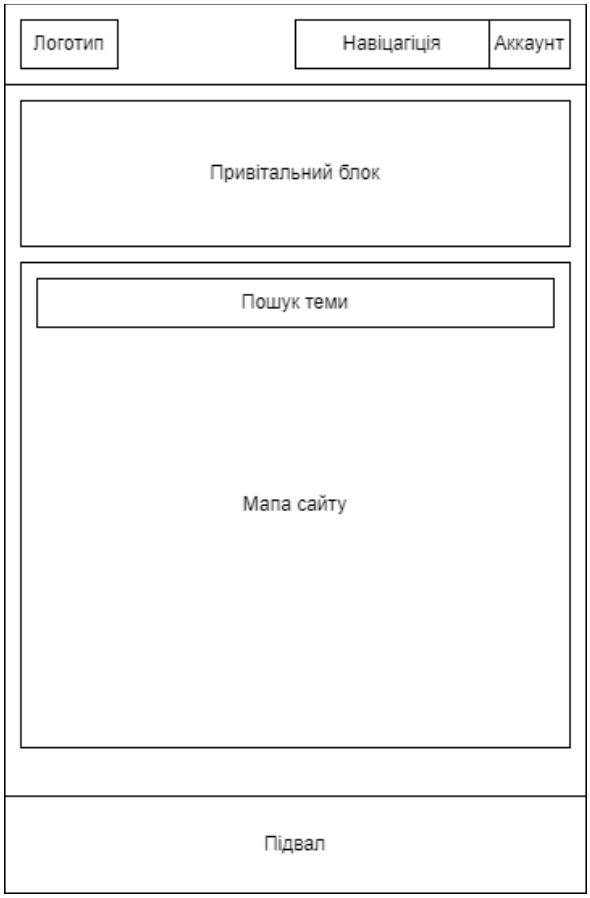


Рисунок 2.1 – Структура головної сторінки веб-системи

Через пункт «Навігація» користувач може потрапити до головних розділів сайту, тобто на головну сторінку, посібник чи на свій аккаунт. Привітальний блок виконує інформативну функцію і дає користувачу інформацію про те, де він знаходиться. Мапа сайту це всі навчальні розділі, тут користувач може перейти до будь-якої теми, яка його цікавить. На підвалі сайту користувач може побачити інформацію про розробника та те, як він може з ним зв’язатися в разі необхідності.

Така структура містить весь головний контент на головній сторінці і не є навантаженою зайвим контентом. Користувач може одразу зрозуміти, як користуватися веб-сайтом, а тому з більшою вірогідністю залишиться саме на ньому.

Отже, було розроблено структуру інтерфейсу головної сторінки веб-системи для вивчення мови програмування JavaScript.

2.3 Розробка навчальної програми для вивчення мови програмування JavaScript

Розробка навчальної програми для веб-сайту головною функцією якого є вивчення чогось є самою головною його частиною. Якщо сайт має чудовий дизайн, широкий функціонал та високу продуктивність, але не виконує функції освітнього порталу, яким він себе представляє, то користувач покине його одразу після того як зрозуміє, що він не має змоги знайти інформацію, яка його цікавить.

Найголовніше, що мають надавати такі сайти, – це легкодоступну інформацію, яку може використовувати людина будь-якого рівня навичок, тобто теми не повинні буде складно поясненими, але й мають мати всі головні аспекти в собі, щоб навіть досвідчені користувачі могли вивчати щось нове.

Навчальну програму було складено з самих головних аспектів мови програмування JavaScript. Користувач може як проходити їх по черзі, так і вчити лише ті теми, які його цікавлять. Для початківців бажано проходити теми по черзі, так як програму складено так, що навчання починається с самого простого і далі йде вверх по складності.

Якщо ж користувач забудеться щось в процесі вивчення складних тем, в них завжди будуть посилання на старі, з нагадуванням про вивчений матеріал.

Розроблена навчальна програма поділена на розділи та підрозділи. Розділи першого рівня показують на угрупування схожих тем, які будуть розглядатися, а розділи другого рівня – це вже самі теми, які будуть вивчатися. Це зроблено для того, щоб теми не виглядали як просто набір непов’язаних один з одним розділів, а мали певну структуру, по якій користувач може зрозуміти, які саме аспекти мови програмування JavaScript є важчими за інші.

Вся структура навчання знаходиться на головній сторінці веб-сайту (див. рис. 2.2), тому користувач одразу може побачити, що саме він буде вивчати. Вона представляє собою перелік всіх розділів, які користувач може відвідати під час роботи з веб-сайтом.

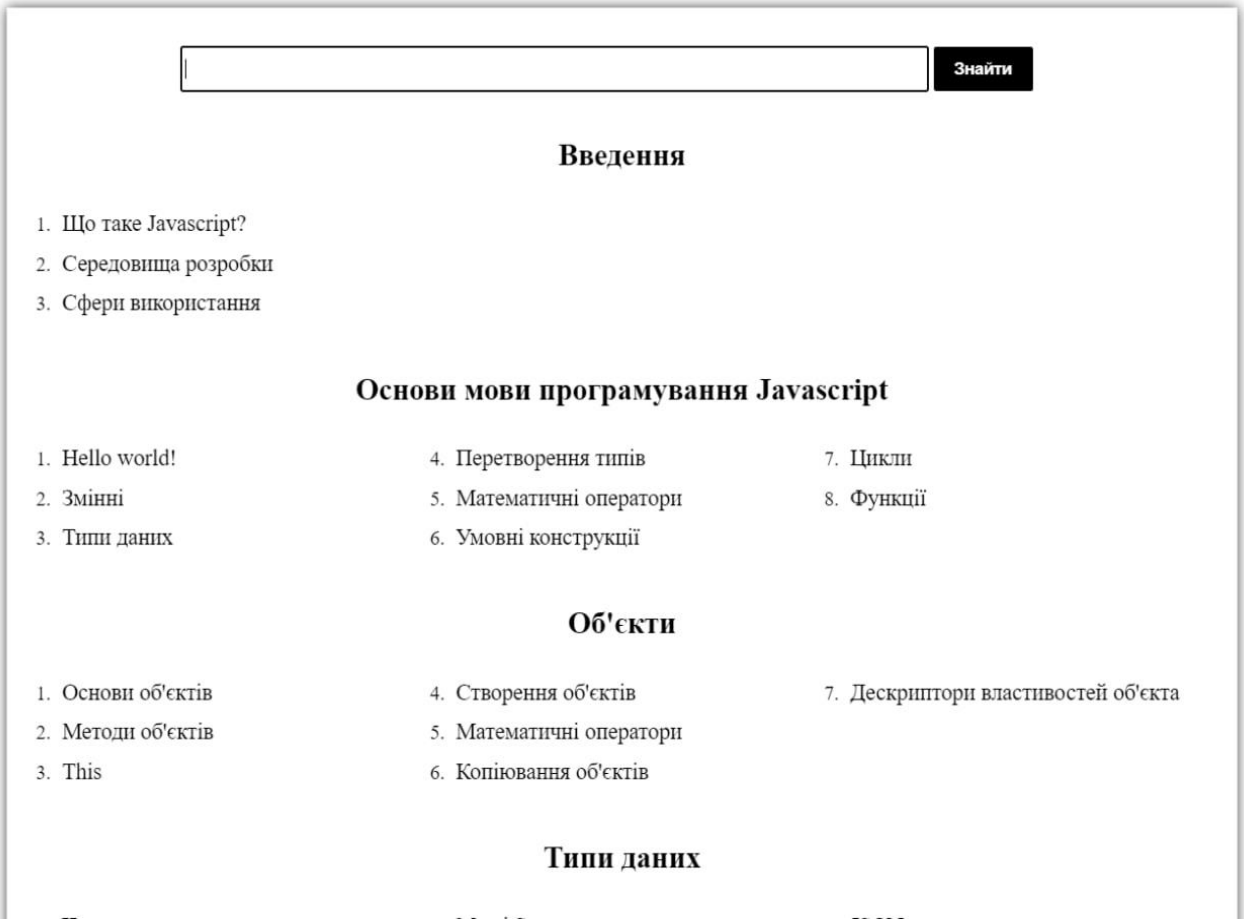
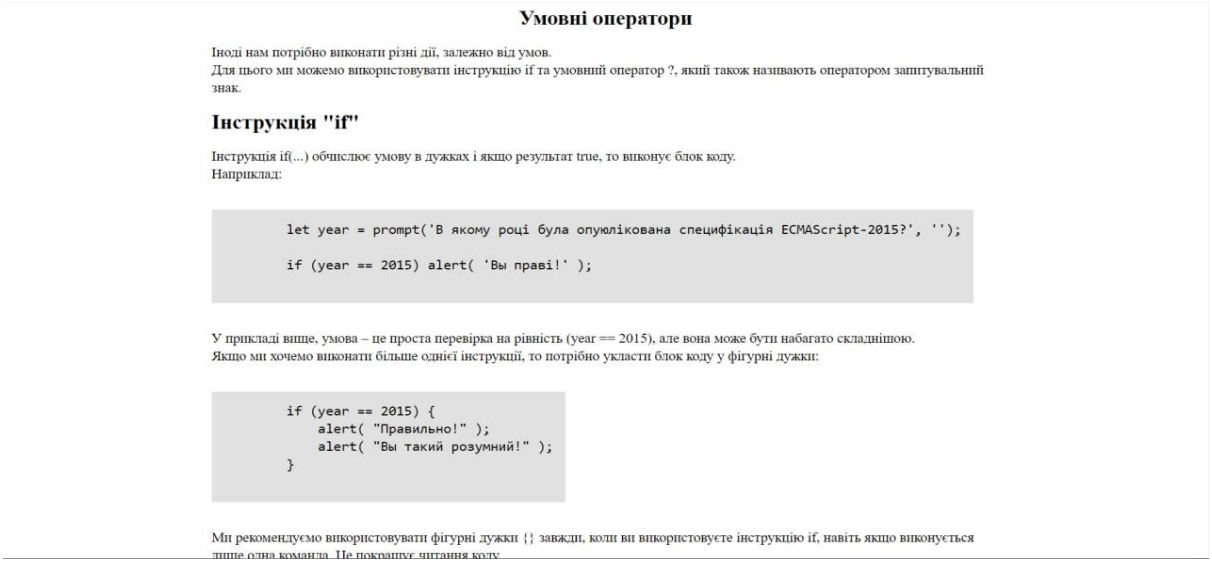


Рисунок 2.2 – Навчальна програма веб-системи

Кожен урок має однакову структуру: визначення теми, інформація з прикладами коду, задачі для перевірки знань. Сам інформаційний контент повністю відрізняється по всім темам, але його структура залишається схожою, щоб користувач знав, що саме він може дізнатися в розділі.

Єдиним виключенням з структури можуть бути інформаційні розділи які не пов’язані з написанням коду, а лише з поясненням роботи мови JavaScript, або чогось з нею пов’язаного. До таких сторінок також можуть відноситися сторінки, в яких розповідається про бібліотеки чи фреймворки до мови програмування JavaScript. Вони не можуть мати таку ж саму структуру сторінки тому що інформація в них сильно відрізняється від простих розділів для вивчення. Приклад структури сторінки наведено на рисунку 2.3.



Умовні оператори

Іноді нам потрібно виконати різні дії, залежно від умов.
Для цього ми можемо використовувати інструкцію if та умовний оператор ?, який також називають оператором запитувальний знак.

Інструкція "if"

Інструкція if(...) обчислює умову в дужках і якщо результат true, то виконує блок коду.
Наприклад:

```
let year = prompt('В якому році була опуюлікована специфікація ECMAScript-2015?', '');

if (year == 2015) alert( 'Вы праві!' );
```

У прикладі вище, умова – це проста перевірка на рівність (year == 2015), але вона може бути набагато складнішою.
Якщо ми хочемо виконати більше однієї інструкції, то потрібно укласти блок коду у фігурні дужки:

```
if (year == 2015) {
    alert( "Правильно!" );
    alert( "Вы такий розумний!" );
}
```

Ми рекомендуємо використовувати фігурні дужки {} завжди, коли ви використовуєте інструкцію if, навіть якщо виконується лише одна команда. Це покращує читання коду.

Рисунок 2.3 – Приклад структури уроку

Отже, було розглянуто розробку начальної програми для вивчення мови програмування JavaScript.

2.4 Розробка алгоритму роботи веб-системи

Для кращого розуміння роботи веб-системи потрібно розробити алгоритми її роботи. Вона містить 2 головних логічних блоки: реєстрація/авторизація користувачів та збереження прогресу навчання.

Алгоритм – набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій; система правил виконання дискретного процесу, яка досягає поставленої мети за скінченний час. Для візуалізації алгоритмів часто використовують блок-схеми. Для комп'ютерних програм алгоритм є списком деталізованих інструкцій, що реалізують процес обчислення, який, починаючи з початкового стану, відбувається через послідовність логічних станів, яка завершується кінцевим станом [5].

Алгоритм реєстрації/авторизації користувача представляє собою перевірку чи має користувач аккаунт в системі, якщо так, то починається процес авторизації, в інакшому випадку – реєстрації. При авторизації користувач заповняє 2 поля

даних, логін та пароль, і якщо щось з них невірне то система просить його ввести їх знову. Якщо проходить процес реєстрації, то користувачу потрібно ввести вже 4 поля: логін, пароль, ім’я та прізвище. Якщо всі дані введено правильно то система створює нового користувача, інакше – просить ввести дані знову. Блок-схема алгоритму представлена на рисунку 2.4.

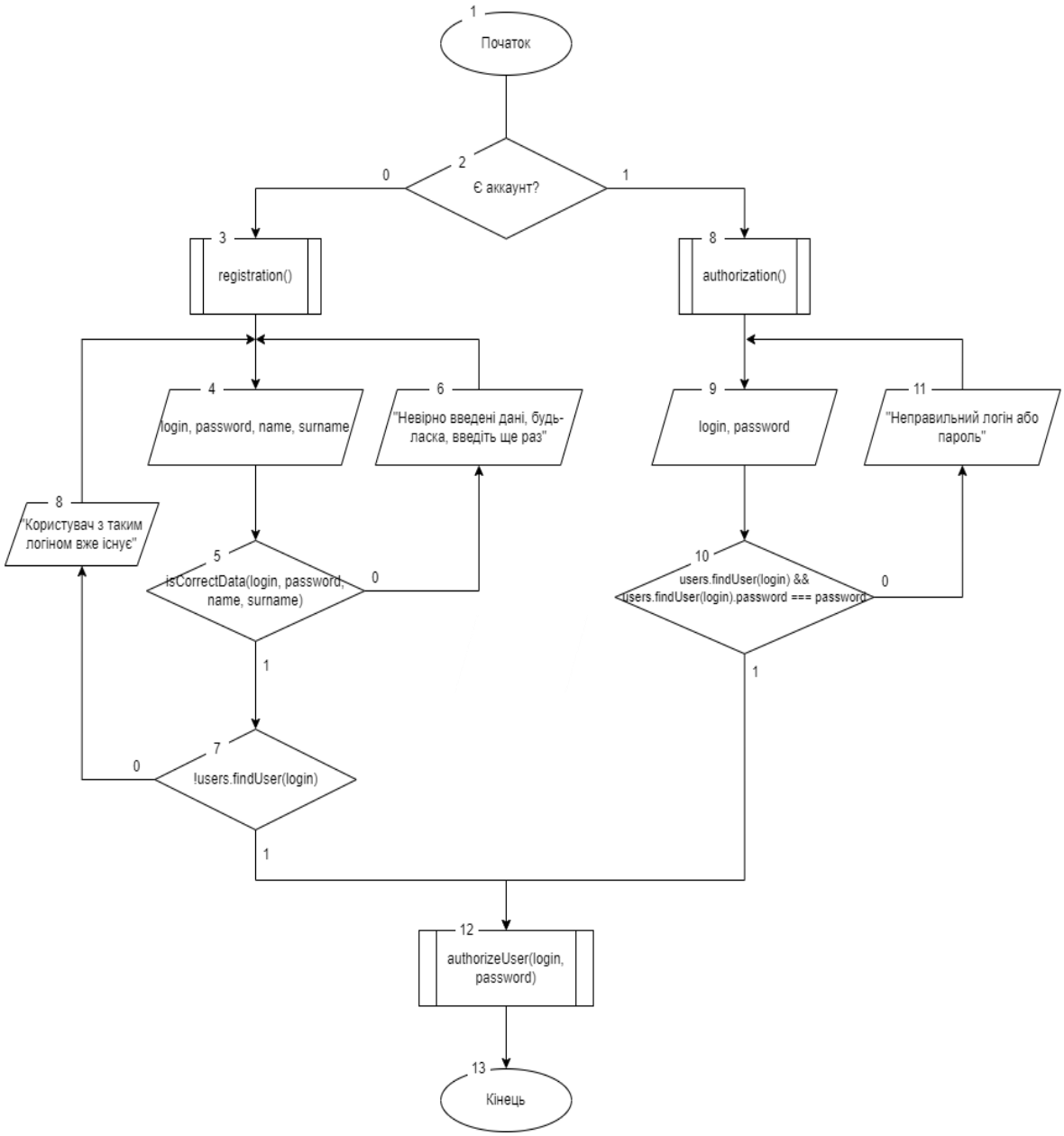


Рисунок 2.4 – Блок-схема алгоритму реєстрації/авторизації

Алгоритм збереження прогресу навчання починає свою роботу коли користувач закінчує вивчення теми і проходить задачі в її кінці. Спочатку алгоритм перевіряє чи авторизований користувач, якщо ні, то його перенаправляє на вікно авторизації. Після цього система додає йому в профіль результат проходження завдань. Блок-схема алгоритму представлена на рисунку 2.5.

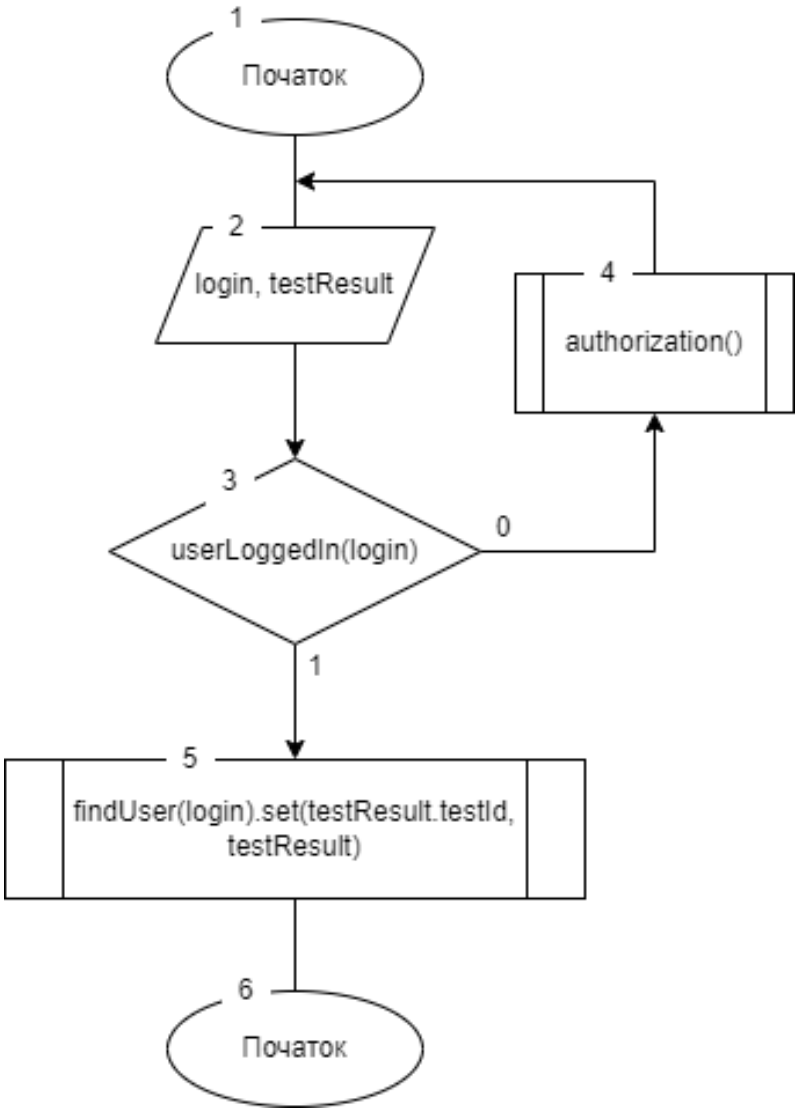


Рисунок 2.5 – Блок-схема алгоритму збереження прогресу навчання

Отже, було розглянуто алгоритми роботи модулів веб-системи для вивчення мови програмування JavaScript.

2.5 Висновки

У другому розділі було проаналізовано структурування навчальних порталів та його типи. Було розроблено структуру інтерфейсу веб системи для вивчення мови програмування JavaScript. Розроблено навчальну програму та її дизайн з темами для вивчення мови програмування JavaScript. Було розглянуто головні алгоритми роботи веб-системи та створено блок-схеми для них.

3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ПРОДУКТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації веб-системи

Раніше для розробки веб-сторінок використовувався однаковий стек технологій і майже ніхто не використовував нічого іншого тому що це було неможливо, або складно в реалізації. Сьогодні веб-розробка надає широкий вибір технологій, які можна обрати і від їх вибору буде залежати розширюваність продукту і його швидкодія.

Для розробки було обрано мову розмітки HTML, каскадні таблиці стилів CSS, мову програмування JavaScript та платформу Node.js.

HTML – це мова гіпертекстової розмітки завдяки якій браузер відображає контент на сторінці. Структура файлу такої мови повністю складається з тегів, які в залежності від назви, дають браузеру зрозуміти, що знаходиться всередині них. Теги використовуються для відображення контенту, який ми в них записуємо, або для команд браузеру як відображати цей контент чи по яким словам можна буде знайти цю сторінку в пошукових системах. Оскільки ця мова дозволяє відображати контент на веб-сторінці, її було обрано для розробки веб-системи для вивчення мови JavaScript [6].

CSS – це каскадні таблиці стилів, які використовуються для стилізування HTML сторінок. Завдяки ним можна повністю змінювати зовнішній вигляд сторінки і відображати її не лише звичайним текстом, а в стилізованих блоках. За допомогою CSS можна, наприклад, змінити колір заднього фону сторінки або тексту, по різному відображати контент, змінювати розмір та форму блоків, змінювати вигляд елементів при взаємодії з ними, створювати анімації тощо. Було вирішено використовувати ці каскадні таблиці стилів, тому що сторінка з лише текстом не приверне уваги користувача, а стилізована сторінка зробить це з більшою вірогідністю [7].

Для покращення використання каскадних таблиць стилів CSS, було використано препроцессор SCSS для нього. SCSS – це синтаксис скриптованої метамови SASS, яка перетворює написаний код в CSS стилі. Його перевагами є наступне [8]:

1. Можливість вкладення правил одне в одне при написанні коду. При компіляції це перетвориться на те ж саме, як би користувач писав це на мові CSS, але це полегшує написання коду.

2. Створення змінних для правил, які часто використовуються. Наприклад, можна створити змінну з кольором і дати їй назву цього кольору. Таким чином, якщо вам код буде доповнятися кимось другим, він одразу зрозуміє який колір тут використовувався.

3. Створення спеціальних змінних з набором правил які можуть бути використані знову. Якщо на сторінці є, наприклад, кнопки з однаковим стилем, але різним розташуванням, то можна створити набір правил який буде використовуватися в для цих кнопок. Завдяки цьому можна уникнути повторення коду.

Для того, щоб перетворювати SCSS код в CSS стилі, потрібно використовувати компілятор. Для цього було використано Webpack, який дозволяє прописувати правила по тому, як буде будуватися фінальний вигляд продукту. Окрім компіляції стилів, він може збирати багато JavaScript файлів в один. Це робиться для того, щоб можна було писати окремі модулі коду з назвами файлів, які описують, що в них написано, а для фінального продукту збирати їх всіх в один файл і підключати до сторінки [9].

Javscript – це високорівнева мова програмування завдяки якій можна «оживити» веб сторінку. Завдяки цій мови можна додавати новий контент на сторінку або оновлювати старий, обробляти введені користувачем дані в формах або керувати додаванням стилів до блоків на сторінці. Без цієї мови, сторінку можна буде лише переглядати, а єдина взаємодія, яку користувач зможе з нею

провести це перейти на іншу сторінку по посиланню. Було вирішено використовувати цю мову, бо для пошуку розділів, зберігання прогресу вивчення, реєстрації та багато іншого потрібна взаємодія користувача з сторінкою [10].

Node.js – це платформа яка дозволяє виконувати JavaScript скрипти на сервері та відправляти користувачеві результат їх виконання. Завдяки їй, можна розробляти серверну частину сайту на JavaScript, тобто можна обробляти форми, працювати з базами даних, надсилати користувачу HTML сторінки тощо. Було вирішено використовувати цю платформу через можливість використання мови Javscript для серверної частини, найголовніше для зв’язку з базою даних та обробкою форм [11].

Отже, було проведено варіативний аналіз і обґрунтовано вибір засобів для реалізації веб-системи для вивчення мови програмування JavaScript.

3.2 Розробка бази даних веб-системи для вивчення мови JavaScript

Для організації контенту на веб-системі було вирішено створити базу даних. Вона буде зберігати такі відомості: інформація про користувача, прогрес проходження тем, інформація про теми, прогрес проходження завдання.

Система управління базами даних – комплекс програм, що дозволяють сформувати базу даних та маніпулювати даними (вставляти, оновлювати, видаляти та вибирати). Система гарантує безпеку, надійність збереження та цілісність даних, а також надає засоби для адміністрування БД.

Архітектурно СУБД складається з двох великих компонент. За допомогою мови опису даних створюються описи елементів, груп та записів даних, а також взаємозв’язки між ними, які, як правило, задаються у вигляді таблиць. Для виконання операцій з базою даних в прикладних програмах використовується мова маніпулювання даними. Фактична структура фізичного зберігання даних відома тільки СУБД [12].

MySQL Workbench – інструмент для візуального проектування баз даних,

що інтегрує конструювання, моделювання, створення та використання БД у єдине безшовне оточення для системи баз даних MySQL [13]. Надає можливість із легкістю створювати базу даних, додавати, редагувати та видаляти таблиці, а також їх елементи.

Виходячи із всього вищеописаного, було вирішено, що для розробки бази даних буде використовуватися СУБД MySQL Workbench.

Перечислені суті та визначені ключі, які однозначно ідентифікують поля даних сутей:

– user (<login>, password, name, surname);

– topic (<topicId>, name);

– question (<questionId>, answer).

Таблиця «user» містить інформацію про користувача веб-системи. Нові записи у таблицю відбуваються під час реєстрації облікового запису юзера.

Таблиця «topic» містить інформацію про теми, які потрібно проходити у веб-системі.

Таблиця «question» містить інформацію про завдання, які потрібно виконати під кінець проходження теми. За рахунок неї рахується процес проходження теми.

Створену таблицю характеристики зв’язків бази даних соціальної мережі можна побачити у таблиці 3.1.

Таблиця 3.1 – Характеристики зв’язків бази даних веб-системи

Ім’я суті 1	Ім’я суті 2	Тип зв’язку	Клас належності
user	question	N:M	Обов., обов.
user	topic	N:M	Обов., обов.
topic	question	1:N	Обов., обов.

У результаті зв’язку сутностей «багато-до-багатьох», створилися дві проміжні таблиці: «progress» та «questionprogress». Таблиця «progress» створена унаслідок зв’язку таблиць «user» та «topic». У свою чергу таблиця

«questionprogres» створена унаслідок зв’язку таблиць «user» та «question».

На основі таблиці 3.1 було розроблено ER-модель бази даних веб-системи. На рисунку 3.1 представлено результуючу ER-модель бази даних веб-системи для вивчення мови JavaScript.

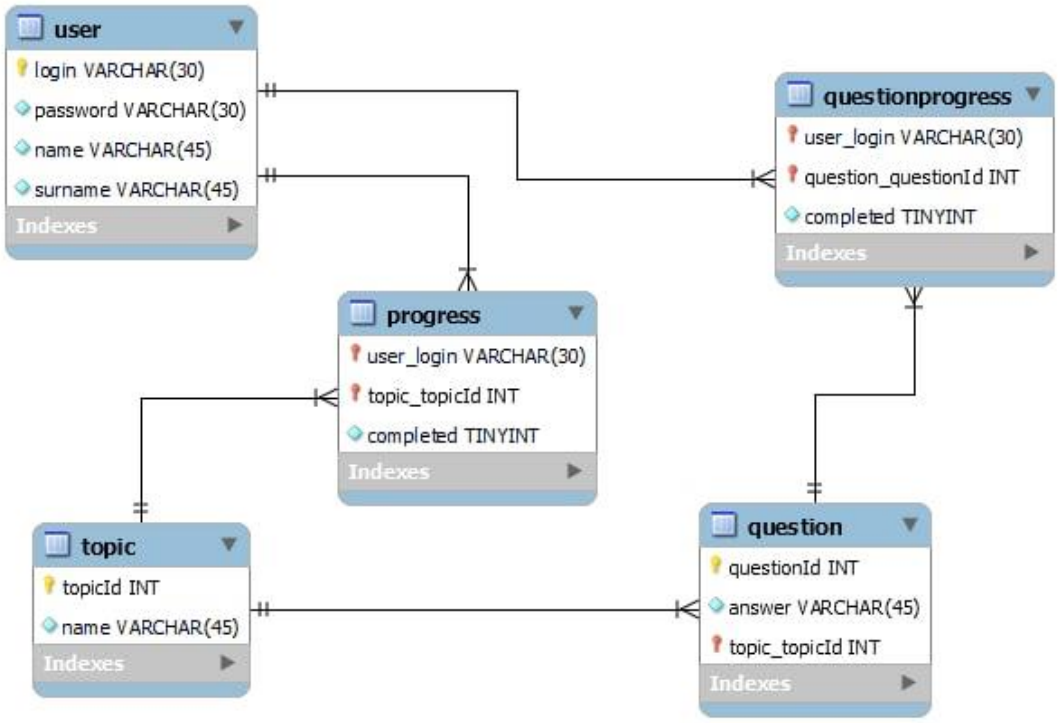


Рисунок 3.1 – ER-модель бази даних

Розглянемо на прикладі таблиці «user» табличний вигляд сутності. Для того, щоб вивести усі записи по всім атрибутам, потрібно використати найпростіший SQL-запит:

```
SELECT * FROM mydb.user;
```

У результаті запиту отримуємо таблицю із усіма записами, які містяться у базі даних. Кількість записів дорівнює кількості користувачів, які мають обліковий запис у веб-системі для вивчення мови програмування JavaScript.

На рисунку 3.2 показано SQL-запит та табличний вигляд сутності «user».

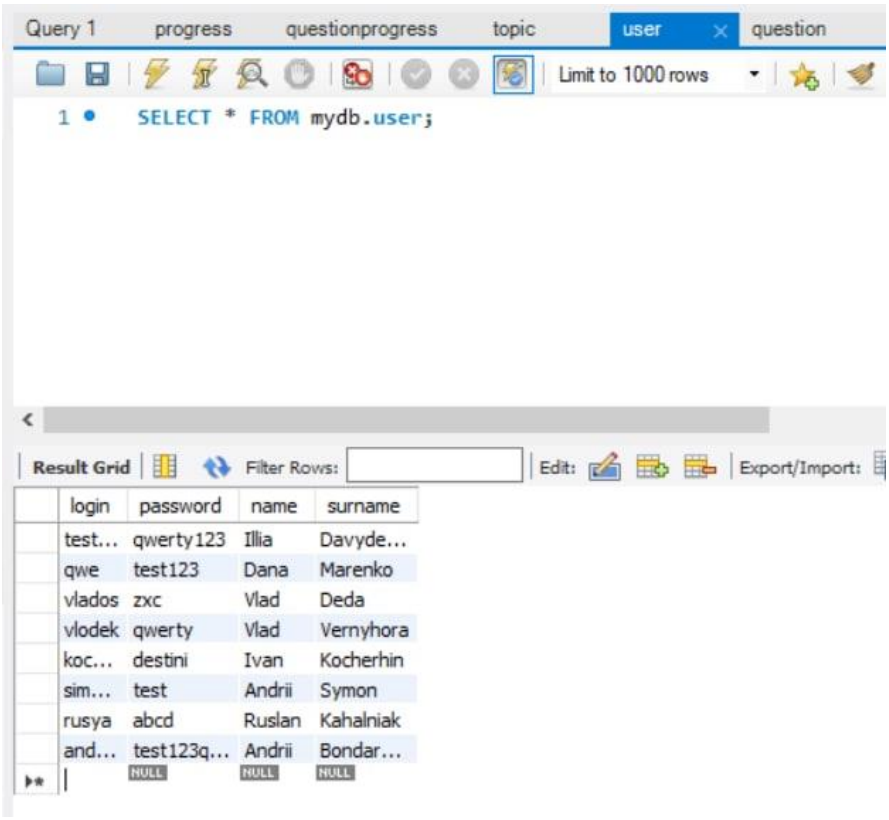


Рисунок 3.2 – SQL-запит для виведення таблиці «user»

Отже, було обрано СУБД для розробки, розроблено та описано базу даних веб-системи для вивчення мови JavaScript.

3.3 Розробка клієнтської частини веб-системи для вивчення мови JavaScript

Клієнтська частина сайту це та частина, яку бачить користувач. Вона відіграє велику роль бо сам від неї формується перше враження про сайт, а вже потім користувач починає думати за його швидкодію та функціонал.

Майже всі веб-сайти створюються по певним правилам, які є стандартом при їх створенні. Наприклад, на всіх сайтах повинен бути «дах», на якому буде логотип сайту і навігація по ньому та «підвал», який буде містити інформацію про розробника, його контакти, посилання на користувацьку інструкцію тощо.

Кожна веб-сторінка поділена на певні блоки контенту, які повинні мати в собі той контент, який від них очікується на перший погляд. Зверху веб-сайту присутній його «дах», тобто місце де розташовані логотип та навігація. Цей блок повинен бути присутнім на всіх сторінках сайту, тому що користувачу необхідно мати доступ до головної навігації по всьому ресурсу. Розроблений «дах» сайту представлено на рисунку 3.3.

Рисунок 3.3 – «Дах» веб-сайту

Оскільки було обрано вільну структуру веб-сайту, на головній сторінці присутні посилання на всі інші сторінки. Вигляд головної сторінки відображено на рисунку 3.4.

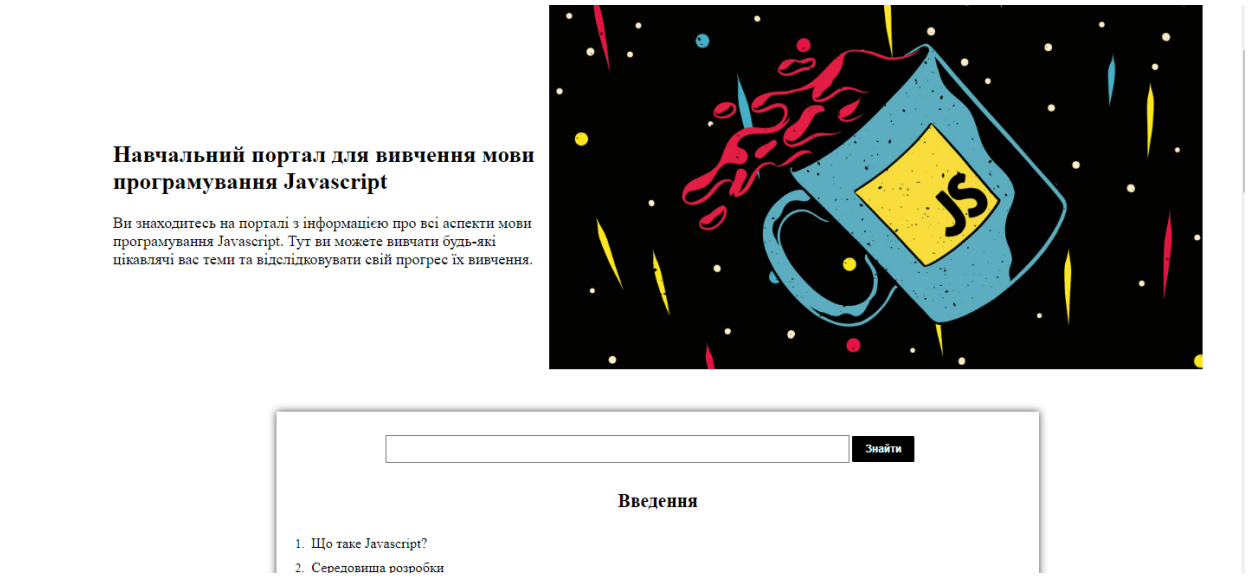


Рисунок 3.4 – Головна сторінка веб-сайту

Клієнтською частиною також вважається функціонал сайту який пов’язаний з відображенням контенту або зміною його виду. На веб-сайті присутній пошук по

темам для вивчення. Для цього користувачу потрібно ввести назву теми, яка його цікавить, і в мапі сайту залишаться лише ті теми, назви яких відповідають набраному тексту.

Приклад роботи пошуку представлено на рисунку 3.5.

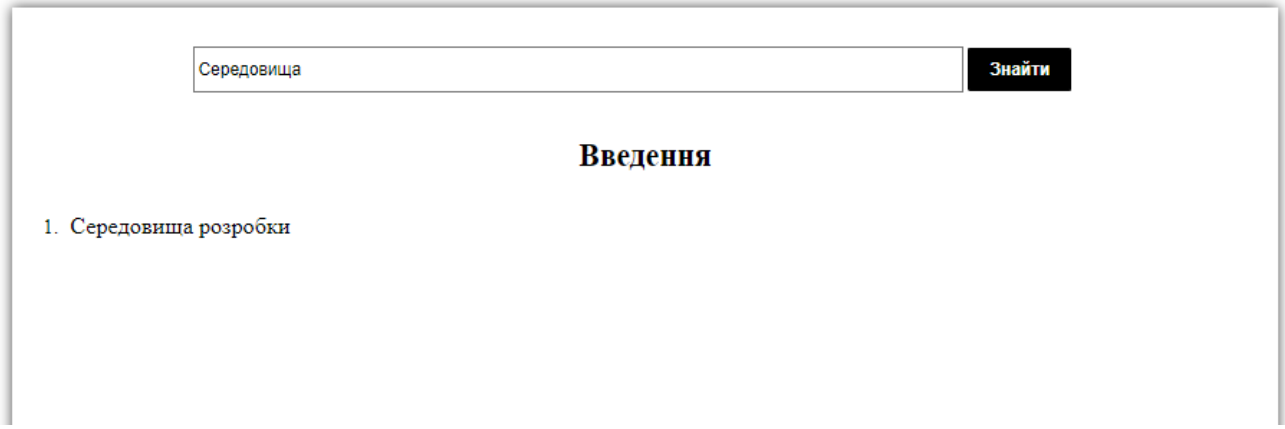


Рисунок 3.5 – Пошук тем по сторінці

Функціонал пошуку тем на сторінці було розроблено мовою JavaScript. Лістинг коду модулю пошуку тем відображено на рисунку 3.6.

```
const searchTopics = document.querySelector('.b-topics_search');

searchTopics.addEventListener('keyup', () => findTopics(searchTopics.value))

function findTopics (inputValue) {
    inputValue = inputValue.toLowerCase();
    const topics = document.querySelectorAll('.b-topics_topic');

    for (const topic of topics) {
        const topicHeading = topic.querySelector('.b-topics_first-level');
        const topicList = topic.querySelector('.b-topics_second-level');
        let hideHeading = false;
        let hiddenChildren = 0;

        changeTopicsView(topicList, inputValue);

        for (const child of topicList.children) {
            if (child.style.display === 'none') {
                hiddenChildren++;
            }
        }

        if (hiddenChildren === topicList.children.length) {
            hideHeading = true;
        }

        if (hideHeading) {
            topicHeading.style.display = 'none';
        } else {
            topicHeading.style.display = 'block';
        }
    }
}

function changeTopicsView (list, inputValue) {
    for (const topic of list.children) {
        if (!topic.innerText.toLowerCase().includes(inputValue)) {
            topic.style.display = 'none';
        } else {
            topic.style.display = 'list-item';
        }
    }
}
```

Рисунок 3.6 – Лістинг коду модулю пошуку тем на сторінці

Коли користувач переходить до сторінки «Аккаунт», перед ним постає вибір авторизуватися чи зареєструватися на веб-сайті. Ця сторінка представлена на рисунку 3.7.

Рисунок 3.7 – Сторінка з вибором входу на веб-сайт

Для стилізування цих кнопок було обрано створити спеціальну змінну з правилами, які можна використовувати повторно без написання однакового коду. Єдине, що в них є різного – це їх позиція на сторінці. Лістинг змінної наведено на рисунку 3.8.

```
1  @mixin button {
2      background-color: $black;
3      color: $white;
4      border-radius: 2px;
5      font-weight: bold;
6      height: 60px;
7      width: 400px;
8      position: relative;
9      top: 200px;
10 }
```

Рисунок 3.8 – Лістинг коду стилів змінної для кнопки

В зележності від натиснутої кнопки, користувача буде направлено на сторінку з формою. Якщо користувач вибере авторизацію, то його буде направлено на сторінку з наступною формою (див. рис. 3.9):

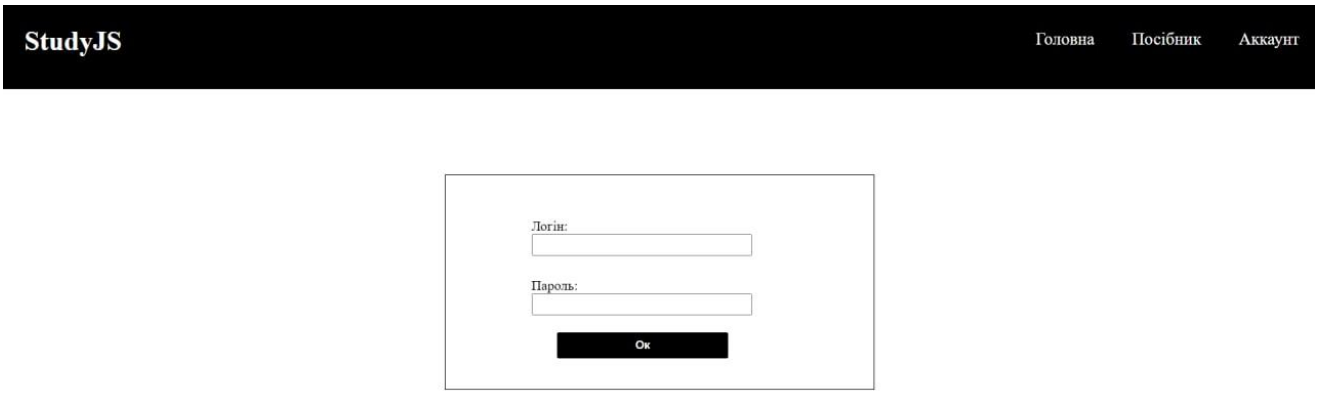


Рисунок 3.9 – Форма авторизації користувача

Якщо ж було обрано форму реєстрації користувача, то вона буде мати наступний вигляд (див. рис. 3.10):

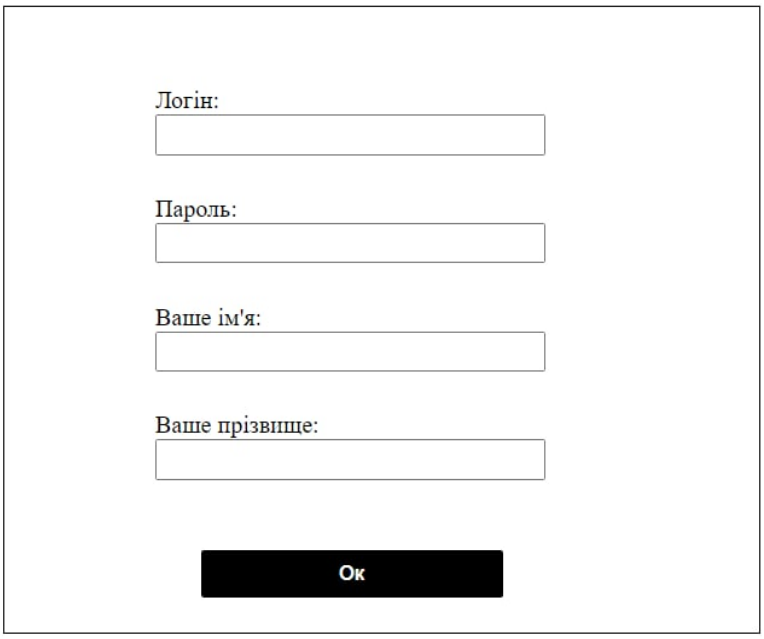


Рисунок 3.10 – Форма реєстрації користувача

Було розроблено блок з задачами після завершення розділу. Приклад задачі після розділу наведено на рисунку 3.11.

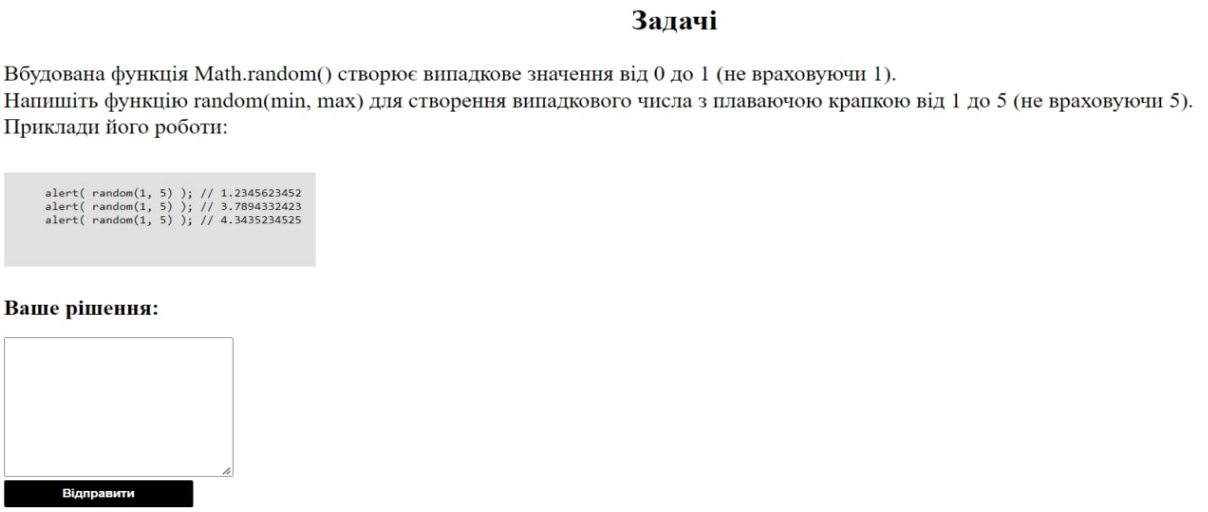


Рисунок 3.11 – Приклад задачі

Було розроблено профіль користувача в якому відображаються теми, які він пройшов та його логін, ім’я і прізвище (див. рис. 3.12);

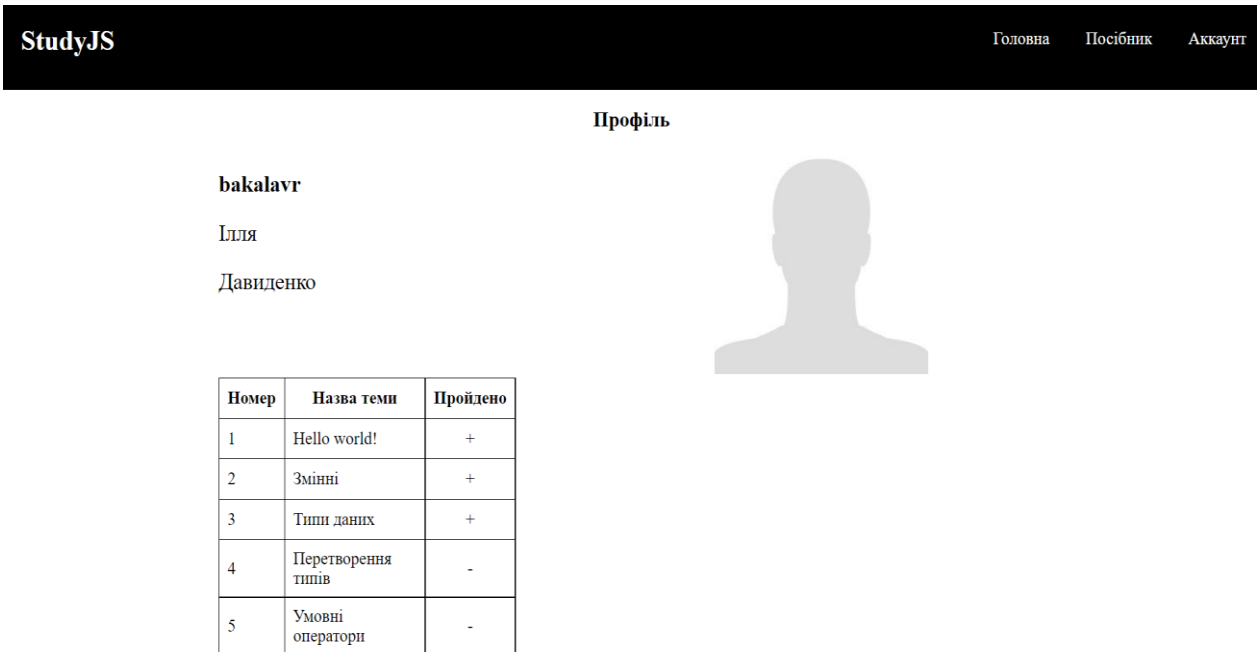


Рисунок 3.12 – Профіль користувача

Отже, було розглянуто розробку клієнтської частини веб-системи для вивчення мову програмування JavaScript.

3.4 Розробка серверної частини веб-системи для вивчення мови JavaScript

Майже кожен веб-сайт повинен бути пов’язаний з сервером. Завдяки цьому зв’язку можна обробляти форми, надіслані з клієнтської частини, перенаправляти користувача на інші сторінки і взагалі мати змогу відвідати веб-сайт з пошукових систем як повноцінний застосунок. Якщо ж в сайту немає сервера то його використання можливе лише за допомогою відкриття локальних файлів, тобто переглядати цей сайт зможе лише його розробник і ті кому він надав доступ до файлів.

Сервер необхідний для будь-якого сайту, бо якщо його немає, то його подальше просування буде неможливим. Мета кожного сайту це дати щось користувачу, будь то інформація, браузерна гра, відео контент тощо. Для деяких ресурсів він необхідний лише для того щоб відвідувати сторінки в мережі інтернет, а в деяких його використання полягає ще й у тому, що необхідно обробляти форми, виконувати обчислення, тощо.

Для зв’язку між клієнтською та серверною частинами було обрано HTTP протоколу передачі даних. HTTP – це прикладний протокол передачі даних у мережі. Наразі його використовують для отримання інформації з веб-сайтів. Протокол HTTP заснований на використанні технології «клієнт-сервер»: клієнт, що відправляє запит, є ініціатором з’єднання; сервер, який отримує запит, виконує його і відправляє клієнту результат [14].

Головний файл містить створення та підключення до серверу. В ньому об’являються всі можливі переходи до сторінок і сам запуск сервера. На рисунку 3.13 представлено лістинг коду створення та запуску локального серверу на порту 3001.

```
const express = require('express');
const http = require('http');
const app = express();
const server = http.createServer(app);

const authorization = require('./dist/js/authorization');
const registration = require('./dist/js/registration');
const topic = require('./dist/js/topic');
const profile = require('./dist/js/profile');
const port = 3001;

app.use('/authorization', authorization);
app.use('/registration', registration);
app.use('/topic', topic);
app.use('/profile', profile);


app.use(express.static(__dirname + '/dist'));
app.use(express.static(__dirname + '/images'));


app.get('/', function(req, res){
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
    res.setHeader('Access-Control-Allow-Credentials', 'true');
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.sendFile(__dirname + '/index.html');
});

server.listen(port, () => {
    console.log(`Listening on ${port}`);
});
```

Рисунок 3.13 – Лістинг коду створення локального серверу

На початку файлу йде підключення необхідних компонентів Node.js, завдяки яким і відбувається створення серверу. Після об’явлення цих змінних йде об’явлення файлів контролерів як змінних. Це необхідно для того, щоб сервер міг використовувати шлях до них, і перенаправляти користувача на посилання які будуть містити назву контролера в ньому. Ще нижче було об’явлено шлях до папок які будуть використані для статичного контенту, тобто HTML сторінок, таблиць стилів та картинок. Без цього об’явлення сервер не знав би звідки йому брати ці файли, а спробу взяти їх по абсолютному шляху до файлу закінчились би неможливістю відобразити цей контент.

Серверу необхідно буде робити запити до бази даних, а цьому може завадити CORS. Cross-Origin Resource Sharing – механізм, що використовує додаткові заголовки HTTP, щоб дати можливість агенту користувача отримувати дозволи на доступ до вибраних ресурсів із сервера на джерелі (домені), відмінному від того, що сайт використовує в даний момент. Говорять, що агент користувача робить запит з іншого джерела (cross-origin HTTP request), якщо джерело поточного

документа відрізняється від ресурсу домену, протоколом або портом [15]. По замовчуванню неможливо отримати дані з веб-сайту з доменом який відрізняється від того, на якому він знаходиться, а так як сервер на знаходиться безпосередньо в одному місці з його клієнтською частиною, то дати на нього запит не вийде без заголовків які дозволять цього уникнути. Для цього було додано заголовки за допомогою res.setHeader(), в які прописується інформація про запити, які буде дозволено на сервері та домени з яких можна зробити цей запит.

І в самому кінці створюється локальний сервер. Він запускається на порті, який було об'явлено в змінній вверху файлу, тобто на 3001.

Розглянемо блок реєстрації користувача. В нього користувач заходить одразу як переходить по посиланню до сторінки реєстрації і перебуває там до введення своїх даних і відправки форми. Для цього було написано контролер лістинг коду якого присутній на рисунку 3.14.

```
router.get('/', function(req, res){
    res.sendFile(path.join(__dirname, '..', 'pages', 'registration.html'));
})

let connection = mysql.createConnection({
    database: 'studyJsDB',
    host: "localhost",
    port: 3306,
    user: "root",
    password: "D@rkScr0lI$52143"
});

router.post('/', urlencodedParser, function (req, res){
    if (!req.body) {
        return res.sendStatus(400);
    }
    const userData = req.body;

connection.query(`SELECT * FROM studyJSdb.users WHERE login = ${userData.login}`,
    function(err, results, fields) {
        if (err) {
            res.send(path.join(__dirname, '..', 'pages', 'loginfail.html'));
        }
    });
})

connection.query("INSERT INTO user(login,password,name,surname) VALUES (?, ?, ?, ?)",
[req.body.login, req.body.password, req.body.name, req.body.surname], function(err, results) {
    if (err) {
        console.log(err);
    } else {
        console.log('Аккаунт створено');
    }

    res.send(path.join(__dirname, '..', 'pages', 'regsuccess.html'));
})
```

Рисунок 3.14 – Лістинг контролера реєстрації користувача

На початку роботи користувачу надсилається сторінка реєстрації, на ній він заповняє поля і відправляє її для подальшої обробки. В цей же час сервер підключається до бази даних. Після цього програма переходить до обробки POST запиту, тобто запиту на додавання контенту. В ньому дається запит на базу даних для отримання користувача з логіном, який було введено при реєстрації. Це робиться для того, щоб перевірити чи існує вже користувач с таким логіном, бо логін є ключем в таблиці і декілька користувачів не може мати однакове його значення. В випадку співпадіння логінів, користувач перенаправляється на сторінку на якій йому повідомляється про неможливість реєстрації і пропонується заповнити поля ще раз. В іншому випадку до бази даних дається запит на створення нового користувача і сайт перенаправляє його на сторінку з повідомленням про успішну реєстрацію.

Сторінки з вивченням тем по мові програмування JavaScript відправляються користувачу за допомогою передачі їх назв в посилання. Передана змінна в посилання називається рядком запиту і її значення можна отримати через JavaScript код. Він передається за допомогою додання знаку питання, назви змінної і її значення через дорівнює. Завдяки цьому немає необхідності прописувати в контролері шлях до кожного файлу щоб відобразити його, достатньо лише передавати параметри в посилання. Лістинг коду цього підходу відображено на рисунку 3.15.

```
router.get('/', function(req, res) {
    res.sendFile(path.join(__dirname, '..', 'pages', `${req.query.topicName}.html`));
})
```

Рисунок 3.15 – Передача сторінки користувачу

Отже, було розглянуто розробку серверної частини веб-системи для вичення мови JavaScript.

3.5 Висновки

В третьому розділі було проаналізовано і обґрунтовано вибір засобів для реалізації веб-системи для вивчення мови програмування JavaScript. Для цього було обрано такі засоби: HTML, CSS, JavaScript, Node.js. Розроблено базу даних веб-системи. Було розглянуто розроблену клієнтську частину веб-системи. Розроблено серверну частину веб-системи.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування розробленої веб-системи

Перед завершенням розробки будь-якого продукту необхідно його протестувати щоб виявити чи немає в ньому недоліків, які можуть викликати в користувачів неприємний досвід використання сайту, програмного забезпечення, тощо.

Тестування – це перевірка відповідності між реальною поведінкою програми та її очікуваним поведінкою на кінцевому наборі тестів, обраному певним чином. Його цілями є виявлення та усунення дефектів і підвищення впевненості в рівні якості [14].

Тестування веб-сайту відрізняється від тестування іншого програмного забезпечення і включає в себе багато різних аспектів. В такому тестуванні необхідно багато часу витратити на перевірку оформлення і логічності в інтерфейсі.

На початку тестування веб-сайту, необхідно в першу чергу подивитися на розташування блоків з контентом. Вони мають бути достатнього розміру, щоб користувач зміг побачити все, що всередині них. Також всі блоки повинні бути в логічній послідовності один за одним, щоб користувач на відволікався на намагання зрозуміти де буде знаходитися контент. Необхідно перевірити чи відображаються всі зображення, а якщо якесь з них відсутнє, на його місці повинен знаходитися текст, який описує, що на ньому зображено.

Для аналізу було обрано три методики функціонального тестування, а саме тестування методами білого, сірого та чорного ящика.

При тестуванні білого ящика (прозорого ящика) розробник тесту має доступ до вихідного коду програм та може писати код, який пов'язаний з бібліотеками тестованого програмного забезпечення. Це типово для модульного тестування, при якому тестуються лише окремі частини системи. Такий вид тестування

забезпечується тим, що компоненти конструкції – працездатні та стійкі, до певної міри. При тестуванні білого ящика використовуються метрики покриття коду або мутаційне тестування.

Цей метод має такі переваги:

− присутня можливість аналізу правильності внутрішніх структур даних;

− тестування може проводитися без створеного інтерфейсу, тобто на ранніх етапах.

Недоліками є наступне:

− для проведення тестування потрібно мати навички програмування для розуміння коду.

При тестуванні чорного ящика, тестувальник має доступ до програми тільки через ті ж інтерфейси, що й замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншому комп'ютеру або іншому процесу підключитися до системи для тестування. Як правило, тестування чорного ящика ведеться з використанням специфікацій або інших документів, що описують вимоги до системи. Зазвичай в даному виді тестування критерій покриття складається з покриття структури вхідних даних, покриття вимог та покриття моделі (у тестуванні на основі моделей).

Переваги є такими:

− присутня можливість як функціонального так і нефункціонального тестування;

Недоліками є наступне:

− важко створити тест-кейси без специфікації;

− тестується дуже обмежена кількість шляхів виконання програми.

При тестуванні сірого ящика розробник тесту має доступ до вихідного коду, але при безпосередньому виконанні тестів доступ до коду, як правило, не потрібен. Метод сірого ящика поєднує в собі методи білої та сірої скриньки. Тобто, внутрішній устрій програми нам відомо лише частково. Передбачається,

наприклад, доступ до внутрішньої структурі та алгоритмам роботи ПО для написання максимально ефективних тест-кейсів, але саме тестування проводиться за допомогою техніки чорного ящика, тобто, з позиції користувача [16].

Метод сірого ящика має такі переваги:

− включає в себе всі переваги “чорного” та “білого” ящиків;

− максимальна ефективність тест-кейсів.

Недоліками є такими:

− неможливість тестування всіх потоків введення та виведення.

Отже, було проведено аналіз трьох методів тестування: “білого ящика”, “чорного ящика” та “сірого ящика”, і вирішено обрати для тестування метод “сірого ящика”, адже алгоритми роботи додатку було описано частково і він є дуже зручним для створення тест-кейсів. Було проведено тестування перевірки правильності виконання задач. Задача з заповненим полем рішення від користувача представлена на рисунку 4.1.

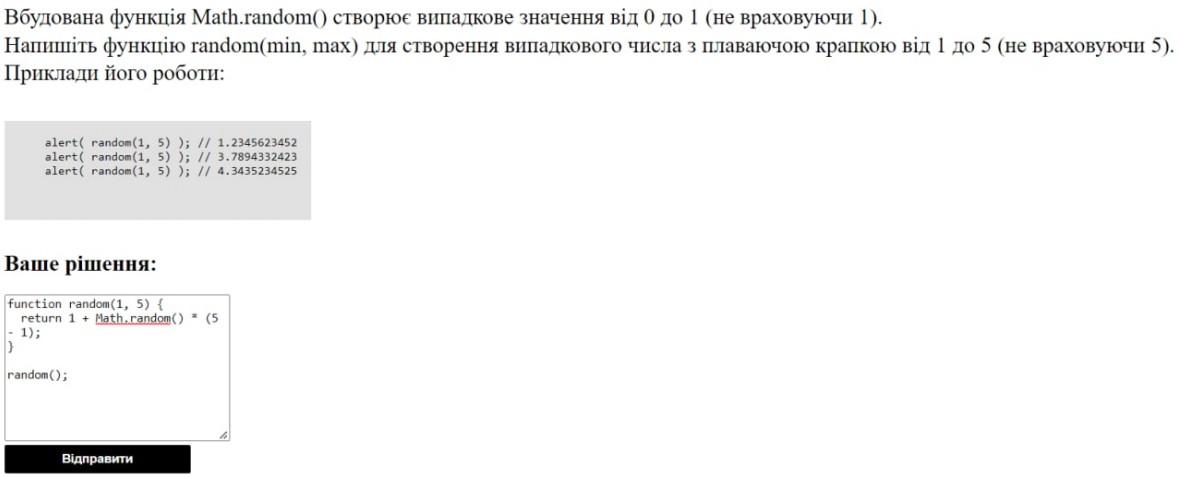


Рисунок 4.1 – Задача з введеним від користувача рішенням

Для тестування і додання результату до статистики користувача було написано JavaScript код який представлено на рисунку 4.2.

```
function checkMinMax (userSolution, user) {
    let isPassed = false;

    try {
        if (eval(userSolution) < 5 && eval(userSolution) > 1) {
            alert('Ваше рішення підходить, вітаємо!')
        }
    } catch (err) {
        alert('Код не виконується, спробуйте ще раз')
    }

    if (isPassed) {
        user.passedTasks.set('minmax', 'true');
    } else {
        user.passedTasks.set('minmax', 'false');
    }
}
```

Рисунок 4.2 – Лістинг коду для перевірки рішення

Код в прикладі було написано вірно, тому при натисканні кнопки відправити повинне з’явитися повідомлення «Ваше рішення підходить, вітаємо!». Результат тестування представлено на рисунку 4.3.

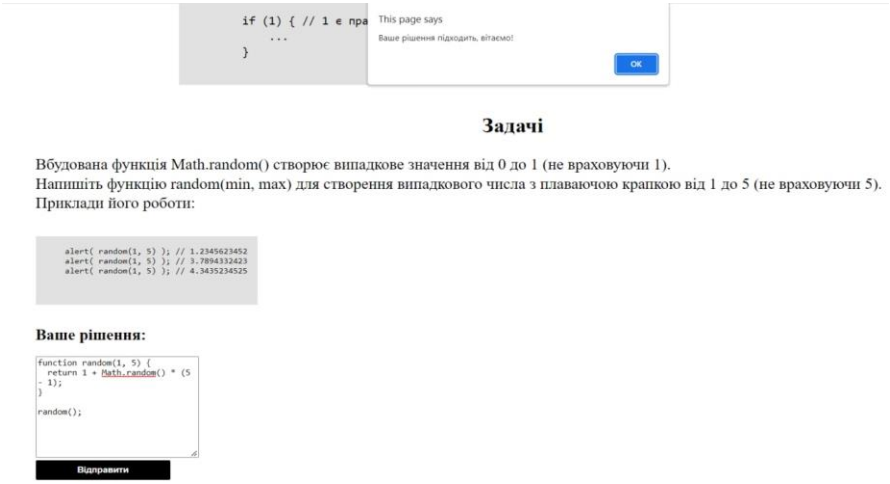


Рисунок 4.3 – Результат тестування

Після цього було проведено тестування з неправильно введеними даними, результат тестування наведено на рисунку 4.4.

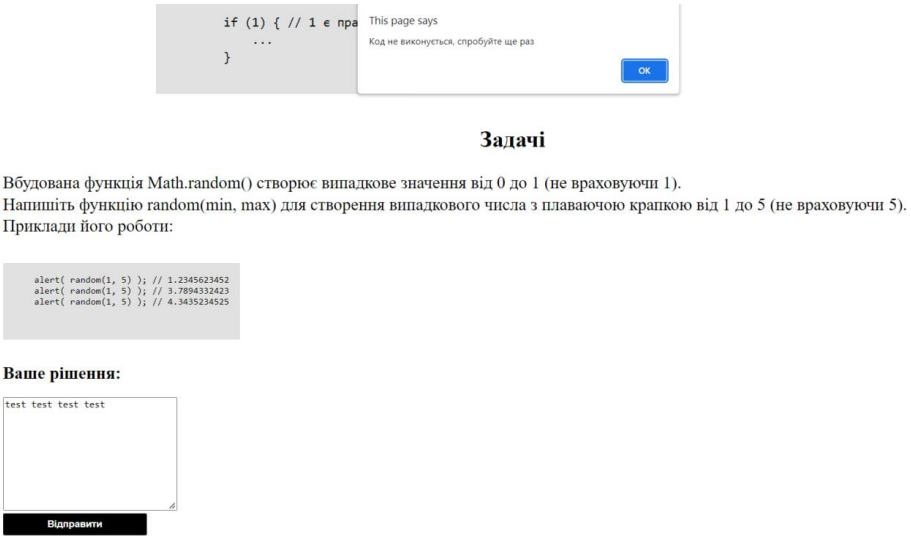


Рисунок 4.4 – Результат тестування з неправильним введеним рішенням

Отже, було проведено тестування блоку перевірки задач після розділів.

4.2 Розробка інструкції користувача

Інструкція користувача – це так званий гід по речам, які вона описує. У цьому випадку буде проводитись опис саме веб-системи для вивчення мови програмування JavaScript.

Для того, щоб зайти на сайт, потрібно натиснути на елемент навігаційної панелі «Акаунт». На рисунку 4.5 зображено вибір при вході в акаунт.

Рисунок 4.5 – Вхід в акаунт

Обравши варіант «Реєстрація» відкривається сторінка з анкетою для створення облікового запису (див. рис. 4.6).

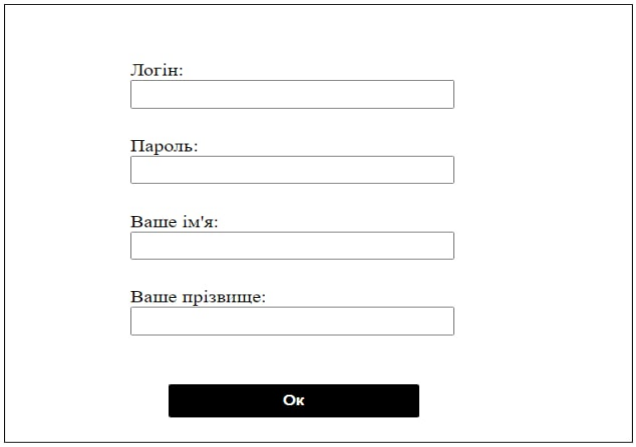


Рисунок 4.6 – Реєстрація у веб-системі

Якщо користувач введе свої дані при реєстрації, і в базі даних не буде існувати користувача з таким логіном, то його буде додано до бази даних. Приклад введених користувачем даних представлено на рисунку 4.7.

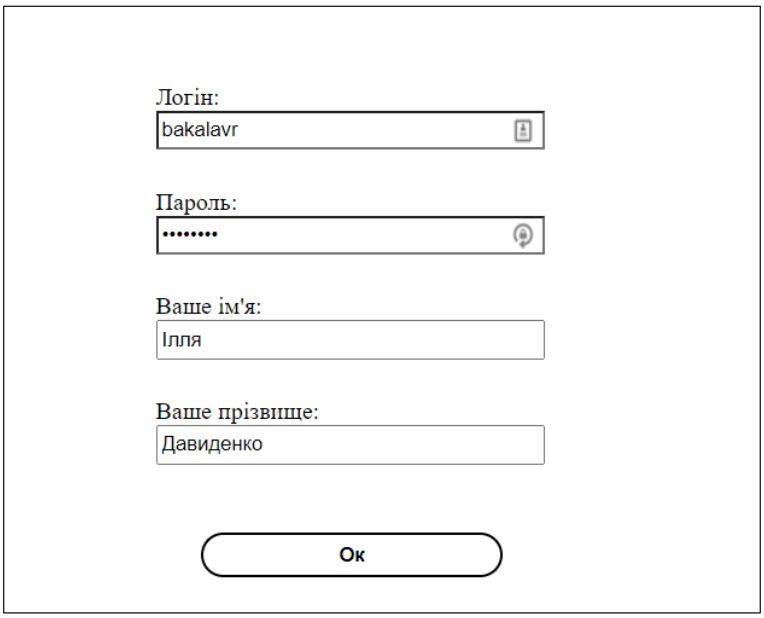
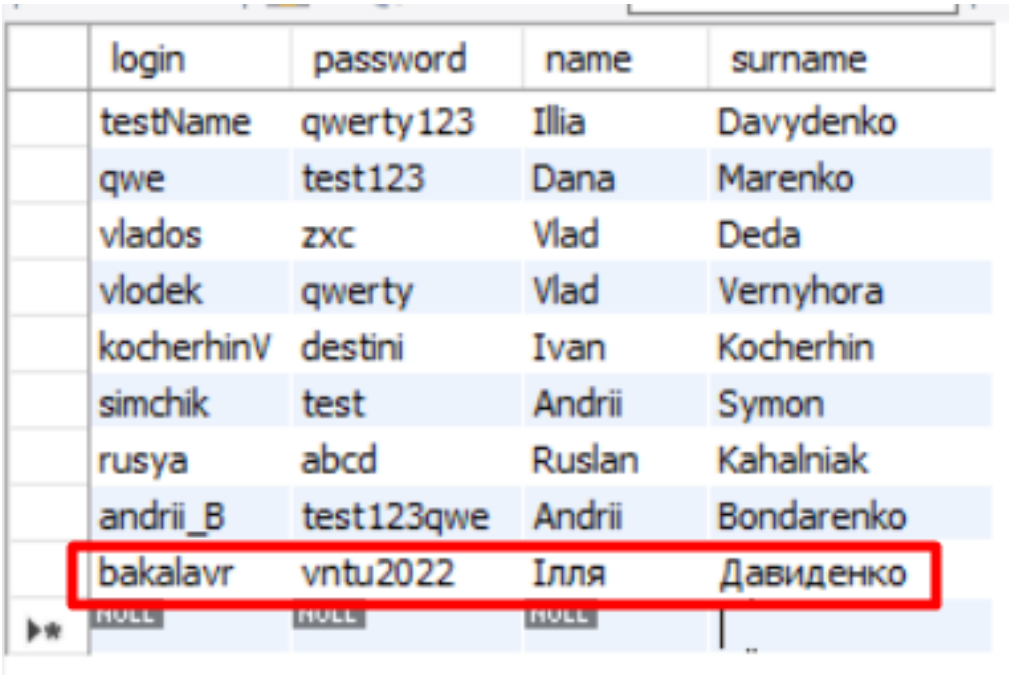


Рисунок 4.7 – Форма реєстрації з введеними даними

Після того, як юзер натисне кнопку «Ок», його аккаунт буде додано до бази даних (див. рис. 4.8).



login	password	name	surname
testName	qwerty123	Illia	Davydenko
qwe	test123	Dana	Marenko
vlados	zxc	Vlad	Deda
vlodek	qwerty	Vlad	Vernyhora
kocherhinV	destini	Ivan	Kocherhin
simchik	test	Andrii	Symon
rusya	abcd	Ruslan	Kahalniak
andrii_B	test123qwe	Andrii	Bondarenko
bakalavr	vntu2022	Ілля	Давиденко
NULL	NULL	NULL	

Рисунок 4.8 – Доданий до бази даних користувач

Обравши варіант «Авторизація» відкривається сторінка, на якій потрібно ввести логін та пароль існуючого облікового запису (див. рис. 4.9).

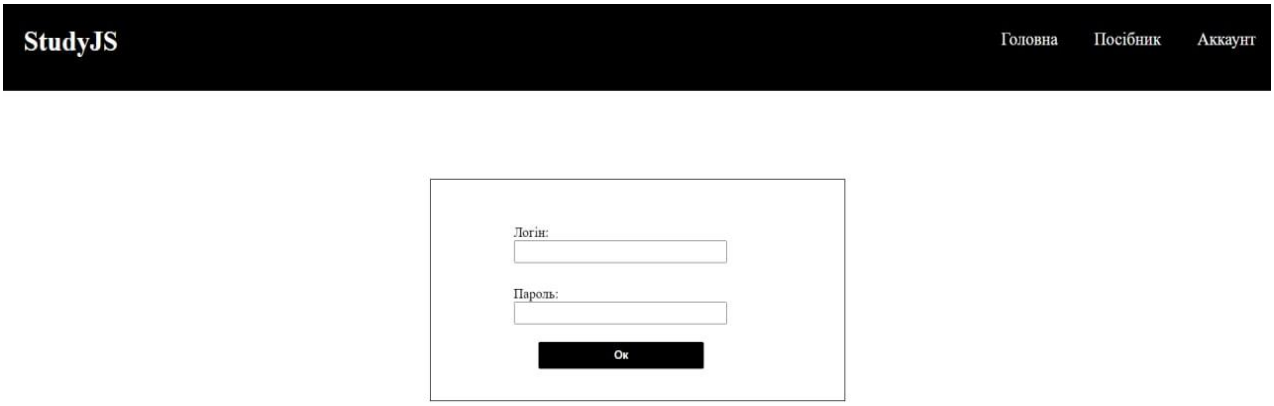


Рисунок 4.9 – Авторизація у веб-системі

Для пошуку на сайті використовується поле пошуку. Достатньо написати назву теми, яку потрібно знайти. На рисунку 4.10 показано пошук по веб-сайту.

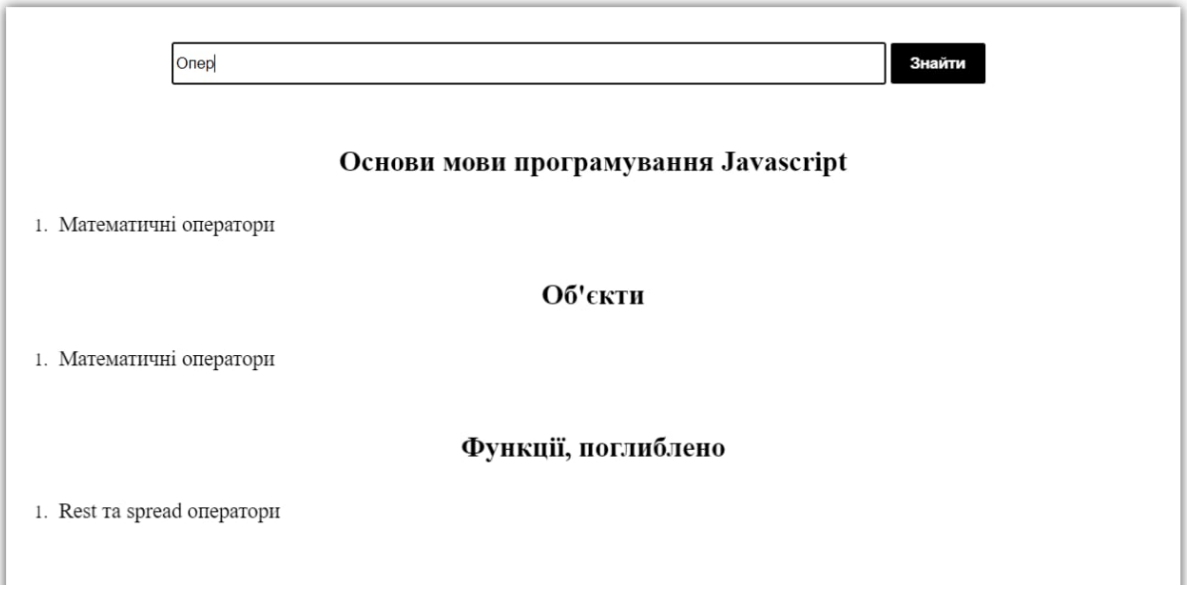


Рисунок 4.10 – Пошук по веб-сайту

Після знаходження потрібного матеріалу, потрібно натиснути на його назву. Далі відкриється сторінка з інформацією про мову програмування JavaScript, яку можна побачити на рисунку 4.11.

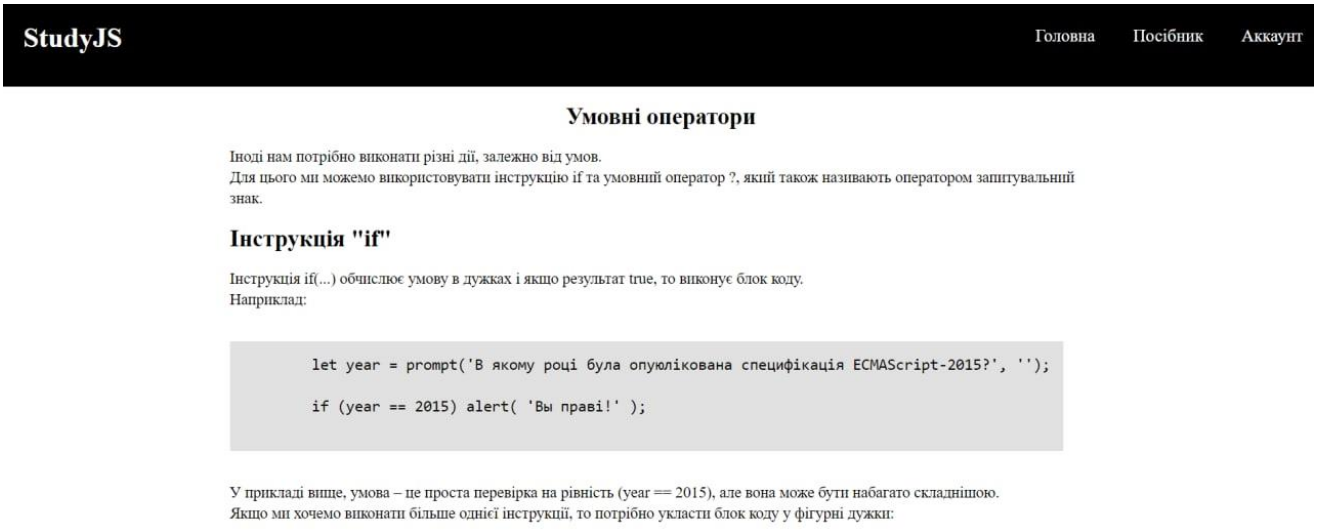


Рисунок 4.11 – Сторінка освітнього матеріалу

Після ознайомлення з матеріалом користувачу потрібно виконати певні задачі, описані в кінці ознайомлення із темою, щоб її закріпити. На рисунку 4.12 показано задачу та поле для введення її розв’язку.

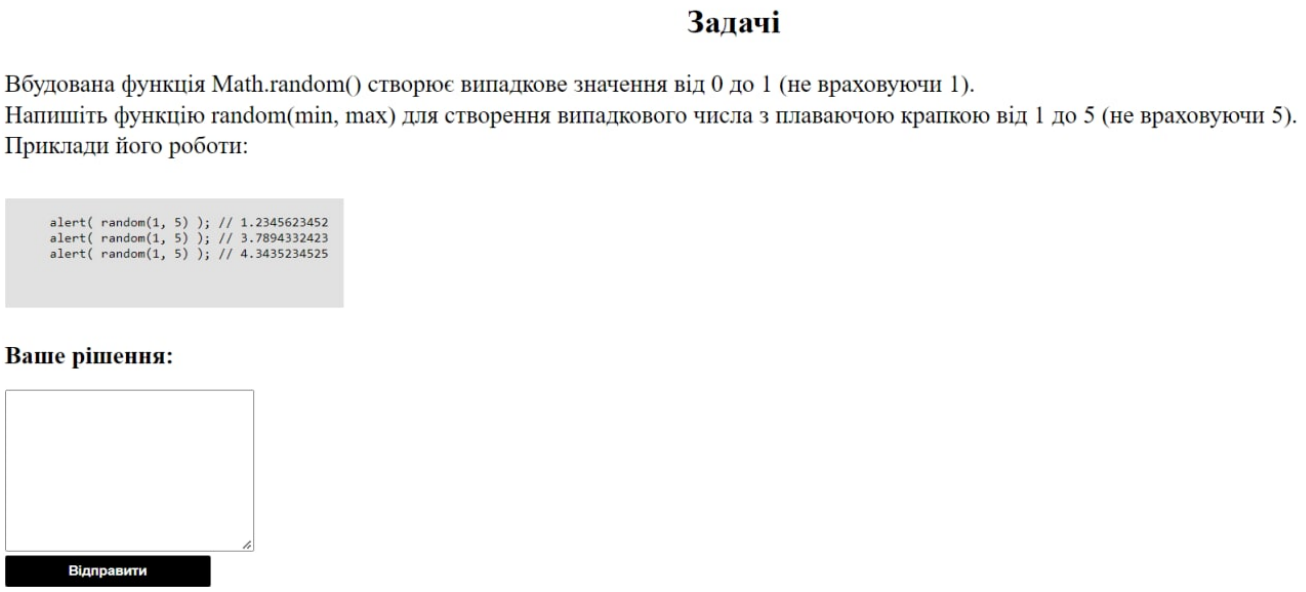


Рисунок 4.12 – Задача для закріплення знань

Для того, аби пройти тему, потрібно правильно розв’язати поставлену задачу. Розв’язок приймається у вигляді лістинга коду на мові програмування JavaScript. При правильному рішенні тема зараховується як пройдена.

Отже, було розроблено інструкцію користувача для веб-системи для вивчення мови програмування JavaScript.

4.3 Висновки

У четвертому розділі було проаналізовано методи тестування та вирішено обрати для тестування метод «сірого ящика», адже алгоритми роботи додатку було описано частково і він є дуже зручним для створення тест-кейсів. Проведено тестування веб-системи. Розроблено інструкцію користувача для веб-системи.

ВИСНОВКИ

Проаналізовано сучасний стан галузі вивчення мови програмування JavaScript. Провівши порівняльний аналіз аналогів, було зроблено висновок, що дана розробка є актуальною та задовільнить потреби користувачів, які бажають увійти в сферу інформаційних технологій або бажають покращити свої знання. Обрано метод розробки веб-системи власноручно з самого початку через його суттєві переваги. Було визначено задачі дослідження, які потрібно виконати для успішної розробки на тему «Розробка веб-системи для вивчення мови програмування JavaScript».

Проведено аналіз структурування навчальних порталів та його типи. Було розроблено структуру інтерфейсу веб системи для вивчення мови програмування JavaScript. Розроблено навчальну програму та її дизайн з темами для вивчення мови програмування JavaScript. Було розглянуто головні алгоритми роботи веб-системи та створено блок-схеми з їх описом.

Проаналізовано і обґрунтовано вибір засобів для реалізації веб-системи для вивчення мови програмування JavaScript. Для цього було обрано такі засоби: HTML, CSS, JavaScript, Node.js. Розроблено базу даних веб-системи. Було розглянуто розроблену клієнтську частину веб-системи. Розроблено серверну частину веб-системи.

Проведено тестування розробленої веб-системи для вивчення мови JavaScript, у результаті якого було зроблено висновок, що програма працює правильно. Розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Давиденко І. С. Бабюк Н. П. Аналіз мови програмування JavaScript. / LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця: ВНТУ, 2022. 2 с.

2. MDN Web Docs. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/

3. W3Schools. [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/

4. JavaScript.info. [Електронний ресурс]. – Режим доступу: https://uk.JavaScript.info/

5. Рональд Л. Алгоритми. Побудова та аналіз: пер. з англ. Красиків І. / Л. Рональд – Санкт-Петербург: Вільямс, 2020. – 1328 с.

6. Роббінс Д. HTML5: кишеньковий довідник 5-е видання. / Д. Роббінс – Київ: Діалектика, 2015. 192 с.

7. Мейер Е., Уейл Е. CSS повний довідник. / Е. Мейер, Е. Уейл – Київ: Діалектика, 2017. 1088 с.

8. Уроки SASS / SCSS. [Електронний ресурс]. – Режим доступу: https://itproger.com/course/sass

9. Webpack: керівництво для початківців. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/514838/

10. Резіг Д. Секреты JavaScript ниндзя. / Д. Резіг – Санкт-Петербург: Диалектика 2019. – 544 с.

11. Що таке Node.js. [Електронний ресурс]. – Режим доступу: https://uk.theastrologypage.com/node-js

12. Романюк О. Н. Організація баз даних і знань [Текст] : навчальний посібник / О. Н. Романюк, Т. О. Савчук. - Вінниця : УНІВЕРСУМ-Вінниця, 2003. – 217 с.

13. MySQL Workbench. [Електронний ресурс]. – Режим доступу: https://www.mysql.com/products/workbench/

14. Гурлі Д. HTTP: повний посібник. / Д. Гурлі – Массачусетс: О'Рейлі Медіа 2002. – 656 с.

15. CORS. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTTP/CORS

16. Ляхов О.Л. Методи тестування і оцінки якості програмного забезпечення. / О. Л. Ляхов, О.О. Бородіна – Полтава: ПолтНТУ, 2015. – 372 с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

«25» березня 2022 р.

Технічне завдання

на бакалаврську дипломну роботу

«Розробка соціальної мережі для онлайн-навчання користувачів**»**

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

к.т.н., доцент кафедри ПЗ Бабюк Н. П

«____»_____2022 р.

Виконав:

студент гр. 2ПІ-18б Давиденко І. С.

«____»_____2022 р.

Вінниця – 2022 року

1 Найменування та галузь застосування.

Бакалаврська робота: «Розробка веб-системи для вивчення мови програмування JavaScript».

Сфера застосування – навчальна (інформаційні технології).

2 Підстава для розробки.

Підставою для розробки бакалаврської дипломної роботи є рішення засідання кафедри програмного забезпечення (протокол № 12 від "07" лютого 2022 року).

3 Мета та призначення розробки.

Метою дослідження є підвищення продуктивності вивчення мови програмування Javscript шляхом створення веб-системи для її вивчення.

Призначення роботи – розробка програмного продукту для вивчення мови програмування JavaScript.

4 Вихідні дані для проведення НДР.

Вихідні дані для розробки є індивідуальне завдання на бакалаврську дипломну роботу на розробку веб-системи для вивчення мови програмування JavaScript.

5 Технічні вимоги.

Методи вивчення мови програмування JavaScript; архітектура веб-додатку; формування матеріалу для вивчення; масив тасків для перевірки знань; сформована веб-система.

6 Конструктивні вимоги.

Веб-додаток має відповідати ергономічним та технічним вимогам. текстова та графічна документація повинна відповідати стандартам України.

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

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинг програми.

8 Вимоги до рівня уніфікації та стандартизації.

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9 Стадії та етапи розробки.

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач дослідження	18.02 – 03.04	Виконано
2	Розробка структури та алгоритмів програмного продукту	04.04 – 29.04	Виконано
3	Розробка модулів програмного продукту	30.04 – 20.05	Виконано
4	Тестування програми	20.05 – 29.05	Виконано
5	Оформлення матеріалів до захисту БДР	29.05 – 01.06	Виконано

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється корегування бакалаврської дипломної роботи

Додаток Б – ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка веб-системи для вивчення мови програмування JavaScript

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Бабюк Н.П.

Оригінальність	91.6%
Схожість	8.4%

Аналіз звіту подібності

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

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

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

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Давиденко І.С.

Керівник роботи _____ Бабюк Н.П.

Додаток Б – Лістинг програми

server.js

```
const express = require('express');
const http = require('http');
const app = express();
const server = http.createServer(app);

const authorization = require('./dist/js/authorization');
const registration = require('./dist/js/registration');
const topic = require('./dist/js/topic');
const profile = require('./dist/js/profile');
const port = 3001;

app.use('/authorization', authorization);
app.use('/registration', registration);
app.use('/topic', topic);
app.use('/profile', profile);


app.use(express.static(__dirname + '/dist'));
app.use(express.static(__dirname + '/images'));


app.get('/', function(req, res){
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS,
PUT, PATCH, DELETE');
    res.setHeader('Access-Control-Allow-Credentials', 'true');
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.sendFile(__dirname + '/index.html');
});
```

```
    server.listen(port, () => {
        console.log(`Listening on ${port}`);
    });

    Authorization.js
    const express = require('express');
    const router = express.Router();
    const urlencodedParser = express.urlencoded({extended: false});
    const path = require('path');
    const mysql = require('mysql');

    let connection = mysql.createConnection({
        database: 'studyJsDB',
        host: "localhost",
        port: 3306,
        user: "root",
        password: "D@rkScr0lI$52143"
    });

    router.get('/', function(req, res){
        res.sendFile(path.join(__dirname,          '..',          'pages',
'authorization.html'));
    })

    router.post('/', urlencodedParser, function (req, res){
        if (!req.body) {
            return res.sendStatus(400);
        }
        const userData = req.body;
```

```
        connection.query(`SELECT * FROM studyJSdb.users WHERE login =
${userData.login}`,
        function(err, results, fields) {
            if (err) {
                res.send(path.join(__dirname,        '..',        'pages',
'loginfail.html'));
            }

            results = JSON.stringify(results);

            if (results.password === userData.password) {
                document.cookie = `login=${userData.login}`;
                res.send(path.join(__dirname,        '..',        'pages',
'success.html'));
            } else {
                res.send(path.join(__dirname,        '..',        'pages',
'loginfail.html'));
            }

            router.get('/', function(req, res) {
            res.send(results);
            })
        });
    })

    module.exports = router;

    registration.js
    const express = require('express');
    const res = require('express/lib/response');
    const router = express.Router();
    const urlencodedParser = express.urlencoded({extended: false});
```

```
    const path = require('path');
    const mysql = require('mysql');

    router.get('/', function(req, res){
                res.sendFile(path.join(__dirname,     '..',     'pages',
'registration.html'));
    })

    let connection = mysql.createConnection({
        database: 'studyJsDB',
        host: "localhost",
        port: 3306,
        user: "root",
        password: "D@rkScr0lI$52143"
    });

    router.post('/', urlencodedParser, function (req, res){
        if (!req.body) {
            return res.sendStatus(400);
        }
        const userData = req.body;

    connection.query(`SELECT   *   FROM   studyJSdb.users   WHERE   login   =
${userData.login}`,
        function(err, results, fields) {
            if (err) {
                            res.send(path.join(__dirname,   '..',   'pages',
'loginfail.html'));
            }
        });
    })
```

```
    connection.query("INSERT INTO user(login,password,name,surname) VALUES
(?, ?, ?, ?)",
    [req.body.login, req.body.password, req.body.name, req.body.surname],
function(err, results) {
        if (err) {
            console.log(err);
        } else {
            console.log('Аккаунт створено');
        }

        res.send(path.join(__dirname, '..', 'pages', 'regsuccess.html'));
    })

    module.exports = router;

    script.js
    const searchTopics = document.querySelector('.b-topics_search');

    searchTopics.addEventListener('keyup',              ()              =>
findTopics(searchTopics.value))

    function findTopics (inputValue) {
        inputValue = inputValue.toLowerCase();
        const topics = document.querySelectorAll('.b-topics_topic');

        for (const topic of topics) {
              const  topicHeading  =  topic.querySelector('.b-topics_first-
level');
               const  topicList  =  topic.querySelector('.b-topics_second-
level');
            let hideHeading = false;
            let hiddenChildren = 0;
```

```
        changeTopicsView(topicList, inputValue);

        for (const child of topicList.children) {
            if (child.style.display === 'none') {
                hiddenChildren++;
            }
        }

        if (hiddenChildren === topicList.children.length) {
            hideHeading = true;
        }

        if (hideHeading) {
            topicHeading.style.display = 'none';
        } else {
            topicHeading.style.display = 'block';
        }
    }
}

function changeTopicsView (list, inputValue) {
    for (const topic of list.children) {
        if (!topic.innerText.toLowerCase().includes(inputValue)) {
            topic.style.display = 'none';
        } else {
            topic.style.display = 'list-item';
        }
    }
}
```

index.html

```
<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
           <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
        <link rel="stylesheet" href="style.css">
        <title>BDR</title>
    </head>
    <body>
    <header class="b-header">
    <h1 class="b-header_logo">
        StudyJS
    </h1>
    <nav>
        <ul class="b-header_nav">
             <li class="b-header_nav-item"><a href="index.html" class="b-
header_nav-link">Головна</a></li>
            <li class="b-header_nav-item"><a href="#" class="b-header_nav-
link">Посібник</a></li>
              <li  class="b-header_nav-item"><a  href="pages/account.html"
class="b-header_nav-link">Аккаунт</a></li>
        </ul>
    </nav>
    </header>
    <main class="b-main">
    <section class="b-start">
        <div>
            <h2 class="b-start_heading">
               Навчальний портал для вивчення мови програмування JavaScript
```

```
            </h2>
            <p class="b-start_text">
                 Ви знаходитесь на порталі з інформацією про всі аспекти мови програмування JavaScript. Тут ви можете
                вивчати будь-які цікавлячі вас теми та відслідковувати свій прогрес їх вивчення.
            </p>
        </div>
        <div>
            <img src="images/js_cup.jpg" alt="" class="b-start_img">
        </div>
    </section>

    <section class="b-topics">
        <input type="text" class="b-topics_search">
        <button class="b-topics_search-btn">Знайти</button>
        <div class="b-topics_map">
            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Введення
                </h2>
                <ol class="b-topics_second-level">
                        <li class="b-topics_second-level-item"><a href="#" class="b-topics_second-level-link">Що таке JavaScript?</a></li>
                        <li class="b-topics_second-level-item"><a href="#" class="b-topics_second-level-link">Середовища розробки</a></li>
                        <li class="b-topics_second-level-item"><a href="#" class="b-topics_second-level-link">Сфери використання</a></li>
                </ol>
            </div>
        </div>
```

```
        <div class="b-topics_map">
            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Основи мови програмування JavaScript
                </h2>
                <ol class="b-topics_second-level">
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=hello" class="b-topics_second-
level-link">Hello world!</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=vars"  class="b-topics_second-
level-link">Змінні</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=dataTypes"          class="b-
topics_second-level-link">Типи даних</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=typesTransform"     class="b-
topics_second-level-link">Перетворення типів</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=mathOperations"     class="b-
topics_second-level-link">Математичні оператори</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=conditions"         class="b-
topics_second-level-link">Умовні конструкції</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=loops" class="b-topics_second-
level-link">Цикли</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=functions"          class="b-
topics_second-level-link">Функції</a></li>
                </ol>
            </div>
```

```
            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Об'єкти
                </h2>
                <ol class="b-topics_second-level">
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=objects"            class="b-
topics_second-level-link">Основи об'єктів</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=objMethods"         class="b-
topics_second-level-link">Методи об'єктів</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=this"  class="b-topics_second-
level-link">This</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=objectCreation"     class="b-
topics_second-level-link">Створення об'єктів</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=objMath"            class="b-
topics_second-level-link">Математичні оператори</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=objCopy"            class="b-
topics_second-level-link">Копіювання об'єктів</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=descriptors"        class="b-
topics_second-level-link">Дескриптори властивостей об'єкта</a></li>
                </ol>
            </div>

            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
```

```
                Типи даних
            </h2>
            <ol class="b-topics_second-level">
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=numbers"            class="b-
topics_second-level-link">Числа</a></li>
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=strings"            class="b-
topics_second-level-link">Рядки</a></li>
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=arrays"             class="b-
topics_second-level-link">Масиви</a></li>
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=mapset"             class="b-
topics_second-level-link">Map i Set</a></li>
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=destruct"           class="b-
topics_second-level-link">Деструктуризація</a></li>
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=date"  class="b-topics_second-
level-link">Дата та час</a></li>
                            <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=json"  class="b-topics_second-
level-link">JSON</a></li>
            </ol>
        </div>

        <div class="b-topics_topic">
            <h2 class="b-topics_first-level">
                Функції, поглиблено
            </h2>
            <ol class="b-topics_second-level">
```

```
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=recursion"           class="b-
topics_second-level-link">Рекурсія та стек</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=restSpread"          class="b-
topics_second-level-link">Rest та spread оператори</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=closures"            class="b-
topics_second-level-link">Замикання</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=functionObject"      class="b-
topics_second-level-link">Об'єкт функції</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=timeout"             class="b-
topics_second-level-link">SetTimeout, SetInterval</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=attachContext"       class="b-
topics_second-level-link">Call, bind, apply</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=arrowFunc"           class="b-
topics_second-level-link">Стрілочні функції</a></li>
                </ol>
            </div>

            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Прототипи
                </h2>
                <ol class="b-topics_second-level">
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=prototypeNest"       class="b-
topics_second-level-link">Прототипне наслідування</a></li>
```

```
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=buitinProto"         class="b-
topics_second-level-link">Вбудовані прототипи</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=protoMethods"        class="b-
topics_second-level-link">Методи прототипів</a></li>
                </ol>
            </div>

            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Класи
                </h2>
                <ol class="b-topics_second-level">
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=classesSynt"         class="b-
topics_second-level-link">Синтаксис класів</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=inheritance"         class="b-
topics_second-level-link">Наслідування</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=classesMethods"      class="b-
topics_second-level-link">Методи класів</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=classesExpansion"    class="b-
topics_second-level-link">Розширення класів</a></li>
                </ol>
            </div>

            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Обробка помилок
```

```
                </h2>
                <ol class="b-topics_second-level">
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=tryCatch"           class="b-
topics_second-level-link">Try catch</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=userErrors"         class="b-
topics_second-level-link">Користувацькі помилки</a></li>
                </ol>
            </div>

            <div class="b-topics_topic">
                <h2 class="b-topics_first-level">
                    Асинхронний JavaScript
                </h2>
                <ol class="b-topics_second-level">
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=callback"           class="b-
topics_second-level-link">Колбеки</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=promises"           class="b-
topics_second-level-link">Проміси</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=promisesMethods"    class="b-
topics_second-level-link">Методи промісів</a></li>
                                <li  class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=asyncAwait"         class="b-
topics_second-level-link">Async/await</a></li>
                </ol>
            </div>

            <div class="b-topics_topic">
```

```
                <h2 class="b-topics_first-level">
                    Модуль
                </h2>
                <ol class="b-topics_second-level">
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=introduction"        class="b-
topics_second-level-link">Введення</a></li>
                                <li   class="b-topics_second-level-item"><a
href="http://localhost:3001/topic?topicName=exportImport"        class="b-
topics_second-level-link">Експорт та імпорт</a></li>
                </ol>
            </div>
        </div>
    </section>
    </main>
    <footer class="b-footer">
        <h1 class="b-header_logo">
            StudyJS
        </h1>
        <div>
        <h3 class="b-footer_author">Давиденко І.С.</h3>
            <ul class="b-footer_links">
                  <li class="b-footer_links-item"><a href="b-footer_links-
link"><img       src="images/facebook.png"       class="b-footer_links-img"
alt="Facebook icon"></a></li>
                  <li class="b-footer_links-item"><a href="b-footer_links-
link"><img      src="images/instagram.png"      class="b-footer_links-img"
alt="Instagram icon"></a></li>
                  <li class="b-footer_links-item"><a href="b-footer_links-
link"><img       src="images/linkedin.png"       class="b-footer_links-img"
alt="Linkedin icon"></a></li>
```

```
                <li class="b-footer_links-item"><a href="b-footer_links-
link"><img      src="images/telegram.png"      class="b-footer_links-img"
alt="Telegram icon"></a></li>
            </ul>
        </div>
    </footer>
    <script src="js/script.js"></script>
    </body>
    </html>

    Styles.scss
    @import "vars.scss";
    @import "mixins.scss";

    body {
        margin: 0;
    }
    .b-main {
        padding: 0 50px;
    }

    .b-header {
        display: flex;
        justify-content: space-between;
        height: 100px;
        background-color: $black;

        &_logo {
            margin: 25px;
            color: $white;
        }
```

```
    &_nav {
        list-style: none;
        padding: 0;
        margin: 30px 20px;

        &-item {
            display: inline;
            font-size: 20px;
            margin-left: 40px;
        }

        &-link {
            color: $white;
            text-decoration: none;
        }
    }
}

.b-start {
    display: grid;
    grid-template-columns: 40% 60%;
    margin: 50px 0;

    &_heading {
        margin-top: 180px;
        font-size: 30px;
    }

    &_text {
        font-size: 20px;
    }
```

```
    &_img {
        width: 100%;
    }
}

.b-topics {
    margin: 0 auto;
    box-shadow: 0px 0px 10px 0px rgba(0,0,0,0.75);
    width: 70%;

    &_search {
        margin: 30px 0 0 140px;
        width: 60%;
        height: 30px;

        &-btn {
            height: 34px;
            width: 80px;
            background-color: $black;
            color: $white;
            border-radius: 2px;
            font-weight: bold;
        }
    }

    &_first-level {
        text-align: center;
        margin-top: 35px;
    }

    &_second-level {
        display: flex;
```

```
        flex-direction: column;
        flex-wrap: wrap;
        max-height: 100px;

        &-item {
            margin-top: 10px;
            width: 30%;
        }

        &-link {
            color: $black;
            text-decoration: none;
            font-size: 18px;
            margin-left: 5px;
        }
    }
}

.b-topic {
    font-size: 18px;
    line-height: 24px;
    margin: 0 auto;
    width: 70%;
    .topic-heading {
        text-align: center;
    }

    &_code {
        background-color: $light_gray;
        padding: 15px;
        display: inline-block;
    }
```

```
}

.b-account {
    border: 1px solid $black;
    border-radius: 5px;
    width: 300px;
    height: 125px;
    margin: 100px auto;
    padding: 50px 100px 100px;
    background-color: $black;
    color: $white;

    &_label {
        font-size: 18px;
    }

    &_reg {
        width: 300px;
        padding: 50px 100px 100px;
        height: 250px;
        margin: 100px auto;
        border: 1px solid $black;
    }

    &_btn {
        &-auth {
            @include button;
            left: 200px;
            font-size: 18px;
        }

        &-reg {
```

```
            @include button;
            left: 400px;
            font-size: 18px;
        }
    }

    &_input {
        width: 250px;
        height: 20px;
        margin-bottom: 25px;
    }

    &_btn-submit {
        background-color: $white;
        color: $black;
        border-radius: 15px;
        font-weight: bold;
        height: 30px;
        width: 200px;
        position: relative;
        left: 30px;
        top: 20px;
    }
}

.b-task {
    &_heading {
        font-size: 32px;
        text-align: center;
    }

    &_desc {
```

```
        font-size: 24px;
        line-height: 30px;
    }

    &_submit {
        background-color: $black;
        color: $white;
        border-radius: 2px;
        font-weight: bold;
        height: 30px;
        width: 200px;
        position: relative;
    }
}

.b-footer {
    height: 90px;
    background-color: $black;
    padding: 30px 60px;
    color: $white;
    display: flex;
    justify-content: space-between;

    &_author {
        margin: 10px 0 0;
        font-size: 24px;
    }

    &_links {
        list-style: none;
        padding-left: 15px;
```

```
        &-item {
            display: inline-block;
        }

        &-img {
            width: 30px;
            height: 30px;
        }
    }
}

.b-profile {
    width: 70%;
    margin: 0 auto;

    &_heading {
        text-align: center;
    }

    &_info {
        display: grid;
        grid-template-columns: 60% 40%;
    }

    &_name {
        font-size: 26px;
    }

    &_surname {
        font-size: 26px;
    }
```

```
&_login {
    font-size: 26px;
    font-weight: bold;
}

&_table {
    border-collapse: collapse;

    th {
        border: 1px solid $black;
        font-size: 20px;
    }

    td {
        border: 1px solid $black;
        font-size: 20px;
    }

    &-number {
        width: 35px;
        padding: 10px;
    }

    &-name {
        padding: 10px;
        width: 150px;
        height: 25px;
    }

    &-completed {
        width: 15px;
        text-align: center;
```

```
            padding: 10px;
        }
    }
}
```

Додаток В – Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ВЕБ-СИСТЕМИ ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ JAVASCRIPT

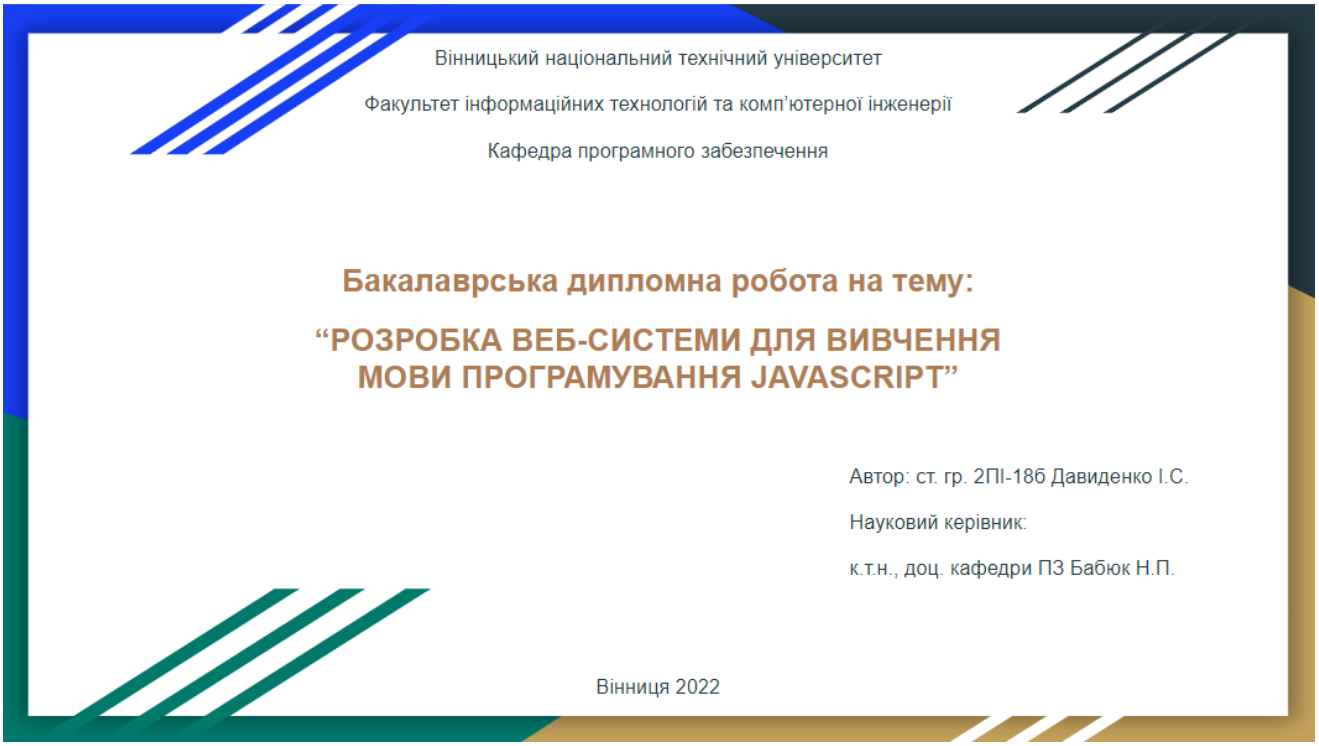


Рисунок В.1 – Титульний слайд

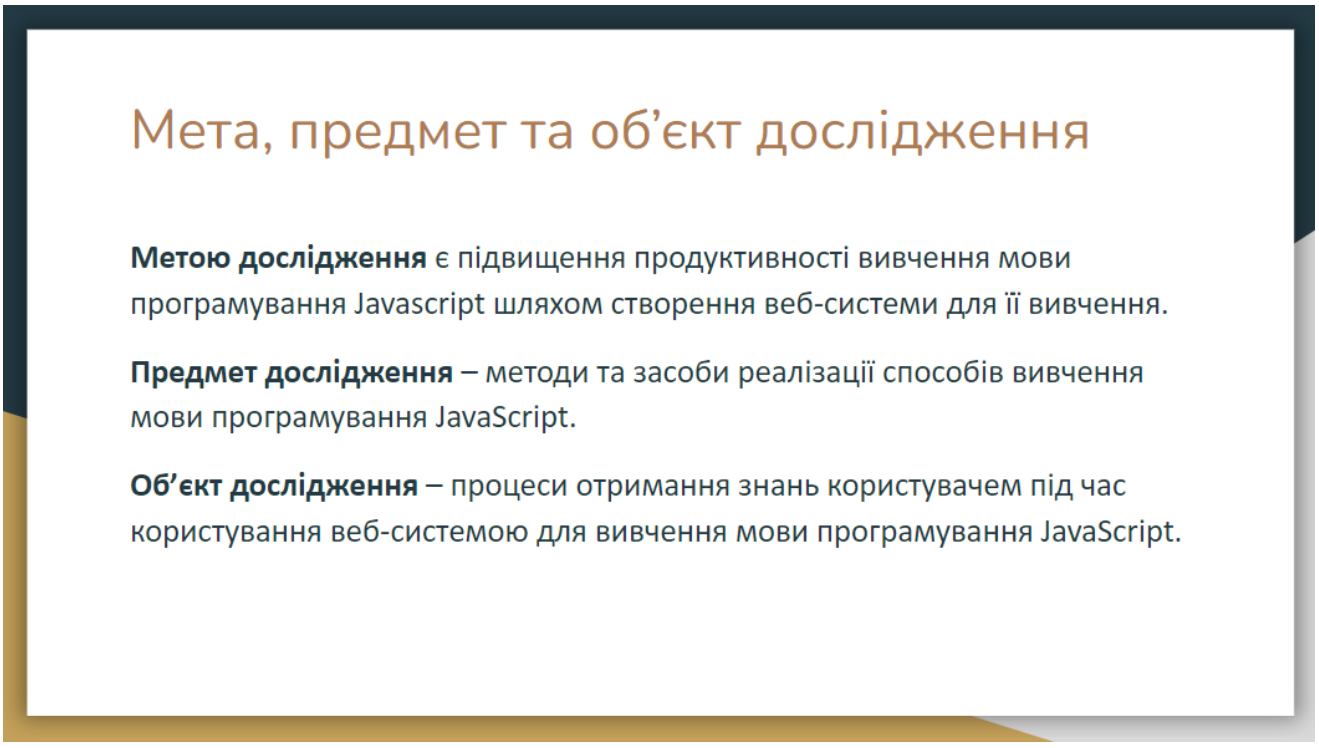


Рисунок В.2 – Мета, об'єкт та предмет дослідження

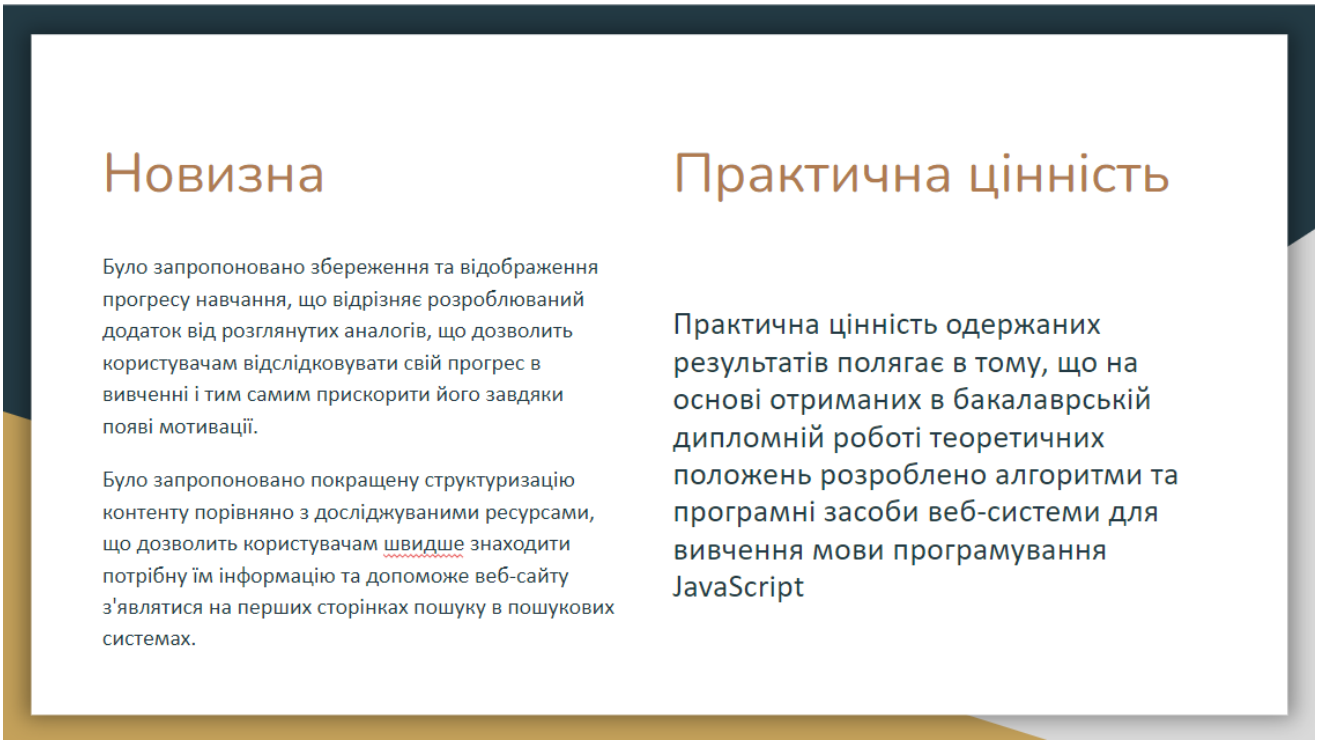


Рисунок В.3 – Новизна та практична цінність

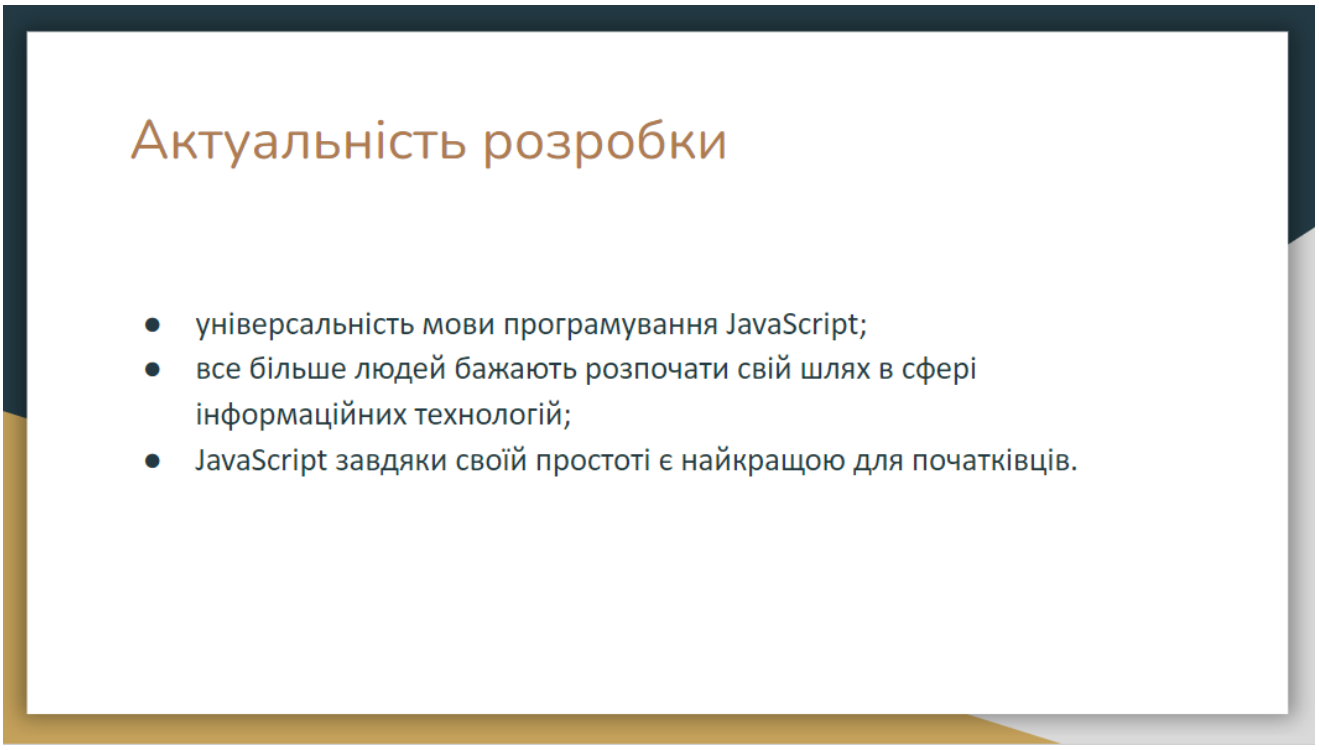


Рисунок В.4 – Актуальність розробки

Задачі дослідження

- аналіз стану галузі вивчення мови програмування JavaScript;
- порівняльний аналіз аналогів;
- аналіз методів розв’язання задачі;
- постановка задач;
- розробку структури і алгоритмів програмного продукту;
- варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу;
- розробка веб-системи для вивчення мови програмування JavaScript;
- тестування програми.

Рисунок В.5 – Задачі дослідження

Розробка структури інтерфейсу веб-системи

Через пункт «Навігація» користувач може потрапити до головних розділів сайту, тобто на головну сторінку, посібник чи на свій аккаунт. Привітальний блок виконує інформативну функцію і дає користувачу інформацію про те, де він знаходиться. Мапа сайту це всі навчальні розділі, тут користувач може перейти до будь-якої теми, яка його цікавить. На підвалі сайту користувач може побачити інформацію про розробника та те, як він може з ним зв’язатися в разі необхідності.

Рисунок В.6 – Розробка структури інтерфейсу веб-системи

Рисунок В.7 – Блок-схема алгоритму реєстрації та авторизації

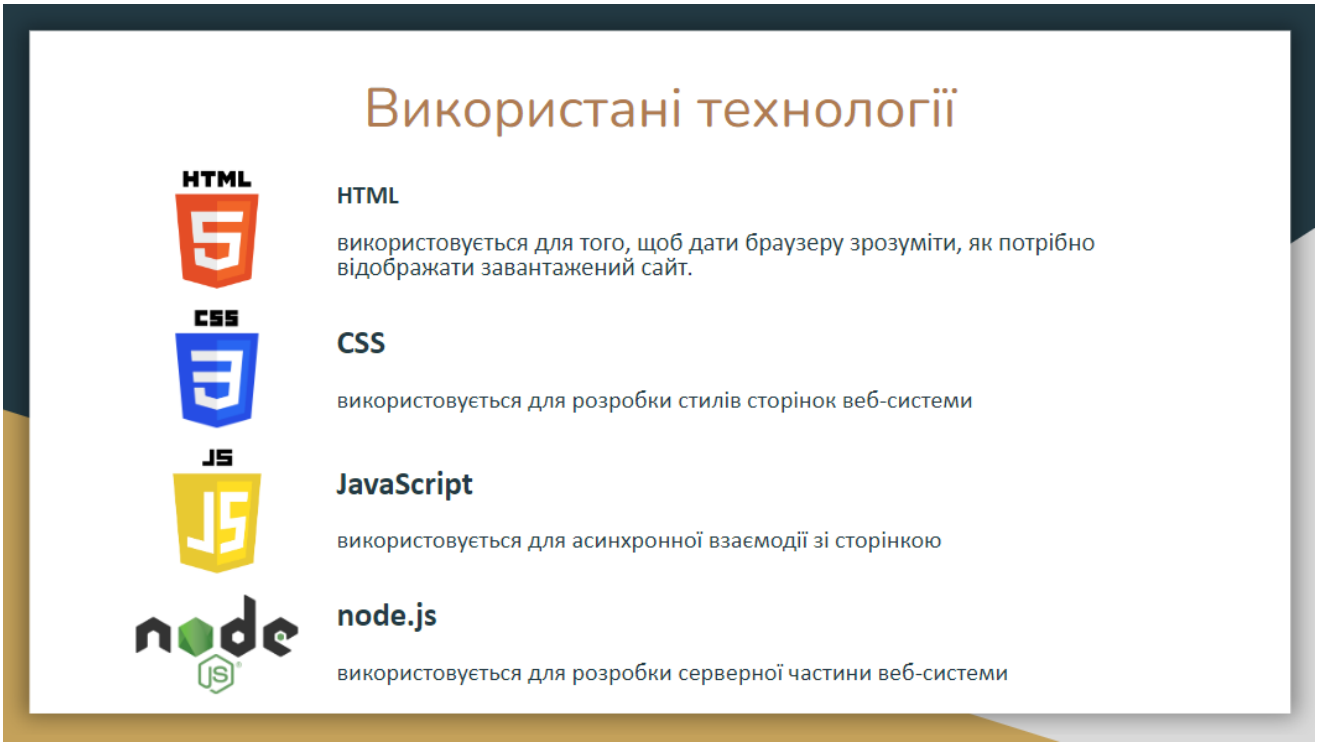


Рисунок В.8 – Використані технології

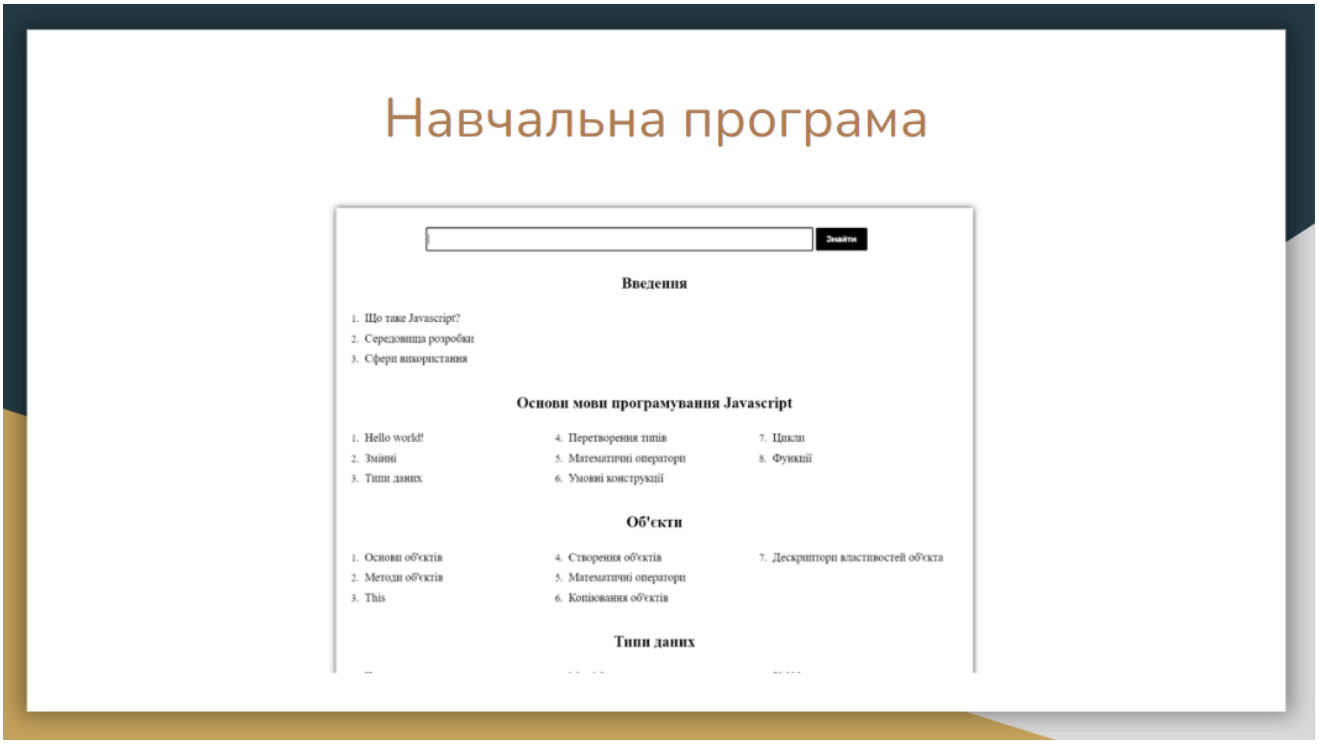


Рисунок В.9 – Навчальна програма

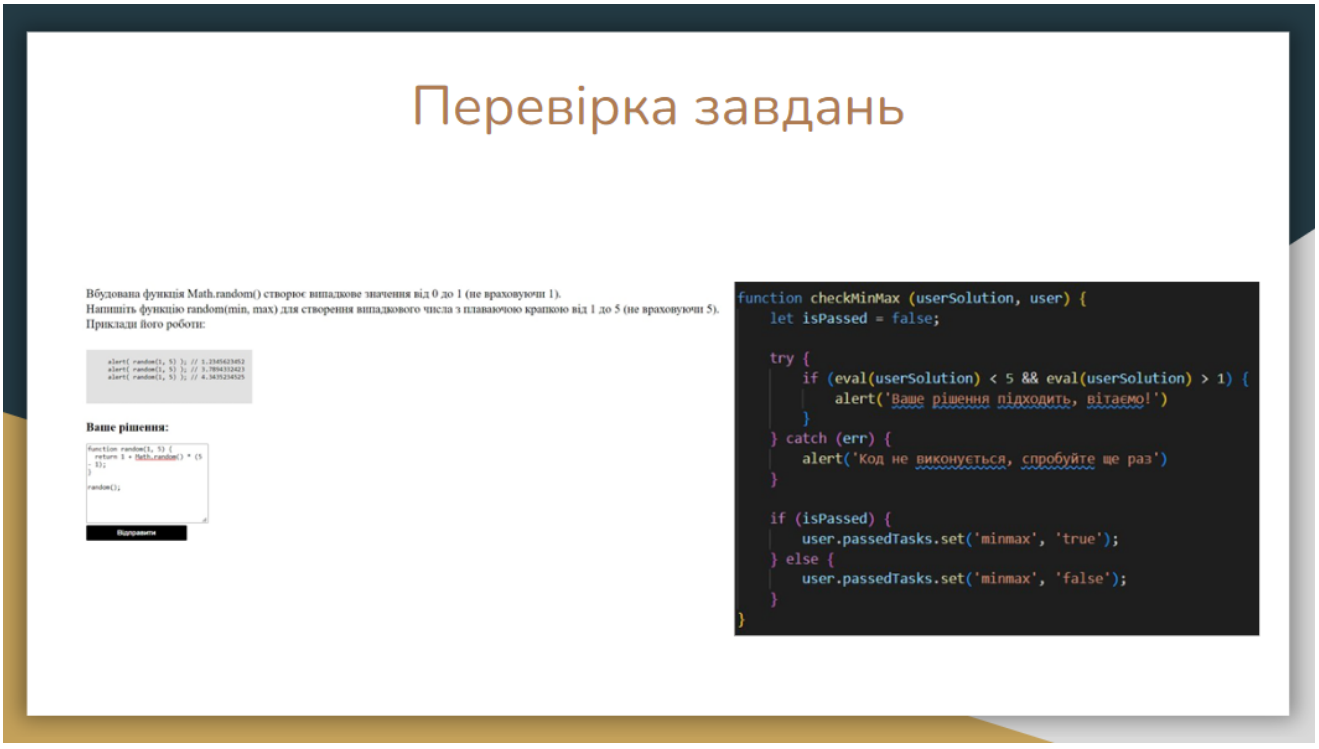


Рисунок В.10 – Перевірка завдань

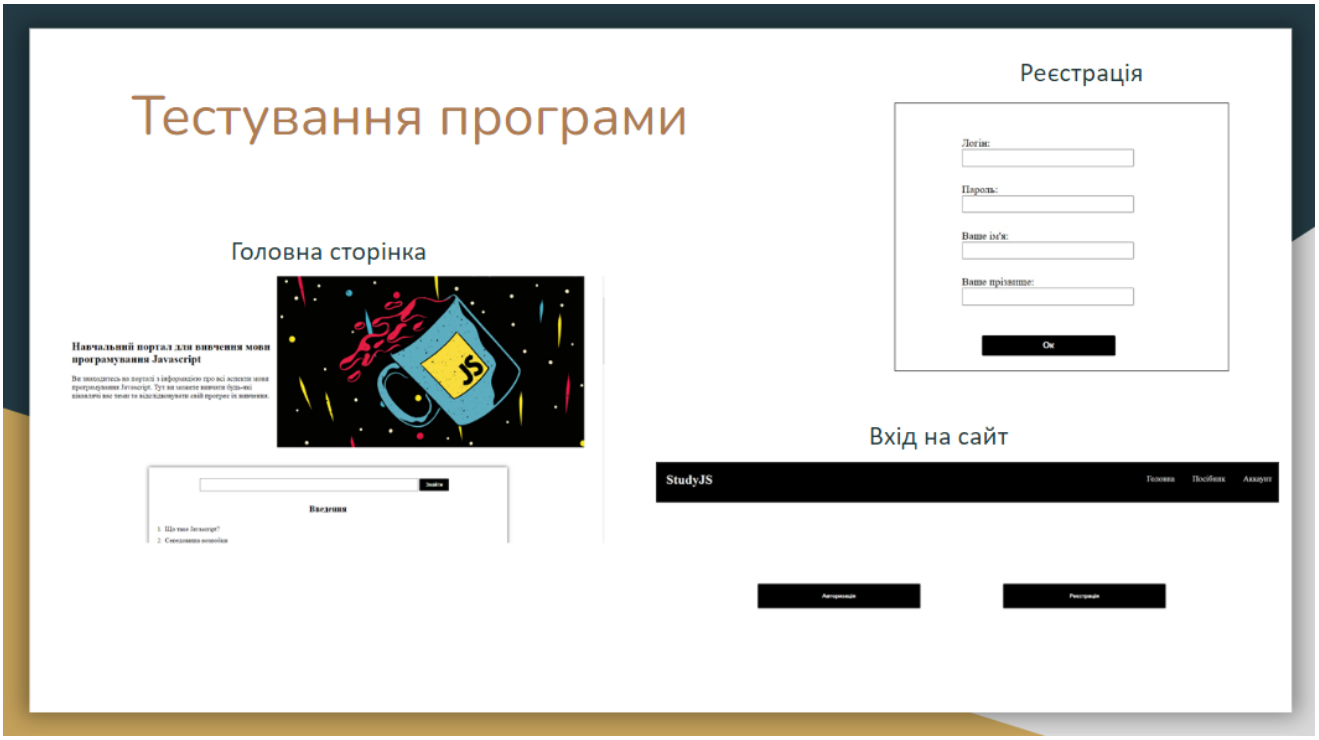


Рисунок В.11 – Тестування програми

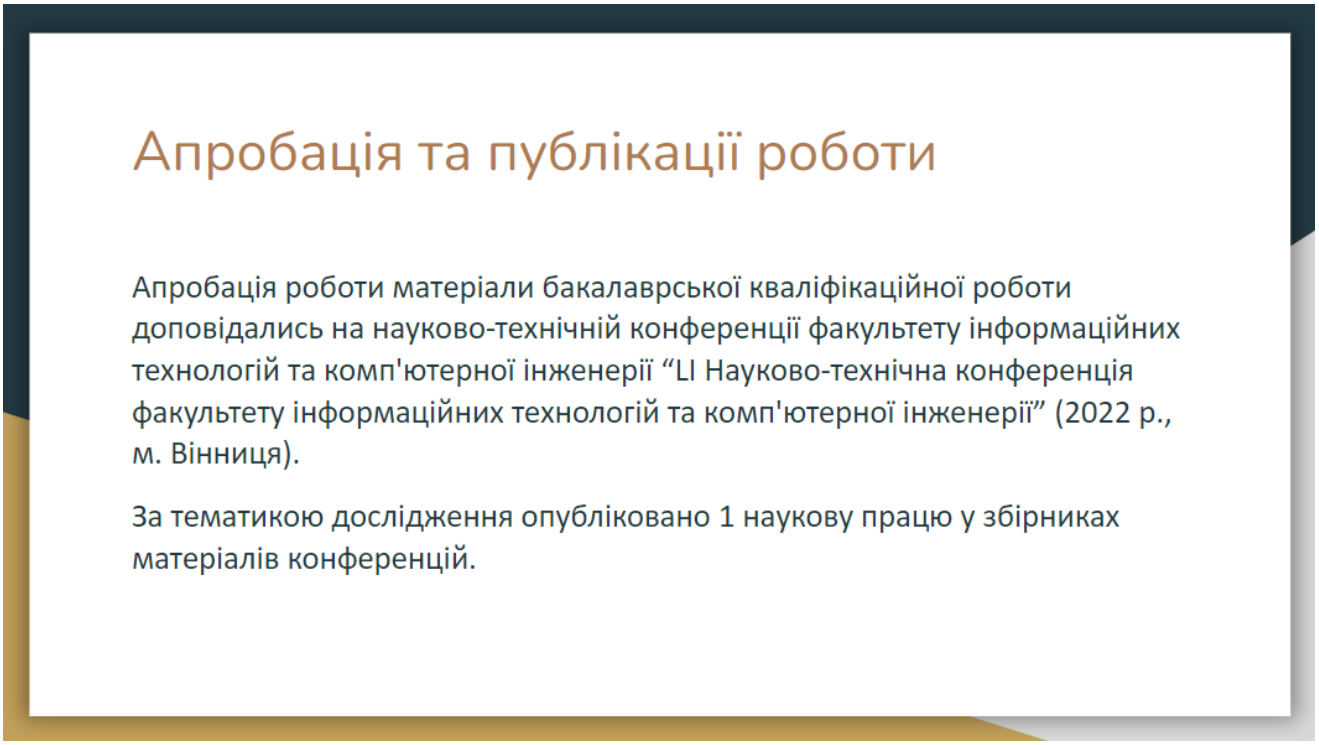


Рисунок В.12 – Апробація та публікації роботи

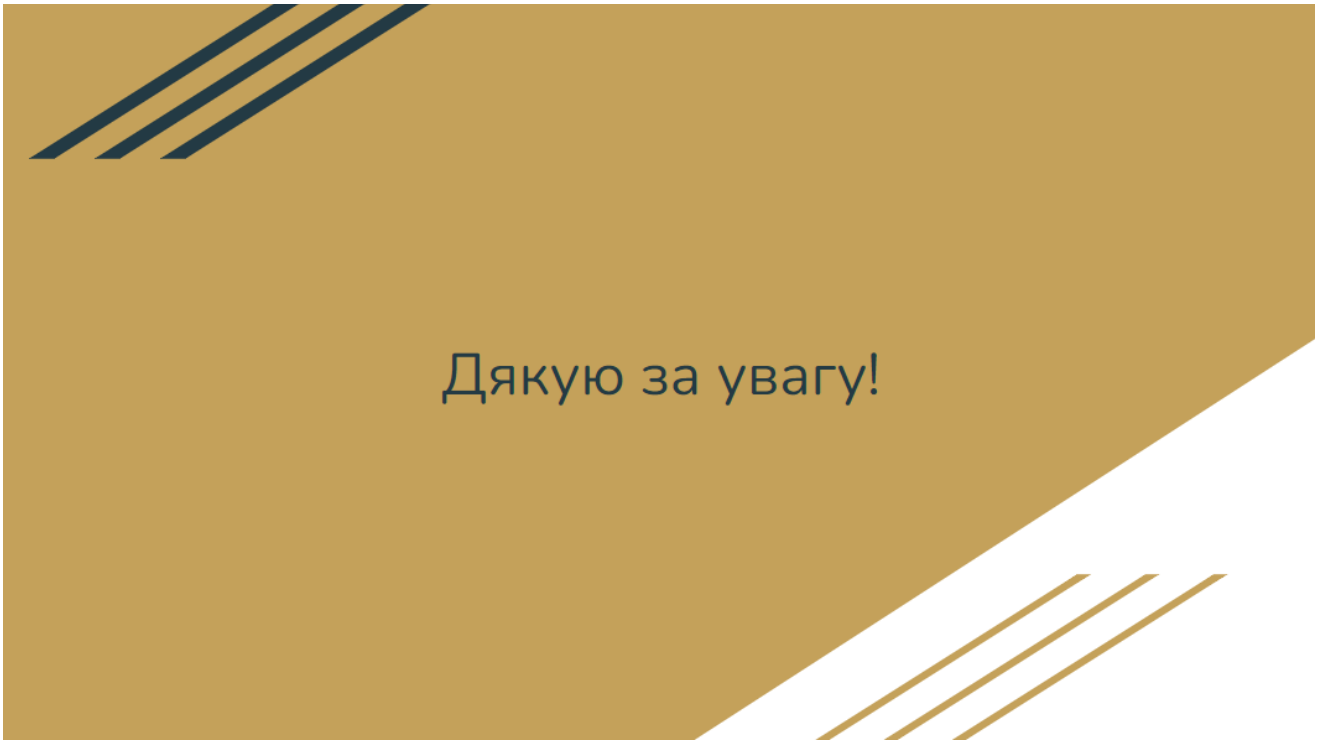


Рисунок В.13 – Фінальний слайд