Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра інфокомунікаційних систем та технологій

Бакалаврська кваліфікаційна робота на тему:

ВЕБ-ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ PYTHON НА ОСНОВІ ПРОТОКОЛУ HTTP

Виконав: студент 4-го курсу, групи ПЗТ-22мс
спеціальності 172 – Телекомунікації та радіотехніка
Базалицька М. Р.

Керівник: к.т.н., доцент каф. ІКСТ
Воловик А. Ю.
«11» 06 2024 р.

Рецензент: д.т.н., проф., професор каф. ІКСТ
Осадчук Я. О.
«11» 06 2024 р.

Допущено до захисту
Завідувач кафедри ІКСТ
д.т.н., проф. В. М. Кичак
«11» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет Факультет інформаційних електронних систем Кафедра інфокомунікаційних систем та технологійРівень вищої освіти перший (бакалаврський) Галузь знань 17 – Електроніка та телекомунікації
(шифр і назва)
Спеціальність 172 – Телекомунікації та радіотехніка
(шифр і назва)
Освітньо-професійна програма – Програмне забезпечення телекомунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКСТ
д.т.н., професор В.М. Кичак
«06» 05 2024 року

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

Базалицька Марина Романівна.
(прізвище, ім'я, по батькові)

1. Тема роботи «Веб-застосунок для вивчення мови програмування Python на основі протоколу http»
керівник роботи Воловик Андрій Юрійович, к.т.н, доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ВНТУ від « 11 » березня 2024р. № 80
2. Строк подання студентом роботи 0 7 ч е р в н я 2 0 2 4 р о к.
3. Вихідні дані до роботи: тип інфокомунікаційної мережі – програмно-керована мережа; режим роботи інфокомунікаційної мережі – багатоканальна; тип стандарту зв'язку – IEEE 802.11; сумарна швидкість інформаційних потоків - Ethernet 100 Мбіт/с згідно технічного завдання; імовірність помилки в радіотракті - 10^{-6}; кількість користувачів центру оброблення даних – до 100000; середовище розробки – Visual Studio Code; тип протоколу – http; мови розробки – HTHL, CSS, JavaScript; операційна система – Windows 10;
4. Зміст розрахунково-текстової частини роботи (перелік питань, які потрі- бно розробити): вступ; обґрунтування вибору методу розробки та постановка задач дослідження; аналіз існуючих протоколів та розроблення концепції веб-застосунку; розробка модулів програмного продукту; тестування програми, охорона праці; висновки; перелік посилань; додатки; графічна частина.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): таблиця порівняння аналогів, блок схема організації зв'язку, структура головної сторінки веб-системи, навчальна програма веб-системи, блок-схема алгоритму реєстрації/авторизації, ER-модель бази даних, головна сторінка веб-сайту, форма реєстрації та авторизації користувача.

2. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Спеціальна частина	Воловик А.Ю. доцент кафедри ІКТС	06.05.2024	11.06.2024
Охорона праці	Дембіцька С. В. професор кафедри БЖДПБ	06.05.2024	06.05.2024.

3. Дата видачі завдання « 06 » т р а в н я 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Формування та затвердження теми та індивідуального завдання бакалаврсь-кої кваліфікаційної роботи (БКР)	06.05.2024р.	
2.	Виконання спеціальної частини БКР. Перший рубіжний контроль виконання БКР	20.05.2024р.	
3.	Виконання спеціальної частини БКР. Другий рубіжний контроль виконання БКР	31.05.2024р.	
4.	Виконання розділу «Охорона праці»	04.06.2024р.	
5.	Нормоконтроль БКР	05.06.2024р.	
6.	Попередній захист БКР	06.06.2024р.	
7.	Рецензування БКР	07.06.2024р.	
8.	Захист БКР	12.06.2024р.	

Студент ____________ Базалицька М. Р.
(підпис)

Керівник роботи ____________ Воловик А. Ю.
(підпис)

АНОТАЦІЯ

УДК: 621.396.67

Базалицька М. Р. Веб-застосунок для вивчення мови програмування Python на основі протоколу http. Бакалаврська дипломна робота. – Вінниця: ВНТУ, 2024. 96 стор., 30 – рис., 7 – табл., 27 – бібл. – української мовою.

Метою даної бакалаврської дипломної роботи є розроблення програмного застосунок мовою JavaScript для підвищення продуктивності вивчення об’єктно-орієнтованої мови програмування Python на основі протоколу http. Проект включав аналіз галузі та вибір оптимального методу розробки, зокрема аналіз та обґрунтування вибору протоколу HTTP та відповідного обладнання (MySQL, phpMyAdmin, Open Server). Застосовані технології (HTML, CSS, JavaScript, Node.js). Реалізація веб-системи, що відповідає сучасним вимогам навчання користувачів з Python в телекомунікаційних системах.

.

ABSTRACT

UDC: 621.396.67

Bazalytska M.R. Web application for learning the Python programming language based on the http protocol. Bachelor thesis. – Vinnytsia: VNTU, 2024. 96 pages, 30 - fig., 7 - table, 27 - bibl. - in the Ukrainian language.

The purpose of this bachelor's thesis is to develop software applications in the JavaScript language to increase the productivity of learning the object-oriented programming language Python based on the http protocol. The project included the analysis of the industry and the selection of the optimal development method, in particular the analysis and justification of the choice of the HTTP protocol and the corresponding hardware (MySQL, phpMyAdmin, Open Server). Applied technologies (HTML, CSS, JavaScript, Node.js). Implementation of a web system that meets the modern requirements of training users with Python in telecommunication systems.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі інформаційних технологій програмування стало одним з важливих навичок, необхідних для професійного зростання і розвитку. Мова програмування Python займає особливе місце серед інших мов завдяки своїй простоті, універсальності та широкому спектру додатків. Python використовується в різних областях, від веб-розробки та аналізу даних до машинного навчання та наукових досліджень. Важливість вивчення цієї мови підтверджується високою популярністю серед розробників і зростаючим попитом на професіоналів, що говорять на Python.[3]

Інтернет та веб-технології відкривають нові можливості для навчання та саморозвитку. Веб-додатки, що дозволяють вивчати програмування в інтерактивному режимі, значно підвищать ефективність навчального процесу. Протокол HTTP, який є основою обміну даними через Інтернет, дозволяє створювати динамічну та інтерактивну навчальну платформу. Таким чином, розробка веб-додатків для вивчення мови програмування Python на основі протоколу HTTP відповідає сучасним вимогам і потребам освітнього процесу. [3]

Python надає студентам, програмістам та інженерам унікальну можливість розвивати навички програмування та застосовувати їх у реальних проектах. За допомогою Python можна створювати масштабовані та ефективні рішення для різноманітних завдань.

Мета роботи спрямована на розробку програмного засобу для підвищення продуктивності вивчення Python для студентів та початківців у програмуванні. Програмний засіб надасть зручний інтерфейс для доступу до навчальних матеріалів, а також інструменти для відстеження прогресу та підвищення продуктивності навчання. Використання JavaScript у розробці дозволить нам створити інтерактивний та зручний користувацький інтерфейс, що забезпечить ефективне сприйняття навчального матеріалу. JavaScript може бути використаний для взаємодії з сервером, зберігання даних та інших аспектів розробки.[8]

Об'єктом дослідження є процес розробки веб-застосунку для навчання програмуванню на мові Python, з використанням протоколу HTTP. Це включає мережеві протоколи, зокрема HTTP, механізми обміну даними між клієнтською та серверною частинами веб-застосунків. Аналіз існуючих протоколів.

Предмет дослідження є технології та методи розробки веб-застосунків, зокрема використання протоколу HTTP для забезпечення інтерактивного навчання мові програмування Python.

Наукова новизна одержаних результатів

Було запропоновано функціональні можливості які включають розгортання структурованого посібника з теоретичними матеріалами, прикладами коду та вправами, створення системи обліку користувачів з можливістю персоналізації та функції відстеження пройдених розділів.

Практичне значення одержаних результатів

Отримані результати дослідження мають практичне значення через розробку конкретних алгоритмів та програмних засобів для створення веб-системи, призначеної для навчання мови програмування Python.

Особистий внесок здобувача

Всі наукові висновки та результати були досягнуті здобувачем індивідуально.

Структура і обсяг роботи Бакалаврська дипломна робота складається зі вступу, 5 розділів, висновків, додатків та списку використаних джерел.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану галузі вивчення мови програмування Python

Популярність мови програмування Python стрімко зростає серед розробників усього світу. За останні роки Python став однією з найбільш використовуваних мов програмування завдяки своїй простоті вивчення, гнучкості та величезному екосистемі бібліотек і фреймворків.

Аналіз вакансій на ринку праці свідчить про постійне зростання попиту на фахівців, які володіють навичками Python. Великі технологічні компанії, такі як Google, Facebook, Microsoft, інвестують в розробку та підтримку проектів на Python. Крім того, індустрія штучного інтелекту та машинного навчання широко використовує Python як основну мову програмування для розробки та досліджень.

Статистика використання мови на GitHub також свідчить про популярність Python серед розробників. Python є однією з найпопулярніших мов серед проектів на GitHub, що демонструє широке використання мови у різних областях розробки програмного забезпечення.

Загальна тенденція свідчить про те, що Python продовжує займати провідні позиції у світі програмування, завдяки своїм зручним синтаксисом, багатофункціональністю та широким спектром застосувань у різних галузях, що робить його надзвичайно привабливим для розробників усього світу.

Веб-застосунки стали необхідною складовою сучасного інтернет-світу, використовуючись у різних сферах, від електронної комерції до соціальних мереж і корпоративних інструментів.

Ключові тенденції в цій галузі включають зростання популярності односторінкових застосунків (SPA), які забезпечують швидку та плавну взаємодію з користувачем без перезавантаження сторінки. Також, все більше уваги приділяється реактивному програмуванню, яке дозволяє створювати веб-застосунки, які реагують на зміни в реальному часі без необхідності оновлення сторінки.

У галузі розробки веб-застосунків також спостерігається широке використання відкритих інструментів та фреймворків, таких як React.js, Angular, Vue.js для фронтенду, і Django, Flask, Node.js для бекенду. Ці інструменти дозволяють розробникам швидко створювати ефективні та масштабовані веб-застосунки з використанням сучасних технологій.

Зростання мобільного трафіку також впливає на галузь розробки веб-застосунків, змушуючи розробників створювати адаптивні та мобільно-дружні додатки, які працюють на різних пристроях і розмірах екранів.

Напрямки, які наразі отримують особливу увагу в галузі, включають Progressive Web Apps (PWA), які поєднують в собі переваги веб-застосунків і мобільних додатків, та використання мікросервісної архітектури для побудови складних та масштабованих систем.

У цілому, галузь розробки веб-застосунків продовжує розвиватися швидкими темпами, пропонуючи нові можливості для розробників та відкриваючи нові шляхи для інновацій та створення корисних та цікавих продуктів для користувачів.

Розробка веб-застосунку для вивчення мови програмування Python за допомогою JavaScript має кілька важливих переваг:

Веб-застосунок може бути доступний з будь-якого пристрою з доступом до Інтернету, що робить його дуже зручним для користувачів. Вони можуть вивчати Python на своєму комп'ютері, планшеті чи навіть смартфоні.

Використання JavaScript дозволяє створити інтерактивні елементи на сторінках, такі як вправи, тести, ігри тощо. Це робить процес навчання більш захоплюючим і ефективним.

JavaScript може бути використаний для створення візуальних ефектів та діаграм, які полегшують розуміння складних концепцій. Це особливо корисно для відображення структури коду, алгоритмів чи даних. JavaScript може взаємодіяти з сервером, що дозволяє зберігати прогрес користувачів, відстежувати їхні досягнення та надавати персоналізовані рекомендації.

Веб-застосунок може легко розширюватися шляхом додавання нових функцій, уроків, вправ тощо без необхідності перевищення обсягу самого застосунку.

Загалом, розробка веб-застосунку для вивчення Python мовою JavaScript є логічним та ефективним вибором, який дозволяє створити потужний та інтерактивний інструмент для навчання, що може бути доступний для широкого кола користувачів.

1.2 Порівняльний аналіз аналогів розроблюваної системи

Одним зі способів визначення актуальності розробки веб-системи є аналіз та порівняння аналогів за ключовими критеріями оцінювання, особливо важливими для навчальних ресурсів. У сучасному світі інтернету існує безліч ресурсів, призначених для вивчення мови програмування Python. Більшість з них містять документацію, але не завжди це є найбільш ефективним методом навчання. Часто документація складається з тематичних статей та посібників, які можуть бути складними для розуміння для початківців. Деякі з цих матеріалів можуть вимагати від користувача певного рівня базових знань.

Крім того, деякі ресурси можуть бути застарілими або не містити актуальної інформації про сучасні практики програмування на Python. Тому важливо вибирати джерела, які надають оновлену та авторитетну інформацію, що відображає найновіші тенденції та рекомендації у світі програмування.

Однак проведення аналізу аналогів не обмежується лише пошуком навчальних ресурсів. Важливо також враховувати розвиток технологій та популярність певних методів навчання, таких як відеоуроки, інтерактивні онлайн-курси, практичні завдання тощо. Таким чином, проведення глибокого аналізу аналогів і порівняння їх за різними критеріями може допомогти визначити найбільш підходящі та актуальні ресурси для вивчення мови програмування Python.

Codecademy (див. рис. 1.1) – цей веб-сайт пропонує інтерактивні курси з Python, які розраховані на початківців із нульовим досвідом програмування. Кожен урок має чітке пояснення та практичні вправи, що допомагають закріпити знання.[12]

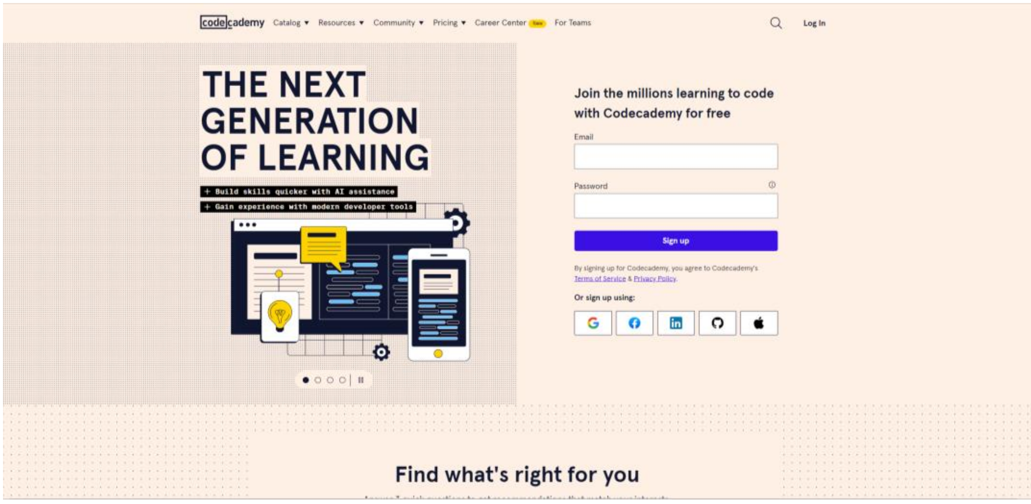

Рисунок 1.1 – Ресурс codecademy.com

Хоча Codecademy є дуже корисним ресурсом для початківців у програмуванні Python, він також має свої недоліки:

Багато змісту на Codecademy доступний тільки за плату. Хоча є безкоштовні курси, доступ до деяких покращених функцій та додаткового матеріалу може бути обмеженим.

Навіть у розділі для початківців, де кожен урок має чітке пояснення, може бути обмежена глибина пояснення. Деякі складні теми можуть потребувати додаткового самостійного дослідження.

Використання стандартного курсу може не враховувати індивідуальні потреби та темпи навчання кожного студента. Деяким користувачам може знадобитися додаткова підтримка або індивідуальне налаштування курсу.

SoloLearn (див. рис. 1.2) – це веб-платформа, де ви можете вивчати Python та інші мови програмування безкоштовно. Він пропонує короткі уроки та квізи, які допомагають засвоїти матеріал ефективним способом але так як платформа призначена для широкої аудиторії, вона може не надавати індивідуального підходу до кожного користувача. Деяким студентам може знадобитися додаткова підтримка або індивідуальне налаштування курсу.[14]

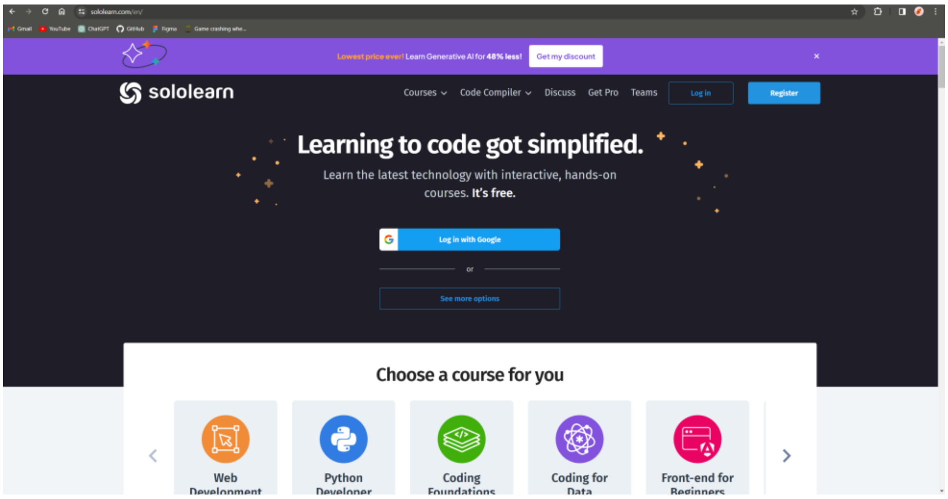

Рисунок 1.2 – Ресурс sololearn.com

Real Python (див. рис. 1.3) – це інтернет-журнал, що пропонує велику кількість безкоштовних статей, уроків та ресурсів для вивчення Python. Вони пропонують практичні поради, приклади коду та проекти для розвитку навичок у програмуванні Python.[13]

Рисунок 1.3 – Ресурс realpython.com

Real Python пропонує безліч інформації, але вона не завжди представлена в структурованому форматі. Це може ускладнити пошук необхідної інформації для початківців. Деякі статті та уроки на Real Python можуть бути занадто складними для початківців, що може призвести до втрати мотивації та зниження інтересу вивчення Python.

Результати порівняння аналогів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів

Критерії оцінювання	realpython	sololearn	codecademy	learnpython
Наявність тестувань	-	+	+	+
Структурова не навчання	-	-	-	+
Збереження прогресу навчання	-	+	+	+
Можливість створення аккаунта	+	+	+	+
Редактор коду	-	-	+	+

З розробленою веб-системою під назвою LearnPython було порівняно її аналоги за такими критеріями:

Наявність тестувань: Тестування дозволяє перевірити отримані знання і визначити слабкі місця. Це допомагає зорієнтуватися, які теми потребують додаткового вивчення.

Структуроване навчання: Структуроване навчання важливо для початківців, оскільки воно дозволяє систематизувати матеріал та легше орієнтуватися в його вивченні.

Збереження прогресу навчання: Можливість збереження прогресу дозволяє користувачам ефективно використовувати свій час, оскільки вони можуть продовжити навчання з того місця, де вони зупинилися.

Можливість створення аккаунта: Наявність особистого облікового запису дозволяє користувачам зберігати свій прогрес, підписуватися на оновлення темю

1.3 Аналіз методів розв'язання поставленої задачі

Після аналізу різних методів розв'язання поставленої задачі важливо ретельно зважити на переваги та недоліки кожного з них, оскільки правильний вибір методу може визначити подальший успіх проєкту.

Перший метод, який був розглянутий, полягає у використанні конструктора сайтів та готових бібліотек для створення зовнішнього вигляду та функціоналу веб-системи. Наприклад, ми можем скористатися такими конструкторами, як Wix, Weebly, або Squarespace, які надають широкий набір інструментів для швидкого створення зовнішнього вигляду та додавання функціоналу до нашого сайту.

Цей метод, безумовно, є простим у реалізації і може забезпечити швидкий початок роботи над проєктом. Однак, недоліком цього підходу є залежність від сторонніх ресурсів. Наприклад, конструктор сайтів може обмежувати нас у виборі дизайну або функціоналу, який ми хочем реалізувати. Також, можем зіткнутися з обмеженнями щодо налаштування або розширення функціоналу сайту у майбутньому. Наприклад, якщо додати складний інтерактивний елемент, може виявитися, що конструктор сайтів не надає необхідних можливостей для його реалізації.

Другий метод, який передбачає переробку вже існуючого схожого проєкту під власні потреби, може бути досить ефективним способом розробки, особливо якщо мати доступ до відповідного вихідного коду або ресурсів. Наприклад, якщо розглядати створення онлайн-магазину на основі вже існуючого рішення, можна скористатися відкритими або комерційними платформами, такими як WooCommerce або Shopify, і адаптувати їх під свої потреби.

Прикладом може бути переробка блогу на WordPress для створення веб-сайту з великою базою знань. Скористатися існуючою інфраструктурою WordPress, такою як система керування контентом (CMS) та відповідні теми та плагіни, для швидкої розробки веб-сайту. Після цього може знадобитися внести зміни у дизайні, структурі або функціоналі, щоб відповісти потребам проєкту.

Однак, при такому підході важливо пам'ятати про можливі складнощі у майбутньому. Залежність від вихідного коду, структури та архітектури вже існуючого проєкту, і це може ускладнити внесення змін або розширення проєкту у майбутньому. Обмеженнями у функціоналі, який може бути недоступним у вихідному коді або вимагатиме значних змін. Також, необхідно враховувати можливість виникнення конфліктів між змінами та оновленнями вихідного коду або плагінів.

Третій метод - розробка веб-системи з самого початку - вимагає більше часу та ресурсів, проте він надає повну свободу в розробці та максимальний контроль над усіма аспектами проєкту.

Розробка веб-системи з самого початку дійсно може бути більш часо- та ресурсомістким процесом порівняно з іншими методами, проте вона надає нам найбільшу свободу та контроль над усіма аспектами проєкту. Під час використання цього методу ми маємо можливість створити архітектуру, базу даних та функціонал з нуля, враховуючи конкретні потреби та вимоги нашого проєкту.

Наприклад, якщо ми розробляємо веб-систему для управління проектами з програмування ми можемо розпочати з проектування бази даних, визначивши структуру для зберігання інформації про користувачів, проекти, завдання, терміни та інші важливі дані. Після цього ми можемо розробити архітектуру програмного забезпечення, визначивши, як користувачі будуть взаємодіяти з системою, та розробивши відповідні модулі та компоненти.

Однією з переваг цього підходу є те, що ми можемо створити систему точно під наші потреби, уникнувши обмежень або недоліків, які можуть виникнути при використанні готових рішень або переробці існуючого коду. Ми також маємо повний контроль над якістю коду та можливістю розробляти систему, використовуючи сучасні технології та кращі практики програмування.

У відповідності до потреб нашого проєкту ми вирішили обрати:

Третій метод - розробка веб-системи з самого початку. Це дозволить нам забезпечити максимальну ефективність та контроль над усіма аспектами проєкту, що є критичним для досягнення успіху нашого проєкту у майбутньому.

1.4 Постановка задачі

Після аналізу сучасного стану галузі вивчення мови програмування Python та методів розробки продукту для розв'язання відповідної теми, були визначені наступні ключові задачі дослідження:

— Визначення найбільш ефективного методу навчання мови програмування Python.

— Розробка структури веб-сайту мовою HTML CSS.

— Створення клієнтської частини, що забезпечить зручне використання користувачами мовою JavaScript.

— Розробка алгоритму перевірки завдань користувачів після проходження тем мовою JavaScript..

— Реалізація бази даних для зберігання інформації про користувачів та їх прогрес навчання.

— Розробка модуля реєстрації та авторизації користувачів у веб-системі.

— Проведення тестування програмного модуля.

Отже проведено аналіз сучасного стану галузі вивчення мови програмування Python та методів розв'язання поставленої задачі. На основі порівняльного аналізу аналогів було вибрано метод розробки веб-системи з самого початку, який дозволяє забезпечити максимальну ефективність та контроль над усіма аспектами проєкту.

2 АНАЛІЗ ІСНУЮЧИХ ПРОТОКОЛІВ ТА РОЗРОБЛЕННЯ КОНЦЕПЦІЇ ВЕБ ЗАСТОСУНКУ

2.1 Аналіз та вибір протоколу для веб-застосунку

При створенні веб-застосунку важливо ретельно підібрати протоколи передачі даних, оскільки це визначає його ефективність, рівень безпеки та користувацьку зручність. Це рішення не лише впливає на швидкість обміну інформацією між клієнтом та сервером, але й забезпечує надійність та захист від зловмисних атак. Тому, аналіз та порівняння різних протоколів передачі даних стає ключовим етапом у процесі розробки, щоб зрозуміти, які саме з них найкраще відповідають потребам нашого проекту і сприятимуть його успішному функціонуванню. Найбільш поширені протоколи для реалізації веб-застосунку включають HTTP/HTTPS, WebSocket, REST, FTP. Отже, проведемо детальний аналіз основних протоколів даних.

2.1.1 Аналіз протоколу HTTP

HTTP (HyperText Transfer Protocol) – це основний протокол для передачі даних в Інтернеті, який використовується для завантаження веб-сторінок. HTTPS (HTTP Secure) – це розширення HTTP, яке забезпечує шифрування даних для підвищення безпеки. HTTP використовується для передачі гіпертекстових даних і взаємодії між клієнтськими та серверними пристроями. Цей протокол встановлює з'єднання між веб-клієнтом (наприклад, веб-браузером) і веб-сервером для обміну інформацією, такою як HTML-сторінки, зображення, аудіо- та відеофайли тощо. [6]

HTTPS, який використовує шифрування для захисту конфіденційності даних під час передачі через мережу. Він використовує SSL/TLS протоколи для шифрування даних, забезпечуючи захищену передачу інформації між клієнтом і сервером. HTTPS є особливо важливим для передачі конфіденційних даних, таких як особиста інформація, платежі тощо, оскільки він унеможливлює перехоплення та зловживання даними третіми особами під час їх передачі через мережу. HTTP/HTTPS використовується для обміну документами та іншими ресурсами між клієнтами та серверами. Це стандарт для веб-застосунків, де безпека та захист даних мають високе значення.

Основною перевагою даного протоколу є те що всі сучасні браузери (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera та інші) повністю підтримують протокол HTTP, включаючи його останні версії HTTP/2 та HTTP/3. Завдяки своїй популярності, HTTP має велику кількість документації, навчальних матеріалів, інструментів для розробки та налагодження (наприклад, Postman, cURL), що полегшує навчання та роботу з протоколом для розробників.

Також даний протокол має досить просту структуру яка складається з трьох основних компонентів:

Структура запиту:

— Лінія запиту (Request Line): метод (GET, POST, PUT, DELETE), URL, версія протоколу (наприклад, HTTP/1.1)

— Заголовки (Headers): ключ-значення (наприклад, Host, User-Agent, Accept)

— Тіло запиту (Body): використовується в методах POST, PUT для передачі даних

Структура відповіді:

— Лінія статусу (Status Line): версія протоколу, код статусу (наприклад, 200 OK)

— Заголовки (Headers): ключ-значення (наприклад, Content-Type, Content-Length)

— Тіло відповіді (Body): вміст відповіді (HTML, JSON, XML, тощо)

Структура HTTP дозволяє ефективно передавати дані між клієнтом і сервером, забезпечуючи при цьому простоту реалізації та використання. Це дозволяє початківцям швидко створювати прототипи веб-застосунків і легко розуміти, як працює цей протокол.

Однак HTTP може створювати високе навантаження на мережу. HTTP є безстановим протоколом, що означає, що кожен запит обробляється окремо від попередніх. Кожен запит і відповідь потребують встановлення нового з'єднання, що збільшує накладні витрати на з'єднання.

Кожен запит вимагає встановлення нового TCP (Transmission Control Protocol)-з'єднання, що включає тристоронній обмін (TCP handshake). Це збільшує час очікування і мережеві накладні витрати.

HTTPS додає додаткові витрати на встановлення захищеного з'єднання через SSL/TLS handshake. Кожен HTTP-запит і відповідь містять заголовки, які можуть бути значними за розміром. Це особливо актуально для часто повторюваних запитів, де заголовки можуть складати значну частину від загального обсягу переданих даних.

Для статичних ресурсів, таких як зображення, CSS, JavaScript, що рідко змінюються, HTTP/1.1 підтримує кешування, але без оптимального використання нові з'єднання можуть бути неефективними.

2.1.2 Аналіз протоколу WebSocket

WebSocket – це протокол, що забезпечує двонаправлений зв'язок між клієнтом і сервером. Він дозволяє відкривати постійне з'єднання для обміну даними в реальному часі. WebSocket використовується для застосунків, що вимагають швидкого обміну даними в реальному часі, таких як чати, онлайн-ігри та фінансові біржі.[10]

WebSocket має менший вплив на навантаження на мережу. Оскільки після встановлення WebSocket-з'єднання немає потреби у повторному встановленні з'єднання для кожного повідомлення. Це знижує накладні витрати, пов'язані з відкриттям і закриттям з'єднань, як у випадку HTTP. Завдяки постійному з'єднанню WebSocket забезпечує швидкий і безперервний обмін даними, що робить його ідеальним для застосунків, які потребують мінімальних затримок і високої частоти оновлень.

WebSocket використовує структуру фреймів для передачі даних після встановлення з'єднання. Фрейм у контексті мережевих протоколів передачі даних - це структурована одиниця даних, яка використовується для організації та передачі інформації між комп'ютерами або іншими пристроями в мережі. Фрейми розбивають потік даних на менші сегменти, які можуть бути ефективно передані, зібрані та оброблені.

Структура з'єднання:

— Handshake: відбувається через HTTP, після чого з'єднання переходить у WebSocket

— Дані: передаються як двійкові або текстові фрейми

Фрейм:

— Заголовок фрейму (Frame Header): містить інформацію про тип даних, довжину, флаги

— Тіло фрейму (Frame Payload): безпосередні дані повідомлення

Основні типи фреймів:

— TEXT: текстові дані.

— BINARY: двійкові дані.

— CLOSE: закриття з'єднання.

— PING: перевірка з'єднання.

— PONG: відповідь на PING.

Структура фрейму:

— Байт 0: FIN, RSV1-3, Opcode.

— Байт 1: Маска (Mask) і довжина даних.

— Байти 2-6: Розширена довжина (якщо необхідно).

— Байти 6-10: Маска (якщо є).

— Наступні байти: Дані (Payload).

Однак даний протокол має певні недоліки а саме впровадження WebSocket вимагає додаткових зусиль і знань, оскільки необхідно забезпечити підтримку постійного з'єднання, обробку подій, управління з'єднаннями, а також врахувати різні сценарії відключення і повторного підключення. Замість простої моделі запит-відповідь, як у HTTP, WebSocket використовує подієво-орієнтовану модель, де обидві сторони можуть ініціювати обмін повідомленнями. Це вимагає реалізації логіки обробки подій і повідомлень.

2.1.3 Аналіз протоколу REST

Протокол REST (Representational State Transfer) — це архітектурний стиль для створення веб-сервісів. REST базується на наборі принципів та обмежень, які визначають, як веб-ресурси мають бути адресовані та маніпульовані за допомогою уніфікованого інтерфейсу, що використовує стандартизовані HTTP методи. Клієнти та сервери є окремими компонентами, що дозволяє незалежно розвивати та масштабувати кожну з частин. Кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для розуміння та обробки запиту. Сервер не зберігає жодної інформації про стан клієнта між запитами. Відповіді повинні явно позначатися як кешовані або некешовані. REST використовується для створення веб-сервісів, що легко інтегруються та масштабуються.[11]

Структура базується на HTTP-запитах і відповідях:

— URL: вказує на ресурс (наприклад, /users/1 для отримання інформації про користувача з ID 1)

— Методи HTTP: визначають операції над ресурсами (GET, POST, PUT, DELETE)

— Заголовки: передають метаінформацію (наприклад, Content-Type, Authorization)

— Тіло: використовується для передачі ресурсів у форматах JSON, XML

Основні переваги даного протоколу являють собою те що REST використовує стандартні HTTP методи (GET, POST, PUT, DELETE, PATCH), що є широко відомими і зрозумілими. Використання стандартних форматів даних, таких як JSON та XML, робить обмін даними простим і уніфікованим. JSON, зокрема, є легким для читання та написання, що спрощує процес розробки і зменшує кількість помилок. Клієнт та сервер можуть розвиватися незалежно один від одного, оскільки REST визначає чіткий контракт між ними у вигляді HTTP методів і форматів даних. Це означає, що зміни на боці сервера не вимагають обов'язкових змін на боці клієнта, і навпаки.

Але даний протокол має свої недоліки першим з них є тещо REST не має офіційного стандарту, що призводить до різноманітних підходів у реалізації. REST може мати різні інтерпретації, що створює проблеми з сумісністю та інтероперабельністю між різними сервісами. Відсутність жорстких правил призводить до того, що кожен API може мати свої особливості в структурі URL, форматі відповіді, обробці помилок тощо.

Також його безстанова природа означає, що кожен запит повинен містити всю інформацію для його обробки, включаючи аутентифікацію, ідентифікацію клієнта та будь-які інші дані, необхідні для виконання запиту. Це призводить до збільшення навантаження на мережу адже кожен запит містить всі дані, необхідні для його обробки, що може призводити до більшого об'єму переданих даних порівняно зі станом-серверними протоколами. Повторювані дані в кожному запиті можуть збільшувати об'єм трафіку, особливо в ситуаціях, коли клієнт здійснює багато запитів до одного і того ж ресурсу.

2.1.4 Аналіз протоколу FTP.

Протокол FTP (File Transfer Protocol) - це стандартний протокол, що використовується для передачі файлів між комп'ютерами через мережу Інтернет або локальну мережу. FTP дозволяє користувачам завантажувати та вивантажувати файли з серверів, що підтримують цей протокол. Він є одним з найпоширеніших способів передачі файлів у мережі.[9]

Протокол FTP використовує клієнт-серверну архітектуру, де клієнти встановлюють зв'язок з FTP-серверами для передачі файлів. FTP може бути використаний для завантаження веб-сторінок на веб-сервер, резервного копіювання файлів, віддаленого доступу до файлів та багатьох інших сценаріїв.

Протокол FTP включає в себе ряд команд, які дозволяють клієнту взаємодіяти з сервером, таких як команди для перегляду вмісту каталогу, завантаження файлів, видалення файлів тощо. Крім того, FTP підтримує різні режими передачі, включаючи ASCII і бінарний режими, що дозволяють передавати різні типи даних з використанням оптимального формату.

Ключовими перевагами даного протоколу є тещо він простий у використання FTP робить його доступним. Популярність FTP означає, що він широко підтримується різними програмами і системами, що робить його зручним для використання в різних сценаріях, від особистого використання до великих корпоративних систем. FTP надає різноманітні функції, які роблять його більш гнучким і корисним. Наприклад, можливість перегляду каталогів, створення папок, перейменування файлів тощо дозволяє ефективно керувати файлами на сервері.

Однак має свої недоліки перший з них це використання стандартного порту 21 для з'єднання, що може викликати проблеми з відкриттям з'єднання через файерволи або мережеві обмеження. Це може вимагати налаштування файерволів або використання альтернативних портів, що може бути не зручним. У порівнянні з сучасними протоколами передачі файлів, FTP може мати обмежену швидкість передачі, особливо при великих обсягах даних. Це через відсутність підтримки стиснення даних або оптимізації передачі, яку можна знайти у більш сучасних протоколах, таких як SFTP або FTPS.

2.1.5 Обґрунтування вибору протоколу HTTP

Проаналізувавши найбільш поширені протоколи для реалізації веб-застосунку включають HTTP/HTTPS, WebSocket, REST, FTP. Можна зробити висновок що найкраще обрати саме HTTP , адже він є найпростішим та найлегшу структуру, що дозволяє легко реалізувати його як на стороні клієнта, так і на стороні сервера. Це зробить веб-застосунок більш доступним і сприятиме швидкому розвитку прототипів.

HTTP має вбудовану підтримку в браузерах, що робить його ідеальним вибором для веб-застосунків. У порівнянні з WebSocket, який може вимагати налаштування або підтримки спеціалізованих бібліотек, HTTP є більш доступним і легким у використанні.

HTTP може бути забезпечений безпекою, що забезпечує шифрування даних та безпечну передачу через мережу. У порівнянні з FTP, який передає дані у відкритому вигляді, HTTP є більш безпечним варіантом для обміну конфіденційною інформацією.

2.2 Вибір та обґрунтування серверного обладнання

Для успішного розгортання веб-застосунку необхідно ретельно обрати відповідне серверне обладнання, враховуючи всі технічні та операційні вимоги, які включають високу продуктивність серверів, надійність мережевого обладнання та здатність системи до масштабування при зростанні навантаження. Для забезпечення надійного та ефективного мережевого зв'язку між серверами та клієнтами використовуються ключові компоненти інфраструктури, такі як маршрутизатори та модеми. Ці пристрої відповідають за маршрутизацію даних через Інтернет та забезпечують постійну доступність нашого додатку для користувачів. Використання високоякісного мережевого обладнання дозволяє гарантувати стійкість роботи системи навіть при великому обсязі трафіку, забезпечуючи швидкий та безперебійний обмін даними. Основним пристроєм у розробці є сервер, який відіграє критичну роль у забезпеченні надійності, швидкості та ефективності роботи системи.

Для ефективного обміну даними через HTTP необхідно мати належно налаштовану мережеву інфраструктуру, що включає у себе маршрутизатори, комутатори, а також веб-сервери та клієнтські пристрої. Важливим є також правильне розуміння структури та особливостей протоколу, а також використання відповідних програмних засобів для обробки HTTP запитів і відповідей. Схема зв'язку представлена у вигляді схеми (див. рис. 2.2).

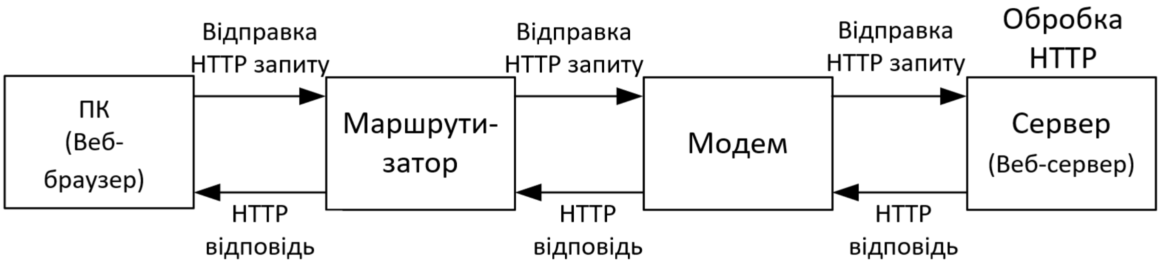

Рисунок 2.2 – Блок схема організації зв’язку

На даній схемі представлено:

ПК, браузер – це клієнт пристрій або програма, яка ініціює HTTP запит до сервера для отримання певної інформації. Зазвичай в клієнтській ролі виступає веб-браузер, який працює на персональному комп'ютері, смартфоні, планшеті або іншому пристрої.

При цьому браузер відправляє запит на сервер, вказуючи URI (Uniform Resource Identifier) ресурсу, який потрібно отримати, та інші необхідні параметри запиту, такі як метод запиту (GET, POST тощо), заголовки, куки та інші. Коли сервер отримує запит, він обробляє його та формує відповідь, яка містить необхідну інформацію або ресурси, які запитує клієнт.

Маршрутизатор – це проміжний пристрій, який дозволяє маршрутизувати пакети даних між локальною мережею та Інтернетом. Він забезпечує з'єднання між ПК і сервером через Інтернет. Маршрутизатор визначає оптимальний шлях для пересилання пакетів даних між локальною мережею і Інтернетом. Він враховує різні фактори, такі як найменший час затримки та доступність шляху, щоб забезпечити ефективну передачу даних. Маршрутизатор може виконувати функцію перетворення адресів мережі (NAT), що дозволяє приховати внутрішні адреси пристроїв у локальній мережі від зовнішнього Інтернету. Багато сучасних маршрутизаторів мають вбудовані брандмауери, які фільтрують трафік, що проходить через них, забезпечуючи додатковий рівень безпеки.

Модем дозволяє ПК підключатися до Інтернету через провідну або бездротову мережу. Його основна функція - перетворення цифрових сигналів, які розуміє комп'ютер, на аналогові сигнали, придатні для передачі по фізичному середовищу, такому як телефонні лінії або кабельні мережі, і навпаки - перетворення аналогових сигналів з лінії в цифрові дані, які може розуміти комп'ютер.

Сервер являє собою комп'ютер або комп'ютерний пристрій, що використовується для зберігання даних та надання їх для доступу іншим пристроям у мережі. У контексті веб-технологій, сервер часто використовується для надання доступу до веб-сторінок та інших веб-ресурсів через Інтернет.

Основна функція сервера полягає в обробці запитів від клієнтів і наданні їм необхідних ресурсів. Для цього на сервері може бути встановлене веб-серверне програмне забезпечення, яке служить для обробки HTTP запитів від клієнтів та надання їм веб-сторінок, зображень, відео, даних з баз даних та інших ресурсів.

Сервер поєднує в собі апаратне забезпечення та програмне забезпечення. Апаратна частина складається з фізичних носіїв даних, які можуть мати різний зовнішній вигляд та технічні характеристики. Ці носії встановлюються в дата центрах або в окремих приміщеннях компаній. Програмна частина включає операційну систему та спеціальні програми, які забезпечують обробку даних та виконання різних завдань.

Виділений сервер – це фізичний сервер, який надається в повне розпорядження одному користувачу або організації. Він надає високий рівень продуктивності, безпеки та контролю, оскільки ресурси сервера не діляться з іншими користувачами. Виділені сервери зазвичай використовуються для важливих додатків, великих веб-сайтів, баз даних та інших критично важливих завдань, що вимагають максимальної надійності та продуктивності.

Виділений сервер ідеально підходить для великих компаній, інтернет-магазинів, популярних веб-додатків та інших проєктів, де потрібна висока продуктивність та надійність.

Віртуальний сервер – це логічний розділ фізичного сервера, який функціонує як окремий сервер завдяки технології віртуалізації. Кожен віртуальний сервер має власну операційну систему, ресурси (процесор, оперативну пам’ять, дисковий простір) та може працювати незалежно від інших віртуальних серверів, розташованих на тому ж фізичному обладнанні.

Віртуальні сервери ідеально підходять для малих та середніх компаній, розробників та тих, хто потребує гнучких та масштабованих рішень для хостингу веб-додатків, тестування програмного забезпечення, або інших IT-завдань.

Віртуальний хостинг, або хмарний сервер, – це тип хостингу, де ресурси фізичного сервера розподіляються між багатьма користувачами. Це означає, що кілька веб-сайтів можуть розміщуватися на одному фізичному сервері, але кожен сайт ізольований від інших. Кожен користувач отримує певну кількість ресурсів (дисковий простір, оперативну пам'ять, процесорну потужність) і обмеження, але сервер обслуговується спільно.

Віртуальний хостинг підходить для малих і середніх веб-сайтів, блогів, особистих сайтів та невеликих інтернет-магазинів, де важливі економічність і простота управління.

Отже, для нашої розробки ми обрали хмарний сервер, оскільки він надає гнучкість, масштабованість та надійність, що необхідні для успішної інтеграції та розвитку наших продуктів. Завдяки хмарним технологіям ми можемо швидко масштабувати наші ресурси у відповідності зі зростанням обсягу роботи, забезпечуючи ефективне використання ресурсів та уникнення перебоїв у роботі. Також хмарні сервери забезпечують нам високий рівень доступності і безпеки, що важливо для забезпечення безперебійної роботи наших сервісів та захисту конфіденційності даних наших користувачів.

2.3 Вибір та обґрунтування програмного забезпечення

Щоб наш додаток міг ефективно працювати, нам потрібно ретельно підібрати програмне забезпечення, яке буде забезпечувати надійну та безперебійну роботу веб-системи. Враховуючи, що важливість збереження та обробки даних, особливо на сервері, є однією з ключових складових нашого додатку, обране програмне забезпечення має забезпечити продуктивність. Також важливо, щоб воно було сумісним із вимогами нашого проекту та забезпечувало необхідний функціонал для роботи з даними, що зберігаються на сервері.

Для ефективного зберігання та організації даних нашого веб-застосунку потрібно обрати відповідну базу даних. База даних відіграє ключову роль у збереженні, оновленні, видаленні та витягуванні інформації, що використовується нашим додатком.

MySQL - це система управління базами даних (СУБД), яка використовується для зберігання та управління структурованою інформацією. Вона є однією з найпопулярніших та найвикористовуваніших СУБД у світі, особливо серед веб-розробників та розробників програмного забезпечення.[]

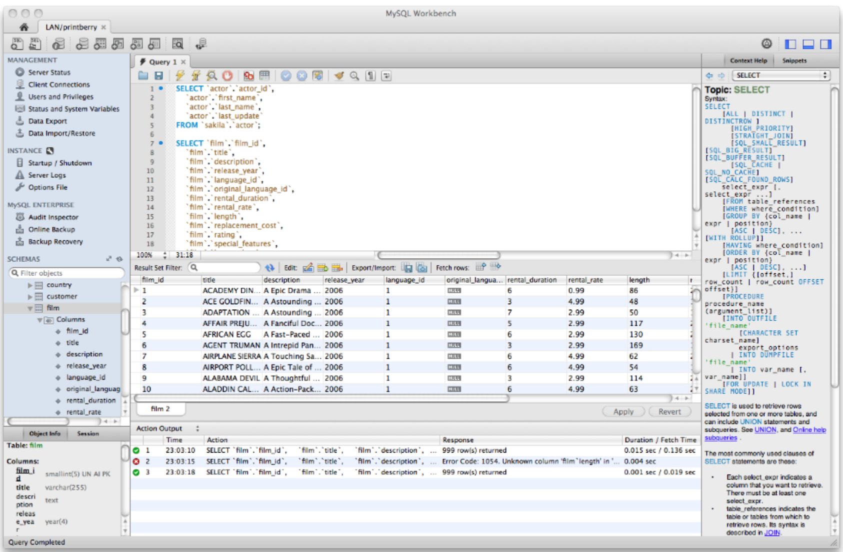

Рисунок 2.3 – MySQL Workbench

MySQL відома своєю швидкодією, надійністю, масштабованістю та простотою використання. Вона підтримує різні операційні системи і мови програмування, що робить її відмінним вибором для різних типів проєктів, включаючи веб-додатки, мобільні додатки, корпоративні системи та інше (див. рис. 2.3).

MySQL має відкритий вихідний код, що означає, що ви можете вільно використовувати, змінювати та розповсюджувати її. Крім того, велика спільнота користувачів та розробників активно підтримує розвиток і підтримку MySQL, що забезпечує швидке виправлення помилок та впровадження нових функцій.

Також для кращої роботи з MySQL слід обрати веб-інтерфейс який забезпечить зручний спосіб керування базою даних через веб-браузер. Програма phpMyAdmin дозволяє легко виконувати запити SQL, керувати таблицями, створювати резервні копії та багато іншого, що робить його ідеальним інструментом для адміністрування бази даних MySQL.[16]

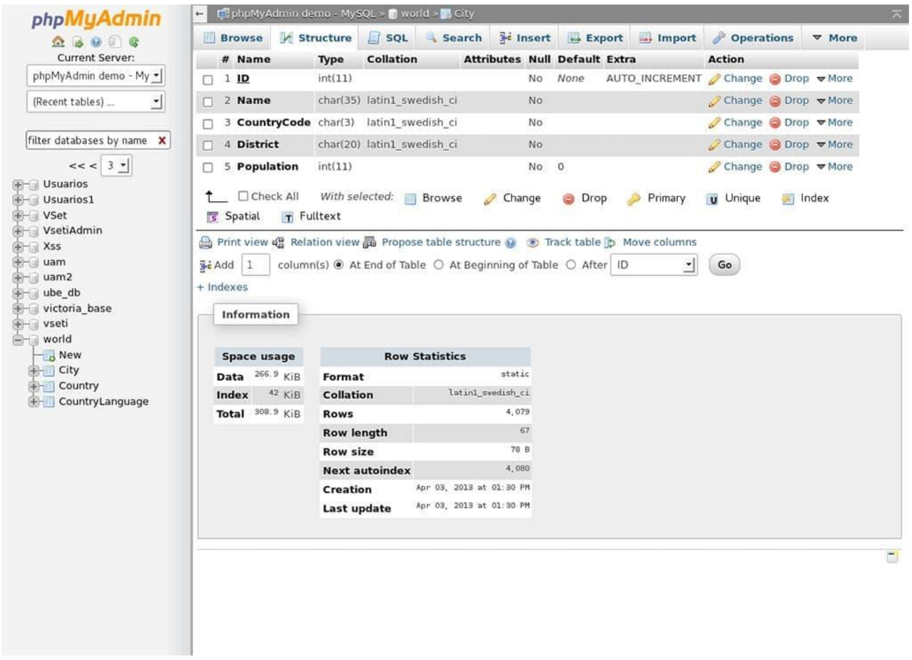

Рисунок 2.4 – phpMyAdmin веб-інтерфейс для роботи з MySQL

phpMyAdmin — це безкоштовний веб-інструмент із відкритим кодом, написаний на PHP, призначений для керування MySQL або MariaDB за допомогою веб-браузера. Інтерфейс зображено на рисунку 2.4.

Він забезпечує інтуїтивно зрозумілий веб-інтерфейс для керування базами даних, таблицями, стовпцями, зв'язками, індексами, користувачами, дозволами тощо. Користувачі можуть виконувати такі завдання, як виконання запитів SQL, імпортування та експортування даних, керування привілеями користувачів і оптимізація продуктивності бази даних.[]

Ще один додаток який покращить роботу та продуктивність нашої системи це серверне середовище, таке як Open Server, набір програмного забезпечення, яке включає в себе всі необхідні компоненти для розгортання та роботи з веб-додатками на вашому локальному комп'ютері. Такі середовища зазвичай включають в себе такі компоненти, як веб-сервер, базу даних, мову програмування та інші необхідні програми (див. рис. 2.5).

Воно включає в себе необхідні компоненти, такі як веб-сервер, мови програмування, бази даних та інші інструменти, які допомагають розробникам створювати та тестувати веб-додатки локально перед їх розгортанням на живому сервері.

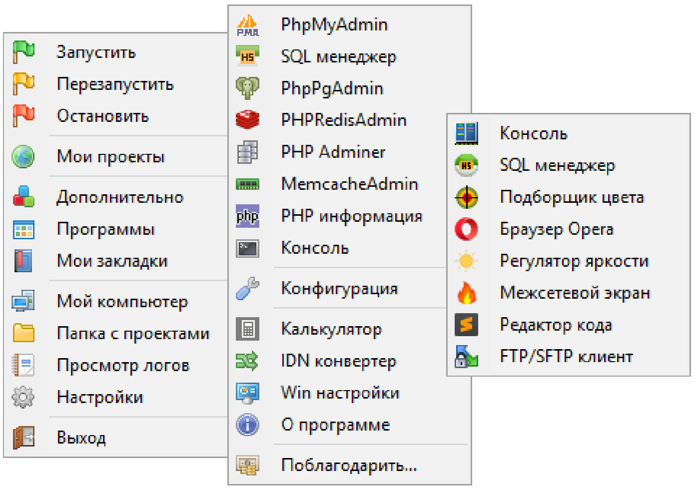

Рисунок 2.5 – додаток Open Server

2.4 Розробка навчальної платформи для підвищення продуктивності вивчення об’єктно-орієнтованої мови програмування Python

Спочатку авжеж необхідно розробити структури веб-сайту - це процес створення плану навігації користувача по сторінках чи секціях сайту з метою забезпечення зручного та швидкого переміщення. Основні вимоги до такої структури включають зрозумілі назви посилань, логічне розташування блоків, зрозумілий зв'язок між сторінками та легкий доступ до основних розділів сайту.

Було розроблено структуру головної сторінки веб-системи і відображено її у вигляді схеми (див. рис. 2.6).

Рисунок 2.6 - Структура головної сторінки веб-системи

На сайті є навігаційне меню, через яке користувач може перейти до головних розділів, таких як головна сторінка, посібник, редактор коду або свій аккаунт. Привітальний блок надає інформацію про поточне місцезнаходження користувача. У підвалі сайту міститься інформація про розробника та можливість зв'язку з ним у разі необхідності. Така структура зосереджена на основному контенті головної сторінки, що дозволяє користувачам легко зорієнтуватися і використовувати сайт.

Навчальна програма веб-сайту має вирішувати основну задачу - забезпечувати легкий доступ до інформації для користувачів будь-якого рівня. Незважаючи на відмінний дизайн, функціональність та продуктивність сайту, його основною метою є надання освітніх матеріалів. Сайт повинен простим та зрозумілим способом передавати інформацію, доступну для різних рівнів навичок користувачів. Теми повинні бути легкими для сприйняття, але при цьому містити всі необхідні аспекти, щоб навіть досвідчені користувачі могли вивчити новий матеріал.

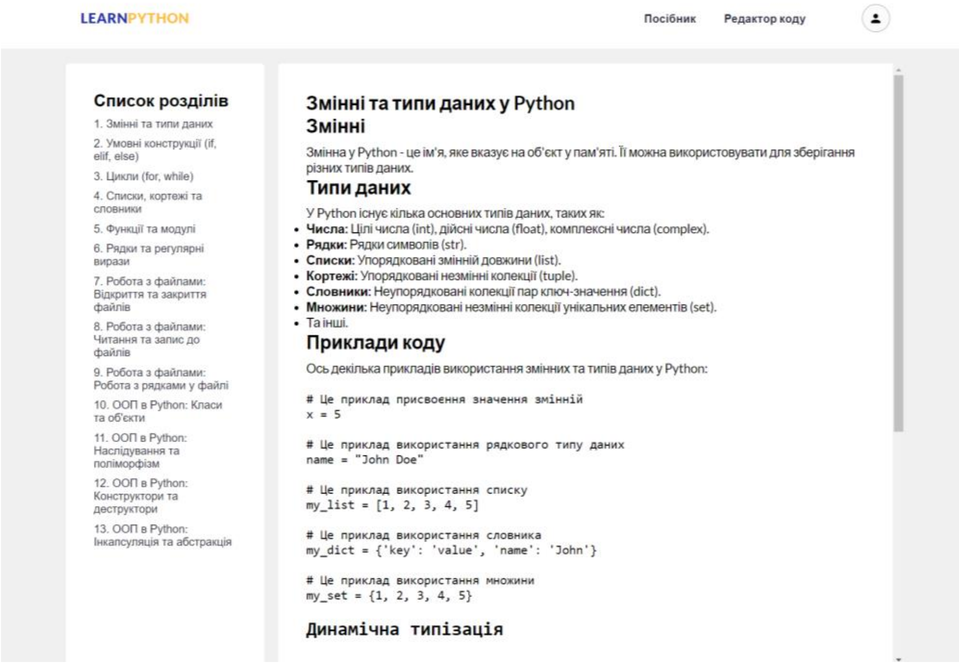

Рисунок 2.7 – Навчальна програма веб-системи

Навчальна програма зазвичай містить розділи, які охоплюють різні аспекти теми чи предмета, який вивчається. Кожен розділ може включати в себе такі елементи.

Вступний розділ надає загальний огляд теми або предмета, який буде вивчатися. Він може містити важливі визначення термінів, ключові концепції та мету навчання.

Теоретичний матеріал містить основну теоретичну інформацію про тему. Він може охоплювати основні принципи, правила, теорії та концепції, пов'язані з обраною темою.

Практичні завдання містять приклади або вправи, які допомагають у закріпленні теоретичного матеріалу та розвитку навичок.

Вся структура навчання знаходиться на сторінці Посібник веб-сайту (див. рис. 2.7).

2.5 Розробка алгоритму роботи веб-системи

Алгоритм - це послідовність конкретних інструкцій або кроків, які визначають послідовність дій, необхідних для виконання певного завдання або досягнення конкретної мети. У своєму найпростішому визначенні, алгоритм - це план дій, який дозволяє розв'язати певну проблему.

Алгоритм реєстрації/авторизації користувача полягає в перевірці наявності аккаунта користувача в системі (див. рис. 2.8). Якщо аккаунт існує, процес переходить до авторизації, в іншому випадку - до реєстрації. Під час авторизації користувач вводить логін та пароль, а система перевіряє їх на вірність. У разі невірних даних користувач повторно вводить їх. При реєстрації користувач повинен ввести логін, пароль, ім'я та прізвище. Після введення усіх даних система створює новий акаунт, якщо вони є правильними, інакше - просить користувача ввести дані знову.

Початок: Є акаунт?. Якщо так, перейти до кроку 10. Якщо ні, перейти до кроку 3.

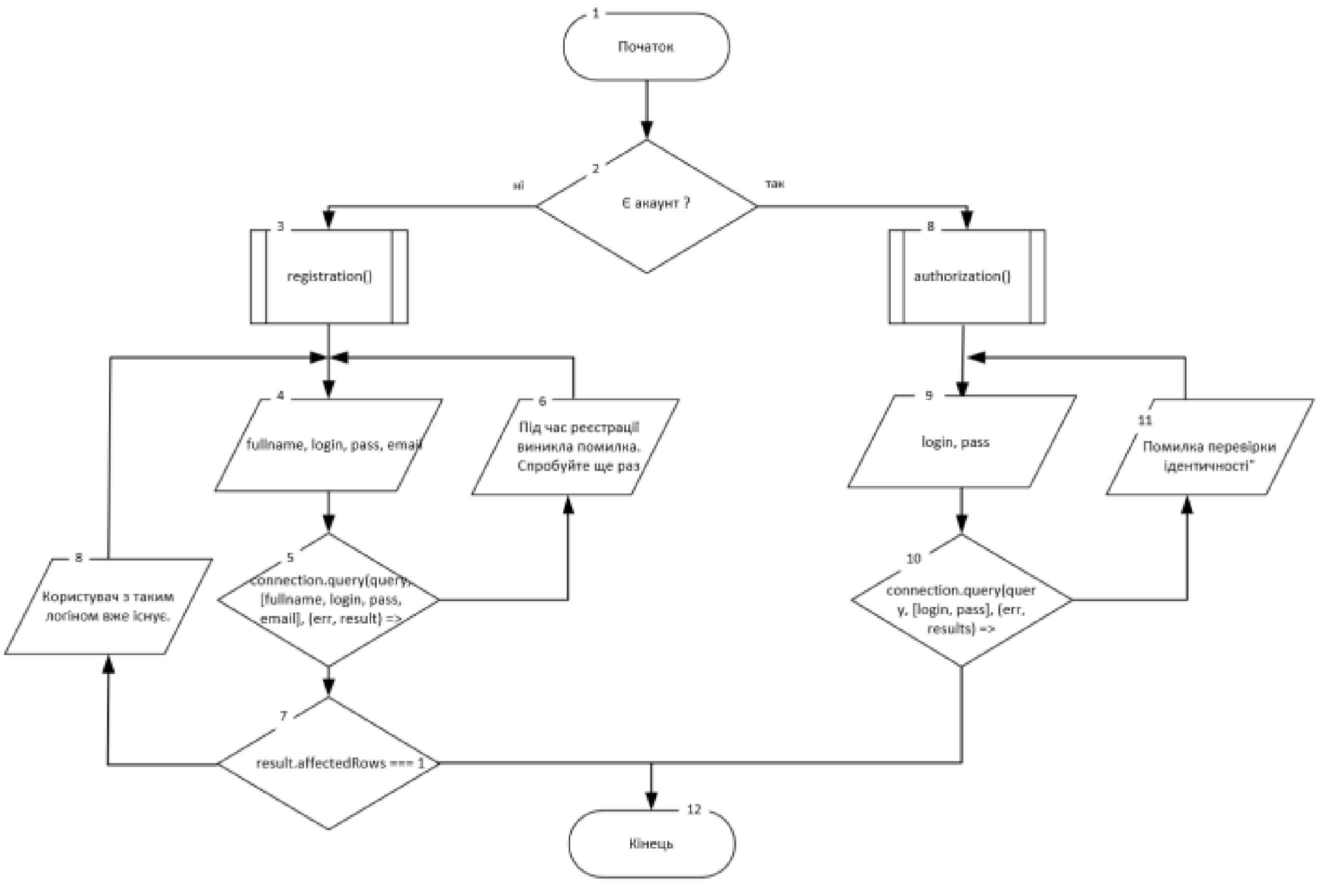

Рисунок 2.8 – Блок-схема алгоритму реєстрації/авторизації

Реєстрація: Користувач вводить свої дані: ім'я (fullname), логін (login), пароль (pass), email (email). Система перевіряє, чи є користувач з таким логіном вже зареєстрованим. Якщо так, вивести повідомлення "Користувач з таким логіном вже існує", перейти до кроку 8. Якщо ні, перейти до кроку 5. З'єднання з базою даних: Система формує SQL-запит query для реєстрації нового користувача. Виконання запиту: Запит query надсилається до бази даних. Якщо реєстрація пройшла успішно (result.affectedRows === 1), перейти до кроку 12. Якщо виникла помилка, вивести повідомлення "Під час реєстрації виникла помилка. Спробуйте ще раз", перейти до кроку 2.

Авторизація: Користувач вводить логін (login) та пароль (pass). Система формує SQL-запит query для авторизації користувача. Запит query надсилається до бази даних. Якщо авторизація пройшла успішно (result.length === 1), перейти до кроку 12. Якщо дані не вірні, вивести повідомлення "Помилка перевірки ідентичності.", перейти до кроку 2.

Завершення: Система повідомляє користувачу про успішну авторизацію або реєстрацію. Далі може йти перехід до іншого алгоритму, який описує дії користувача в системі.

Отже, було обгрунтовано вибір асинхронних HTTP-запитів як оптимального рішення. Обране телекомунікаційного обладнання, як MySQL, phpMyAdmin та Open Server. Наведені аргументи щодо важливості зрозумілості та доступності матеріалів для користувачів будь-якого рівня навичок. Розроблений алгоритм реєстрації/авторизації користувача, який надає послідовність дій для перевірки та обробки введених користувачем даних.

3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ПРОДУКТУ

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

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

JavaScript є однією з основних мов програмування для реалізації клієнтської частини веб-додатків. Оскільки браузери розуміють JavaScript, вона надає можливість взаємодіяти з DOM (Document Object Model), що є ключовим аспектом розробки клієнтської частини веб-додатків. Однією з найбільших переваг JavaScript у серверній розробці є можливість використовувати ту ж мову як на клієнтській, так і на серверній стороні. Це дозволяє розробникам створювати повністю інтегровані веб-додатки без необхідності вивчення різних мов програмування.

Незважаючи на те, що Python також може використовуватися для розробки веб-додатків (наприклад, за допомогою фреймворків Django або Flask), його вибір зазвичай пов'язаний з іншими вимогами проекту, такими як велика обчислювальна складність, наявність великого вже наявного коду на Python, або експертиза команди розробників. У кожному конкретному випадку слід враховувати вимоги проекту, технічні навички команди та інші фактори перед прийняттям рішення.

HTML - це мова розмітки гіпертексту, яка дозволяє визначити структуру веб-сторінки за допомогою тегів. Кожен тег має свою назву, яка вказує браузеру, який контент знаходиться всередині нього. Теги використовуються для відображення тексту, зображень, посилань та іншого контенту на сторінці. Крім того, вони можуть містити команди для браузера щодо відображення контенту або інструкції для пошукових систем. HTML обирається для розробки веб-системи для вивчення мови JavaScript через його здатність ефективно відображати контент на сторінці та відправляти інструкції браузеру для правильного відображення.

CSS, або каскадні таблиці стилів, дозволяють стилізувати HTML сторінки, надаючи їм візуальний зовнішній вигляд. За допомогою CSS можна змінювати фон, кольори тексту, розмір і форму блоків, а також створювати анімації та ефекти взаємодії.

Використання CSS обрано з метою зробити веб-сторінки більш привабливими та привертливими для користувачів, оскільки вони надають можливість створювати естетичний та привабливий дизайн, який привертає увагу.

JavaScript - це мова програмування, яка дозволяє надавати веб-сторінкам інтерактивності та динамічності. Вона забезпечує можливість додавання нового контенту на сторінку, обробки введених користувачем даних у формах, а також керування відображенням та стилями елементів. Без JavaScript сторінка обмежується лише переглядом, а взаємодія з нею обмежується переходом на інші сторінки. Використання цієї мови є необхідним для реалізації функцій, таких як пошук розділів, зберігання прогресу вивчення та реєстрація користувачів.

JavaScript має широку екосистему інструментів та бібліотек, які можуть бути використані для розробки веб-системи. Наприклад, існують багато фреймворків та бібліотек для створення користувацького інтерфейсу, таких як React.js, Angular, або Vue.js, які можна використовувати для покращення користувацького досвіду в навчальній системі.

Node.js - це платформа, яка дозволяє виконувати JavaScript-скрипти на сервері та відправляти результати їх виконання користувачеві. Вона надає можливість розробляти серверну частину веб-сайту на JavaScript, що дозволяє обробляти форми, працювати з базами даних та надсилати користувачам HTML-сторінки. Використання Node.js обрано через можливість використання JavaScript для серверної розробки, особливо для зв'язку з базою даних та обробки форм.

3.2 Розробка бази даних веб-системи для вивчення мови Python

Для організації контенту на веб-системі було прийнято рішення створити базу даних, яка буде зберігати різноманітну інформацію, таку як дані про користувачів, їх прогрес у проходженні тем, інформація про доступні теми та прогрес у виконанні завдань. Система управління базами даних (СУБД) є комплексом програм, які дозволяють створювати, зберігати та обробляти дані шляхом виконання операцій, таких як вставка, оновлення, видалення та вибірка. Вона забезпечує безпеку, надійність збереження та цілісність даних, а також має інструменти для адміністрування бази даних.

Архітектурно СУБД складається з двох великих компонент: мови опису даних та мови маніпулювання даними. Мова опису даних використовується для створення описів елементів, груп та записів даних, а також визначення взаємозв'язків між ними, які, як правило, подаються у вигляді таблиць. Мова маніпулювання даними використовується для виконання операцій з базою даних у прикладних програмах.

Фактична структура фізичного зберігання даних відома лише самій СУБД. Також, для роботи з базою даних, було обрано phpMyAdmin для MySQL, який є популярним інструментом для адміністрування баз даних MySQL.

phpMyAdmin - це безкоштовний веб-інтерфейс для управління базами даних MySQL, розроблений на мові програмування PHP. Він дозволяє адміністраторам баз даних легко виконувати різноманітні завдання, такі як створення, видалення, оновлення та керування таблицями, виконання SQL-запитів, імпорт і експорт даних, управління користувачами та їх привілеями, а також багато іншого.

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

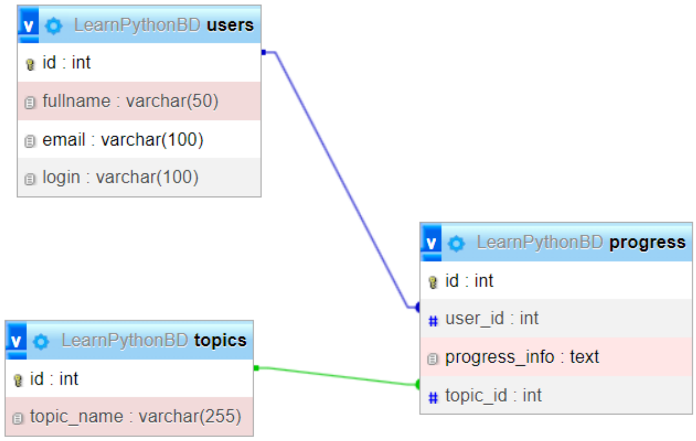

Рисунок 3.1 – ER-модель бази даних

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

SQL-запит:

SELECT * FROM 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 та вигляд сутності «user».

3.3 Розробка клієнтської частини веб-системи для вивчення мови Python

Клієнтська частина веб-сайту є тією частиною, яка відображається користувачеві у веб-браузері і відіграє велику роль у формуванні першого враження про сайт. Вона включає в себе всі елементи і компоненти, з якими користувач може взаємодіяти та переглядати інформацію.

При створенні клієнтської частини веб-сайту важливо керуватися стандартами та правилами, які є загальноприйнятими у веб-розробці. Наприклад, на кожному сайті повинен бути заголовок («дах»), де зазвичай розташовується логотип сайту та основна навігація. Також важливо мати "підвал" (футер), де може міститися інформація про розробника, контактна інформація, посилання на інструкції тощо.

Кожна сторінка веб-сайту поділена на різні блоки контенту, які мають в собі відповідний контент, очікуваний користувачем на перший погляд. Наприклад, на головній сторінці можуть бути блоки з акціями та пропозиціями, новини або каталог товарів.

Створений "дах" сайту є ключовим елементом, який забезпечує доступ користувача до основної навігації та інших важливих функцій сайту. Його дизайн та розташування повинні бути узгоджені зі загальним стилем та цілями веб-сайту, щоб забезпечити зручну та ефективну навігацію для користувачів.

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 рисунок 3.4

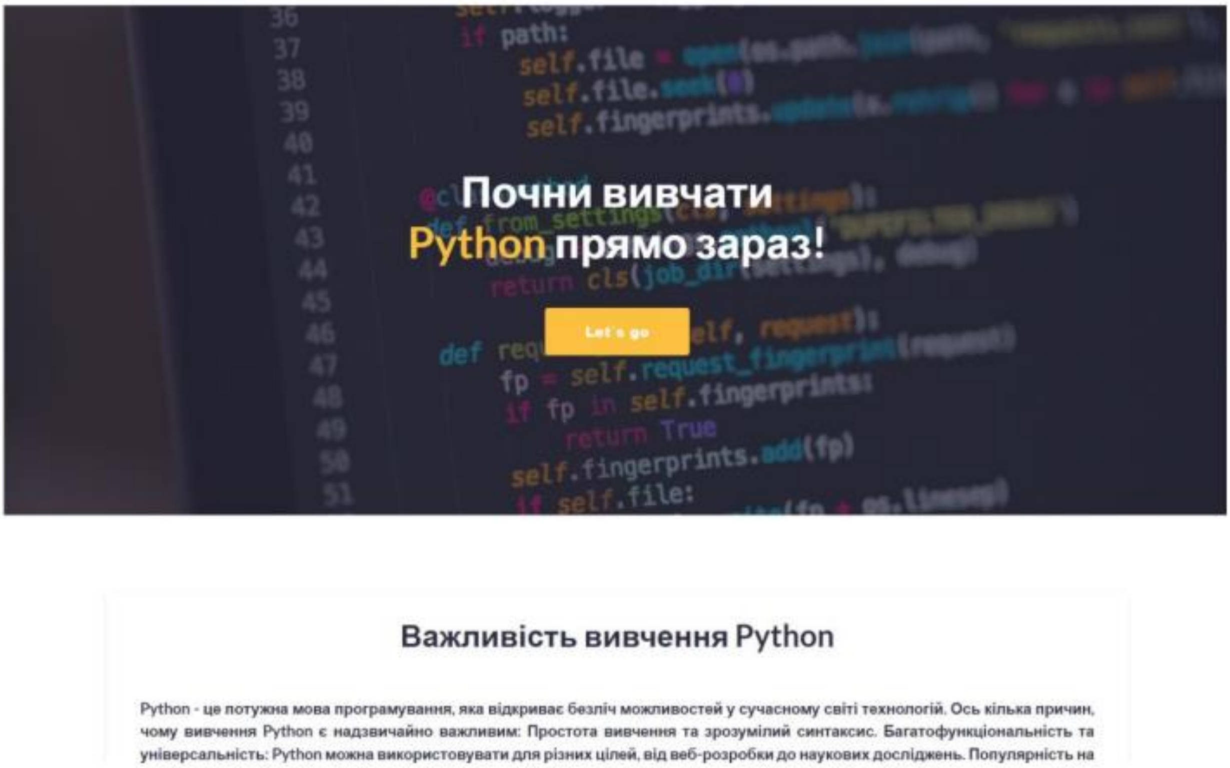

Рисунок 3.4 – Головна сторінка веб-сайту

Коли користувач переходить до сторінки «Аккаунт», йому надається можливість обрати між опціями авторизації, яка передбачає вхід у вже існуючий обліковий запис, або реєстрації, яка дозволяє створити новий обліковий запис на веб-сайті.

Ця сторінка, як показано на рисунку 3.5, надає користувачеві можливість здійснити вибір згідно його потреб і намірів.

В залежності від обраної користувачем опції, його буде автоматично перенаправлено на відповідну сторінку з формою. Якщо користувач обере опцію авторизації, він буде перенаправленний на сторінку з відповідною формою для введення облікових даних. Деталі цього процесу можна побачити на рисунку 3.6.

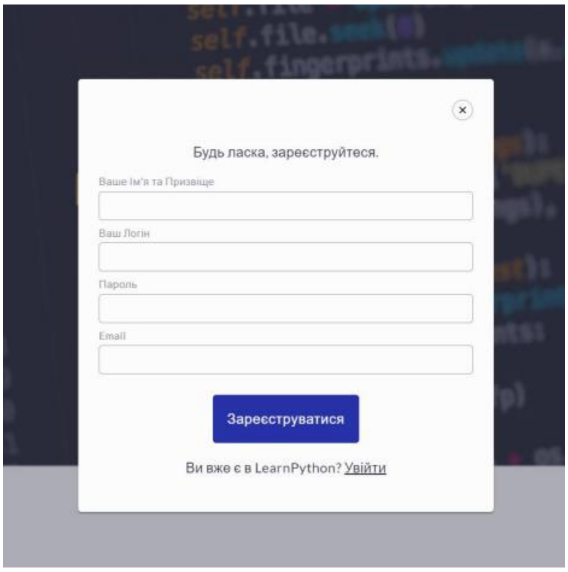

Рисунок 3.5 – Форма реєстрації користувача

Завдяки розробленню нового профілю користувача, тепер маємо можливість представити більш повну інформацію про кожного учасника.

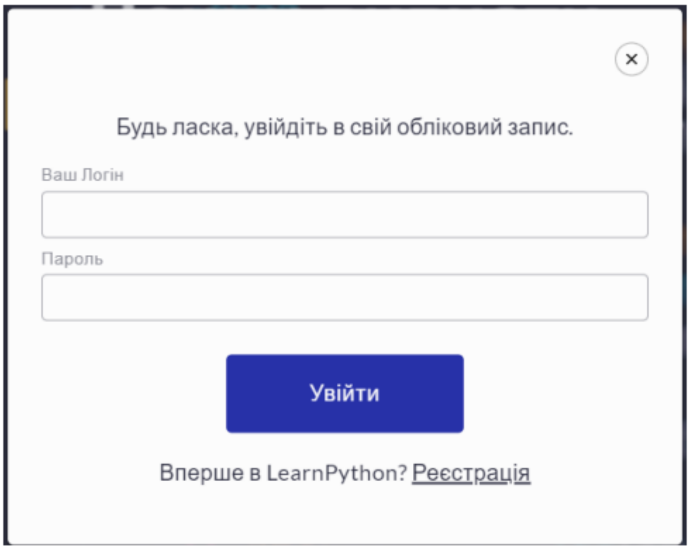

Рисунок 3.6 – Форма авторизації користувача

У цьому профілі відображаються не лише теми, які користувач успішно пройшов, а й його особисті дані, що робить його облік ще більш індивідуалізованим та зручним для використання.

Кожен профіль включає в себе ім'я, прізвище та логін користувача для легкого ідентифікування.

Окрім того, в ньому представлений перелік тем або курсів, які він успішно пройшов, що дозволяє нам отримати уявлення про його академічні досягнення та інтереси. Такий підхід допомагає покращити взаємодію користувачів з платформою, а також дозволяє нам більш точно адаптувати рекомендації та матеріали до їхніх потреб і рівня знань. (див. рис. 3.7);

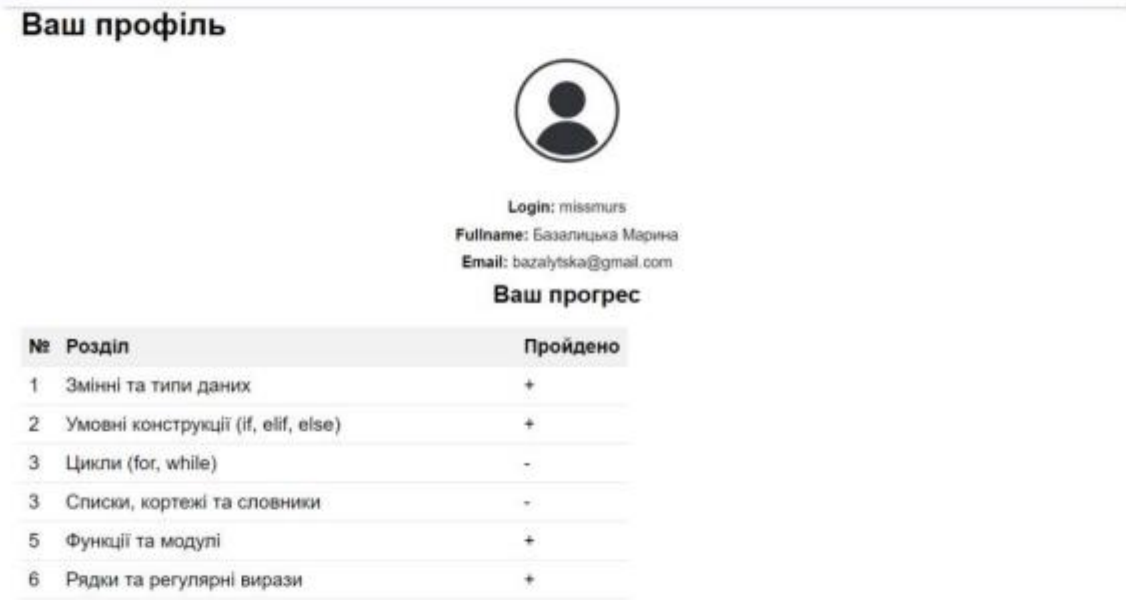

Рисунок 3.7– Профіль користувача

Після засвоєння теоретичного матеріалу з певного концепту або теми, студентам може бути надана можливість вирішити практичне завдання, яке передбачає застосування цих знань у конкретній ситуації. Такий підхід сприяє кращому закріпленню матеріалу і розвитку навичок роботи з конкретними завданнями.

Такий підхід допомагає студентам застосовувати теоретичні знання у реальних ситуаціях і розвивати навички самостійного розв'язання задачі після розділу наведено на рисунку 3.8

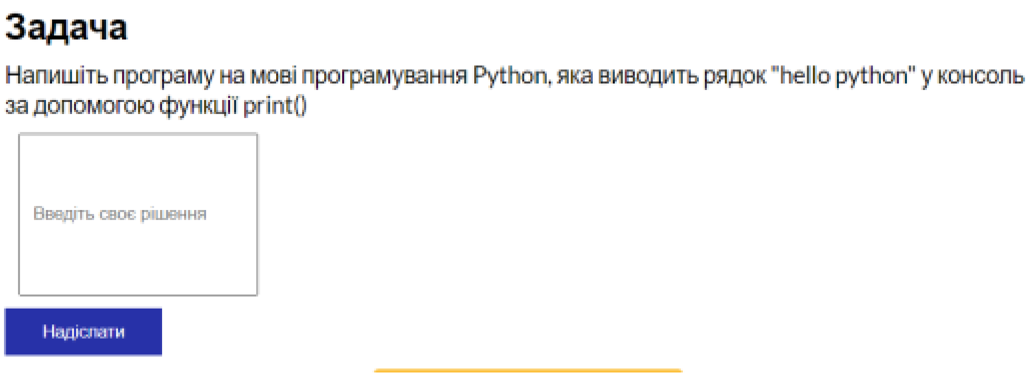

Рисунок 3.8 – Приклад задачі

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

3.4 Розробка серверної частини веб-системи для вивчення мови Python

Практично кожен веб-сайт потребує взаємодії з сервером. Це забезпечує можливість обробки форм, надісланих з клієнтської частини, перенаправлення користувачів на інші сторінки та забезпечення доступу до сайту через пошукові системи. Без наявності сервера доступ до веб-сайту можливий лише для розробника та осіб, яким надано відповідні права доступу до локальних файлів. Коли користувач заповнює форму на веб-сайті, дані з цієї форми надсилаються на сервер для обробки. Сервер може перевіряти дані форми, виконувати різні дії на основі цих даних, наприклад, додавати їх до бази даних або виконувати обчислення. Сервер може направляти користувачів на інші сторінки в залежності від їх дій. Наприклад, після успішного входу користувач може бути перенаправлений на його особистий кабінет або на домашню сторінку. Він також грає важливу роль у взаємодії веб-сайту з пошуковими системами. Він надає доступ до контенту веб-сайту для індексації пошуковими роботами, що дозволяє вашому веб-сайту з'являтися у пошукових результатах.

```
const mysql = require("mysql");
const express = require("express");
const ejs = require("ejs");

const app = express();
const topicsRouter = require("./topics");
app.use(topicsRouter);
const port = 7777;

app.set("view engine", "ejs");
app.use(express.static("public"));

app.get("/", (req, res) => {
  res.render("index");
});
app.get("/code", (req, res) => {
  res.render("code");
});

app.get("/profile", (req, res) => {
  res.render("profile");
});

const connection = mysql.createConnection({
  host: "127.0.0.1",
  user: "root",
  password: "",
  database: "LearnPython",
});

app.use(express.urlencoded({ extended: true }));
app.set("views", "./views");
app.set("view engine", "ejs");
app.use(express.static("public"));

connection.connect((err) => {
  if (err) {
    console.error("Error connecting to database:", err);
    return;
  }
  console.log("Connected to database successfully");
});

const register = require("./register");
app.use(register);
const login = require("./login");
app.use(login);
app.listen(port, () => {
  console.log(`Server started: http://localhost:${port}`);
});
```

Рисунок 3.9 – Лістинг коду створення локального серверу

Сервер є необхідним для розвитку будь-якого веб-сайту, оскільки він забезпечує можливість надавати користувачам різноманітний контент, такий як інформація, ігри або відео. Він також відповідає за обробку форм, виконання обчислень та інші функції, що можуть знадобитися для правильної роботи сайту.

Для забезпечення ефективного зв'язку між клієнтською та серверною частинами нашого веб-додатку використовується протокол HTTP, який є прикладним протоколом передачі даних у мережі. HTTP базується на технології "клієнт-сервер", де клієнт ініціює з'єднання, а сервер відповідає на запити та надсилає результати клієнту.

Основний файл проекту містить код для створення та підключення до сервера. Цей файл включає всі необхідні налаштування для переходів між різними сторінками та запуску самого сервера. Наприклад, на рисунку 3.9 показано, як створюється та запускається локальний сервер на порті 7777.

У першому рядку цього файлу імпортуються модулі mysql, express, та ejs. Модуль mysql використовується для взаємодії з базою даних MySQL, модуль express - для створення веб-сервера, а модуль ejs - для обробки шаблонів HTML. Змінна app створює новий екземпляр веб-сервера Express. Потім встановлюється двигун шаблонів EJS та вказується, що статичні файли (такі як CSS або JavaScript) знаходяться у директорії "public".

Далі встановлюються маршрути для головної сторінки (/), сторінки з кодом (/code), та сторінки профілю (/profile). При запитах на ці URL, вони рендерять відповідні шаблони EJS.

Потім встановлюється з'єднання з базою даних MySQL за допомогою модуля mysql. У цьому прикладі підключення налаштовано для локального сервера, користувача "root" без пароля та бази даних "LearnPython". Після цього підключаємо роутери, які відповідають за обробку POST-запитів для реєстрації та входу користувача. Нарешті, запускаємо сервер на певному порту (у даному випадку - порті 7777) і виводимо повідомлення про його успішний запуск у консоль.

Додатково, блок реєстрації користувача дозволяє користувачам реєструватися на веб-сайті, відвідавши відповідну сторінку і заповнивши необхідні поля форми. Після введення своїх даних вони надсилають форму. Для обробки цієї форми було створено контролер, який відповідає за обробку даних, введених користувачем і їх збереження у базі даних. Код наведено на рис. 3.10.

Також у додатку реалізовано блок авторизації користувачів, який дозволяє вже зареєстрованим користувачам увійти на веб-сайт, використовуючи свої облікові дані. Користувач може перебувати на сторінці авторизації, вводячи свій логін та пароль, і після натискання кнопки "Увійти" відправляти форму для перевірки введених даних. Код наведено на рис. 3.11.

```
const mysql = require("mysql");
module.exports = (req, res, next) => {
  if (req.method === "POST" && req.url === "/register") {
    const fullname = req.body.fullname;
    const login = req.body.login;
    const pass = req.body.pass;
    const email = req.body.email;

    const connection = mysql.createConnection({
      host: "127.0.0.1",
      user: "root",
      password: "",
      database: "LearnPython",
    });

    const query =
      "INSERT INTO user (fullname, login, pass, email) VALUES (?, ?, ?, ?)";

    connection.query(query, [fullname, login, pass, email], (err, result) => {
      if (err) {
        console.error(err);
        document.getElementById("message").innerText =
          "Під час реєстрації виникла помилка. Спробуйте ще раз.";
        return;
      }

      if (result.affectedRows === 1) {
        document.getElementById("message").innerText =
          "Ви успішно зареєструвалися!";
        res.redirect("/");
      } else {
        document.getElementById("message").innerText =
          "Користувач з таким логіном вже існує.";
      }
    });

    connection.end();
  } else {
    next();
  }
};
```

Рисунок 3.10 – Лістинг коду реєстрації користувача

Для обробки введених даних та здійснення процедури авторизації користувача був розроблений код, який включає в себе ряд важливих етапів, включаючи перевірку валідності введених даних та взаємодію з базою даних для перевірки правильності введеного логіну та пароля. Крім того, у коді реалізовані додаткові заходи безпеки, такі як перевірка пароля користувача на відповідність. Такий підхід дозволяє забезпечити рівень безпеки для користувачів та захистити їхні облікові дані від несанкціонованого доступу.

```
const mysql = require("mysql");
module.exports = (req, res, next) => {
  if (req.method === "POST" && req.url === "/login") {
    const login = req.body.login;
    const pass = req.body.pass;

    const connection = mysql.createConnection({
      host: "127.0.0.1",
      user: "root",
      password: "",
      database: "LearnPython",
    });

    const query = "SELECT * FROM user WHERE login = ? AND pass = ?";
    connection.query(query, [login, pass], (err, results) => {
      if (err) {
        console.error("Помилка перевірки ідентичності:", err);
        res.status(500).send("Помилка перевірки ідентичності");
        return;
      }

      if (results.length > 0) {
        res.redirect("/profile");
      } else {
        res.send("Невірний логін або пароль");
      }
    });
  }
};
```

Рисунок 3.11 – Лістинг коду авторизації користувача

Отже, було проаналізовано і обґрунтовано вибір засобів для реалізації веб-системи для вивчення мови програмування Python. 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 провести ґрунтовне тестування, щоб детально вивчити й перевірити реальну поведінку програми відносно її очікуваної роботи за різними сценаріями використання та певним набором тестів. Ця процедура має на меті виявлення всіх можливих недоліків, які можуть призвести до неприємного досвіду для користувачів, зокрема помилок у функціональності, інтерфейсі та продуктивності програмного забезпечення.

Основна мета тестування полягає у виявленні та виправленні дефектів, а також підвищенні рівня якості продукту перед його випуском на ринок. Послідовне вдосконалення продукту за допомогою тестування сприяє стабільності та надійності програмного забезпечення, забезпечуючи задоволення користувачів і підвищуючи їхню віддачу від використання продукту.

Тестування веб-сайту відрізняється від тестування іншого програмного забезпечення і включає в себе багато аспектів. Часто потрібно витратити час на перевірку оформлення та логічності інтерфейсу. Перш ніж переходити до тестування функціональності, важливо перевірити розташування блоків з контентом, їх розміри та логічну послідовність.

Для аналізу було обрано три методики функціонального тестування: тестування білого, сірого та чорного ящика.

При тестуванні білого ящика тестувальник має доступ до вихідного коду програми та може писати код, що пов'язаний з бібліотеками тестувального програмного забезпечення. Це особливо ефективно для модульного тестування, коли тестуються окремі частини системи.

При тестуванні чорного ящика тестувальник має доступ до програми тільки через її інтерфейс, аналогічно звичайному користувачеві. Цей метод зазвичай використовується на основі вимог до системи і дозволяє проводити як функціональне, так і нефункціональне тестування.

При тестуванні сірого ящика тестувальник має доступ до вихідного коду, але не використовує його напряму під час тестування. Цей метод поєднує в собі переваги обох попередніх методів.

Отже, після аналізу було вирішено використовувати метод тестування сірого ящика для перевірки правильності роботи веб-сайту. Це дає змогу ефективно створювати тест-кейси, при цьому не обов'язково розуміти внутрішні структури коду.

Проведено тестування перевірки правильності виконання задач. Задача з заповненим полем рішення від користувача представлена на рисунку 4.1.

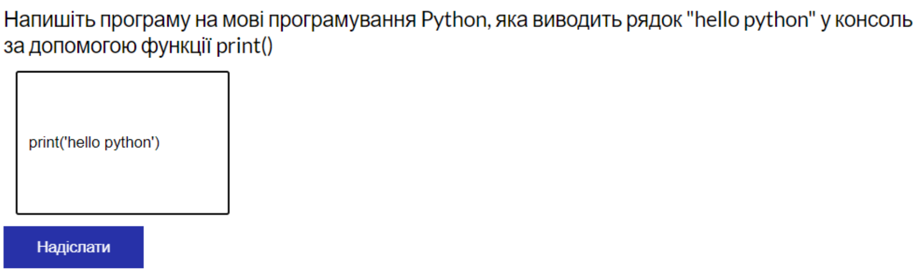

Рисунок 4.1 – Задача з введеним від користувача рішенням

Для тестування відповіді користувача було написано JavaScript код який представлено на рисунку 4.2

```
document.getElementById("submitBtn").addEventListener("click", function () {
  var solution = document.getElementById("solution").value.trim();

  var correctSolution = "print('hello python')";

  if (solution === correctSolution) {
    document.getElementById("result").innerText = "Вірно!";
    alert("Вірно!");
    document.getElementById("result").style.color = "green";
  } else {
    document.getElementById("result").innerText = "Невірно!";
    alert("Невірно!");
    document.getElementById("result").style.color = "red";
  }
});
```

Рисунок 4.2 – Лістинг коду для перевірки рішення

Код в прикладі було написано вірно, тому при натисканні кнопки відправити повинне з'явитися повідомлення «Вірно!». Результат тестування представлено на рисунку 4.3.

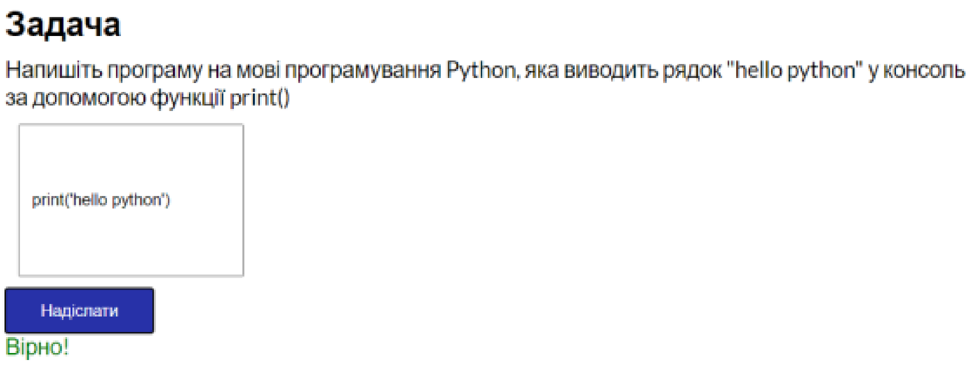

Рисунок 4.3 – Результат тестування

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

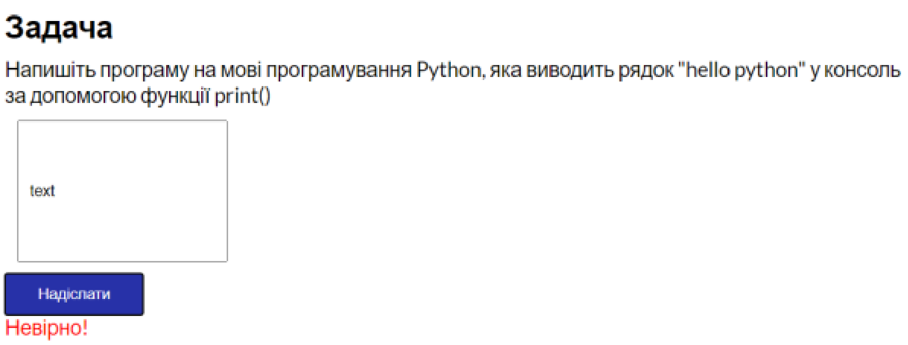

Рисунок 4.4 – Результат тестування з неправильним введеним рішенням

Отже, в рамках фінальних етапів розробки було ретельно проведено обширне тестування блоку, відповідального за перевірку задач в кінці кожного розділу програмного продукту. Ця процедура охоплювала не лише перевірку функціональності, а й враховувала широкий спектр можливих сценаріїв використання, щоб докладно вивчити реальну поведінку системи. Метою цього тестування було не лише виявлення потенційних недоліків, а й підвищення рівня якості цього конкретного функціоналу шляхом виправлення виявлених дефектів та оптимізації процесу перевірки задач.

4.2 Розробка настанов користувача

Інструкція користувача, іноді називана також гідом, представляє собою докладний опис функціональності та можливостей веб-системи, спрямованої на вивчення мови програмування Python. Цей документ включає в себе не лише пошагові інструкції з використання різних функцій системи, а й детальні описи їхнього призначення та потенційних сценаріїв застосування. В інструкції також можуть бути включені приклади коду, які демонструють конкретні методи програмування на Python, разом із зразками вхідних даних та очікуваними результатами. Крім того, інструкція користувача може містити інформацію щодо налаштування та конфігурації системи, а також поради щодо оптимального використання її можливостей з метою ефективного вивчення мови програмування Python.

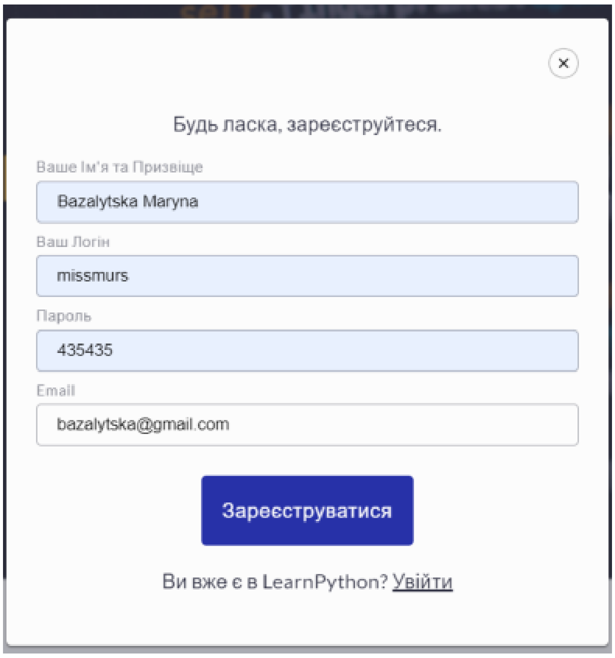

Рисунок 4.5 – Форма реєстрації з введеними даними

Для того, щоб зайти на сайт, потрібно натиснути на елемент навігаційної панелі «Акаунт». Сторінка відкривається з анкетою «Реєстрація» для створення облікового запису (див. рис. 4.5).

Після того, як користувач введе свої дані при реєстрації, система автоматично перевіряє наявність користувача з таким же логіном в базі даних. Якщо користувача з таким логіном не існує, його дані будуть автоматично додані до бази даних після натискання кнопки "Зареєструватися".

Цей процес гарантує, що кожен користувач має унікальний логін, що дозволяє ефективно управляти аккаунтами та забезпечує безпеку особистих даних. Після успішної реєстрації відбувається автоматичне оновлення бази даних з новоствореним аккаунтом, яке можна побачити на відповідному розділі інтерфейсу, як показано на рисунку 4.6.

Рисунок 4.6 – Доданий до бази даних користувач

При обранні варіанту "Увійти", користувач переходить на відповідну сторінку, де йому необхідно буде ввести існуючий логін та пароль свого облікового запису для входу в систему. Після введення логіну та пароля і подальшої аутентифікації користувача, він отримає доступ до свого облікового запису та пов’язаних з ним можливостей відповідно до його ролі та налаштувань (див. рис. 4.7).

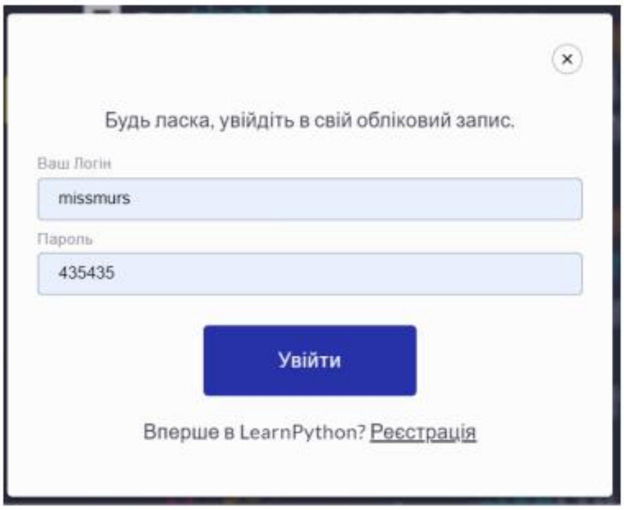

Рисунок 4.7 – Авторизація у веб-системі

Посібник працює наступним чином: зліва розташований перелік розділів, наприклад, тем або глав. Коли користувач натискає на один з цих розділів, з правого боку відображається відповідна інформація про обраний розділ.

Це організовано для зручності навігації та доступу до конкретної інформації. Користувач може швидко знайти необхідний розділ і переглянути його зміст без необхідності прокручувати весь документ або шукати потрібну інформацію в інших місцях (див. рис. 4.8).

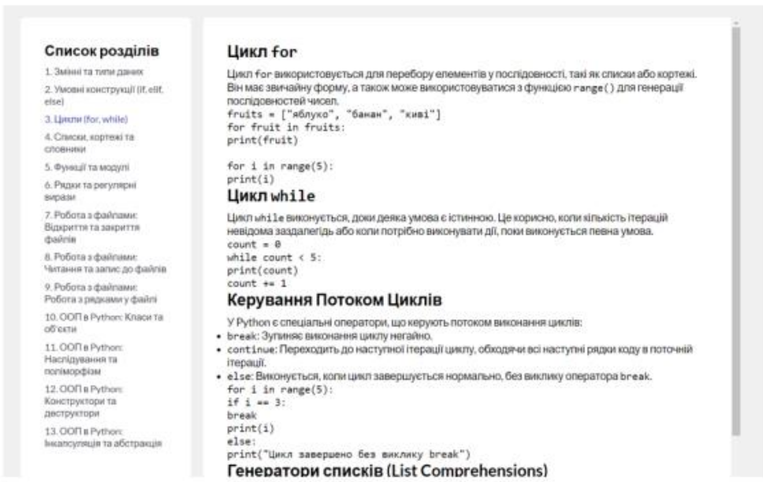

Рисунок 4.8 – Сторінка освітнього матеріалу

5.ОХОРОНА ПРАЦІ

Під час дослідження веб-застосунок для вивчення мови програмування Python на основі протоколу http на працівника могли мати вплив такі небезпечні та шкідливі виробничі фактори:

1. Фізичні:

– підвищена запиленість та загазованість повітря робочої зони;
– підвищений рівень шуму на робочому місці;
– підвищена чи понижена вологість повітря;
– підвищений рівень електромагнітного випромінювання;
– підвищена чи понижена іонізація повітря;
– недостатня освітленість робочої зони;
– підвищена яскравість світла; понижена контрастність;
– пряма і відбита блисткість.

2. Психофізіологічні: статичне перевантаження та розумове перевантаження.

Відповідно до наведених факторів здійснюємо розробку заходів щодо безпечного виконання поставленого завдання.

5.1. Технічні рішення щодо безпечного виконання роботи

Вимоги до організації та обладнання робочих місць повинні включати наступне:

Забезпечення достатньої площі та об’єму для кожного робочого місця, що становить не менше 6 квадратних метрів площі та 20 кубічних метрів об’єму.

Конструкція робочого місця повинна сприяти підтриманню оптимальної робочої позиції, яка дозволяє працівнику виконувати роботу з мінімальним напруженням тіла. Це важливо для запобігання перевтоми та збереження здоров’я працівника.

Раціональна робоча поза є ключовою для запобігання захворювань, таких як сколіоз, варикозне розширення вен, плоскостопість тощо. Робота в зігнутому або значно нахиленим положенні може збільшити затрати енергії на 20% або навіть на 45% порівняно з прямим положенням корпуса.

Таким чином, важливо враховувати ці аспекти при організації та обладнанні робочих місць для забезпечення комфортних та безпечних умов праці [2].

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м.

Робочі місця слід розташовувати відносно джерела природного світла (вікон) таким чином, щоб світло падало збоку, переважно зліва. Також робоче місце має відповідати сучасним вимогам ергономіки:

– стіл повинен мати висоту поверхні 680 - 800 мм., ширину 600 - 1400 мм. і глибину 800 - 1000 мм. (такі параметри забезпечують можливість виконання операцій в зоні досяжності працівника);

– робочий стілець робочий стілець має бути підйомно-поворотним, з можливістю регулювання висоти, бажано зі стаціонарними або змінними підлікотниками і напівм’якою нековзкою поверхнею сидіння, що легко чиститься і не електризується;

– екран комп’ютера має розташовуватися на оптимальній відстані від користувача, що становить 600 – 700 мм., але не менше за 600 мм. з урахуванням літерно-цифрових знаків і символів.

Приміщення, де здійснювалося дослідження використання веб-застосунок для вивчення мови програмування Python на основі протоколу http за небезпекою ураження електричним струмом належить до приміщень без підвищеної небезпеки (сухе, мало запорошене, з нормальною температурою повітря, ізольованими підлогами і малим числом заземлених приладів).

Персональні комп’ютери, периферійні пристрої, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію.

Лінія електромережі для живлення персональних комп’ютерів і периферійних пристроїв виконується як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Усі провідники відповідають номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння людини під напругу.

Персональні комп’ютери і периферійні пристрої підключаються до електромережі тільки за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення. У штепсельних з’єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз’єднання при відключенні має бути зворотним.

5.2. Технічні рішення з гігієни праці та виробничої санітарії

5.2.1. Мікроклімат

Стан повітря робочої зони у виробничому приміщенні називають мікрокліматом або метеорологічними умовами. Мікроклімат або метеорологічні умови виробничих приміщенні, визначаються за такими параметрами:

– температурою повітря у приміщенні, С;
– відносною вологістю повітря, %;
– рухливістю повітря, м/с;
– тепловим випромінюванням, Вт/м3.

Всі ці параметри поодинці, а також у комплексі впливають на фізіологічну функцію організму його терморегуляцію і визначають самопочуття.

Дослідження веб-застосунок для вивчення мови програмування Python на основі протоколу http за енерговитратами відноситься до категорії I а (енерговитрати до 139Дж/с) [3]. Допустимі параметри мікроклімату для цієї категорії згідно ДСН 3.3.6.042-99 [4] наведені в табл.5.2.1.

Таблиця 5.2.1 – Параметри мікроклімату

Період року	Допустимі		
	t,°C	W, %	V, м/с
Теплий	22-28	55	0,1-0,2
Холодний	21-25	75	0,1

Для створення і автоматичної підтримки в приміщенні незалежно від зовнішніх умов допустимих значень температури, вологості, чистоти і швидкості руху повітря обладнані системами опалення та кондиціонування повітря. Систематично проводиться вологе прибирання.

5.2.2. Склад повітря робочої зони

Оточуюче нас повітря (атмосфера) є найважливішим фактором забезпечення життя. В природних умовах повітря, як правило, не забруднене отруйними речовинами і життю людини не загрожує. Органи чутливості людини не дозволяють з достатньою точністю визначати якість повітря і запобігати загрозі отруєння.

В приміщенні, де здійснюється дослідження веб-застосунок для вивчення мови програмування Python на основі протоколу http, у повітрі можуть перевищувати ГДК такі речовини як вуглекислий газ, пил та озон. Джерелами цих речовин є офісна техніка. Пил потрапляє у приміщення ззовні через відкриті вікна та заноситься на одязі і взутті працівниками.

ГДК шкідливих речовин, які знаходяться в досліджуваному приміщені, наведені в таблиці 5.2.2.

Таблиця 5.2.2 – ГДК шкідливих речовин у повітрі

Назва речовини	ГДК, мг/м^3		Клас небезпечності
	Максимально разова	Середньо добова	
Вуглекислий газ	3	1	4
Пил нетоксичний	0,5	0,15	4
Озон	0,16	0,03	4

Рівні позитивних і негативних іонів у повітрі мають відповідати санітарно-гігієнічним нормам (табл..5.2.3).

Таблиця 5.2.3 – Рівні іонізації повітря приміщень при роботі на ПК

Рівні	Кількість іонів в 1 см^3	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально необхідні	50000	50000

Для підтримки допустимих значень мікроклімату та концентрації позитивних та негативних іонів необхідно передбачати установки або прилади зволоження або штучної іонізації, кондиціювання повітря.

5.2.3. Виробниче освітлення

Освітлення відіграє важливу роль у житті людини. Біля 90% інформації сприймається через зоровий канал, тому правильно виконане раціональне освітлення має важливе значення для виконання всіх видів робіт. Недостатня освітленість або її надмірна кількість знижують рівень збудженості центральної нервової системи і, природно, активність усіх життєвих процесів. Раціональне освітлення є важливим фактором загальної культури виробництва. Неможливо забезпечити чистоту та порядок у приміщені, в якому напівтемрява, світильники брудні або в занедбаному стані.

Приміщення, в яких встановлені персональні комп’ютери, повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2018 [5]. Норми освітленості при штучному освітленні та КПО (для III пояса світлового клімату) при природному та сумісному освітленні зазначені у таблиці 5.2.4:

Таблиця 5.2.4 - Норми освітленості в приміщенні

<table>
<tr><th rowspan="3">Характеристика зорової роботи</th><th rowspan="3">Найменший розмір об'єкта розрізнювання</th><th rowspan="3">Розряд зорової роботи</th><th rowspan="3">Підрозряд зорової роботи</th><th rowspan="3">Контраст об'єкта розрізнення з фоном</th><th rowspan="3">Характеристика фона</th><th colspan="2">Освітленість, лк</th><th colspan="4">КПО, $e_н$,%</th></tr>
<tr><th colspan="2">Штучне освітлення</th><th colspan="2">Природнє освітлення</th><th colspan="2">Сумісне освітлення</th></tr>
<tr><th>Комбіноване</th><th>Загальне</th><th>Верхнє або верхнє</th><th>Бокове</th><th>Верхнє або верхнє і бокове</th><th>Бокове</th></tr>
<tr><td>Дуже високої точності</td><td>Від 0,15 до 0,3</td><td>II</td><td>г</td><td>великий</td><td>світлий</td><td>1000</td><td>300</td><td>7</td><td>2,5</td><td>4,2</td><td>1,5</td></tr>
</table>

Штучне освітлення в досліджуваному приміщенні здійснюється системою загального рівномірного освітлення. У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення додатково встановлюються світильники місцевого освітлення).

Для забезпечення достатнього освітлення передбачені такі заходи:

1) Максимальне використання бічного природного освітлення.

2) Систематичне очищення скла від бруду – не рідше двох разів на рік.

3) Штучне освітлення в приміщенні забезпечується світильниками типу РСП08×250 (однолампові) з лампами ДРЛ-250.

5.2.4. Виробничий шум

Виробничий шум – це сукупність різних за гучністю і тоном звуків, які виникають у повітряному середовищі. В досліджуваному приміщенні наявний як постійний, так і непостійний шуми. Нормування непостійного шуму, а також орієнтовна оцінка загального рівня постійного шуму здійснюється скоректованим за частотою загальним рівнем звукового тиску – так званим рівнем звуку, який вимірюється в дБА за шкалою «А» шумоміра.

Непостійний шум характеризується еквівалентним рівнем звуку LA екв., що являє собою середньоквадратичний рівень звуку непостійного шуму, який має такий самий вплив на людину, як і постійний шум.

Для умов виконання роботи допустимі рівні звукового тиску не повинні перевищувати 50 дБА (табл.5.2.5).

Таблиця 5.2.5 – Допустимі рівні звукового тиску і рівні звуку для постійного широкополосного шуму (згідно ДСН 3.3.6.037-99 [6])

Характер робіт	Допустимі рівні звукового тиску (дБ) в стандартизованих октавних смугах зі середньогеометричними частинами (Гц)									Допуст-мий рівень звуку, дБА
	32	63	125	250	500	1000	2000	4000	8000	
Виробничі приміщення	86	71	61	54	49	45	42	40	38	50

Рівень шуму в приміщенні не перевищує допустимих значень.

5.2.5. Виробничі випромінювання

Розрізняють природні та штучні джерела електромагнітних полів (ЕМП). У процесі еволюції біосфера постійно перебуває під впливом ЕМП природного походження (природний фон): електричне та магнітне поля Землі, космічні ЕМП, передусім ті, що генеруються Сонцем. У період науково-технічного прогресу людство створило і все ширше використовує штучні джерела ЕМП. У теперішній час ЕМП антропогенного походження значно перевищують природний фон і є тим несприятливим чинником, чий вплив на людину з року в рік зростає.

Значення напруженості електростатичного поля на робочих місцях (як у зоні екрана дисплея, так і на поверхнях обладнання, клавіатури, друкувального пристрою) мають не перевищувати гранично допустимих за ГОСТ 12.1.045-84 [7]. Значення напруженості електромагнітних полів на робочих місцях з ВДТ мають відповідати нормативним значенням (ДСанПіН 3.3.6-2002 [8], ГОСТ 12.1.045-84 [7]). Інтенсивність потоків інфрачервоного випромінювання має не перевищувати допустимих значень відповідно до ГОСТ 12.1.005-88 [9] (табл.5.2.6).

Таблиця. 5.2.6 - Допустимі параметри електромагнітних неіонізуючих випромінювань і електростатистичного поля

Види поля	Допустима поверхнева щільність потоку енергії, Вт/кв.м	
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру УФ-С (220 — 280 мм)	0,001	
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру УФ-В (280 — 320 мм)	0,01	
Електромагнітне поле оптичного діапазону в ультрафіолетовій частині спектру УФ-А (320 — 400 мм)	10,0	
Електромагнітне поле оптичного діапазону в видимій частині спектру 400 — 760 мм	10,0	
Електромагнітне поле оптичного діапазону в інфрачервоній частині спектру 0,76 — 10,0 мкм	35,0 — 70,0	
Напруженість електричного поля відеодисплейного терміналу	20кВ/м	

Для дотримання наведених нормативів слід використовувати офісну техніку з сертифікатом якості та дотримуватися встановлених режимів праці та відпочинку з ПК.

5.3. Пожежна безпека

Пожежна безпека – стан об’єкта, при якому з регламентованою ймовірністю відкидається можливість виникнення та розвиток пожежі, і впливу на людей її небезпечних факторів, а також забезпечується захист матеріальних цінностей.

З урахуванням ДСТУ Б В.1.1-36:2016 [10] приміщення, де здійснюється дослідження веб-застосунок для вивчення мови програмування Python на основі протоколу http відноситься до категорії вибухопожежонебезпеки В (тверді горючі та важкогорючі речовини і матеріали, за умови, що приміщення, в яких вони знаходяться, не відносяться до категорій А, Б і питома пожежне навантаження для твердих і рідких легкозаймистих та горючих речовин на окремих ділянках площею не менше 10 $м^2$ кожна перевищує 180 $МДж/м^2$).

Клас приміщення і зон з вибухо- і пожежонебезпеки – П-IIa (приміщення, у якому знаходяться тверді горючі речовини та матеріали) [11].

5.3.1. Технічні рішення системи запобігання пожежі

Усі заходи системи запобігання пожежі в приміщенні можна поділити на організаційні, технічні, режимні та експлуатаційні.

Організаційні заходи пожежної безпеки передбачають: організацію пожежної охорони на об’єкті, проведення навчань з питань пожежної безпеки (включаючи інструктажі та пожежно-технічні мінімуми), застосування наочних засобів протипожежної пропаганди та агітації, проведення перевірок, оглядів стану пожежної безпеки приміщень, будівель, об’єкта в цілому та ін.

До технічних заходів належать: суворе дотримання правил і норм, визначених чинними нормативними документами при реконструкції приміщень, будівель та об’єктів, технічному переоснащенні виробництва, експлуатації чи можливому переобладнанні електромереж, опалення, вентиляції, освітлення і т. ін.

Заходи режимного характеру передбачають заборону куріння та застосування відкритого вогню в недозволених місцях, недопущення появи сторонніх осіб у вибухонебезпечних приміщеннях чи об'єктах, регламентацію пожежної безпеки при проведенні вогневих робіт тощо.

Експлуатаційні заходи охоплюють своєчасне проведення профілактичних оглядів, випробувань, ремонтів технологічного та допоміжного устаткування, а також інженерного господарства (електромереж, електроустановок, опалення, вентиляції)

5.3.2. Технічні рішення системи протипожежного захисту

Система протипожежного захисту – це сукупність організаційних заходів а також технічних засобів, спрямованих на запобігання впливу на людей небезпечних чинників пожежі та обмеження матеріальних збитків від неї.

До основних засобів пожежогасіння, що використовуються в приміщеннях такого класу відносять вогнегасники, розміщення яких в приміщеннях позначається встановленим поруч вказівним знаком

Згідно категорії пожежовибухонебезпеки будівлі, класу приміщення і за вибухо- і пожежонебезпекою П-IIа та Наказу Міністерства внутрішніх справ України «Про затвердження Правил експлуатації та типових норм належності вогнегасників», в приміщенні має бути встановлено 2 порошкових або вуглекислотних вогнегасники із зарядом речовини від 3 до 5 кг. [12]

В цілому приміщення по категорії вибухо- і пожежонебезпечності та ступеню вогнестійкості відповідає нормам, але особливу увагу потрібно звернути на утримання в справному стані засобів протипожежного захисту та своєчасне інформування пожежної охорони про несправність пожежної техніки, впровадження систем протипожежного захисту.

ВИСНОВКИ

В результаті виконання бакалаврській дипломної роботи, спрямованої на розробку програмного застосунку для підвищення продуктивності вивчення об'єктно-орієнтованої мови програмування, було важливо систематично аналізувати сучасний стан галузі. З урахуванням широкого спектру можливих методів розв'язання проблеми, було проведено порівняльний аналіз аналогів та вибрано найбільш оптимальний метод розробки веб-системи з самого початку.

Одним із ключовим факторів роботи було обґрунтування вибору використання HTTP протоколу, що дозволяє підвищити продуктивність та ефективність системи. Також було важливо обрати відповідне телекомунікаційне обладнання для забезпечення надійності та швидкодії бази даних, тому були обрані MySQL, phpMyAdmin та Open Server.

Під час розробки веб-системи, особлива увага приділялася зрозумілості та доступності для користувачів будь-якого рівня навичок. Це стало можливим завдяки ретельній розробці алгоритму реєстрації/авторизації користувача та використанню зрозумілих інтерфейсів на клієнтській та серверній частинах веб-системи.

Для створення веб-системи було обрано такі технології, як HTML, CSS, JavaScript та Node.js, що дозволило забезпечити не лише функціональність, але й зручність у використанні. Було проведено тестування веб-системи з використанням методу "сірого ящика" для забезпечення якості та надійності роботи програми.

Крім того, визначено параметри охорони праці для вказаних умов і характеристик навколишнього середовища, включаючи оптимальні рівні освітленості, температури, вологості, вентиляції та шуму.

У підсумку, розробка зуміла обґрунтувати свій вибір та здійснити реалізацію веб-системи, яка відповідає сучасним вимогам продуктивності та ефективності вивчення об'єктно-орієнтованої мови програмування в контексті телекомунікаційних систем. Такий підхід дозволяє забезпечити якісне навчання користувачів та підвищити їхню компетентність у використанні Python.

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. Кормен, Ч., Лейрсон, Р., Рівест, К., Стейн. Вступ до алгоритмів / Т.. – Київ: Наш Формат, 2019. – 1312 с.
3. Маккінні В. Python для аналізу даних / В. Маккінні. - Київ: Наш Формат, 2018. - 496 с.
4. Маттес Е. Python Crash Course / Київ: Наш Формат, 2019. – 432 с.
5. Мейер Е., Уейл Е. CSS повний довідник. / Е. Мейер, Е. Уейл – Київ: Діалектика, 2017. 1088 с.
6. Роббінс Д. HTML5: кишеньковий довідник 5-е видання. / Д. Роббінс – Київ: Діалектика, 2015. 192 с.
7. Романюк О. Н. Організація баз даних і знань [Текст] : навчальний посібник / О. Н. Романюк, Т. О. Савчук. - Вінниця : УНІВЕРСУМ-Вінниця, 2003. – 217 с.
8. Хавербек, М. Гарний JavaScript: Сучасний вступ до програмування / Київ: Видавничий дім "Інтерсервіс", 2019. – 456 с.
9. Філдінг, Р., та Тейлор, Р. (2000). Архітектурні стилі та проектування програмного забезпечення на основі мережі.
10. Фетт, І., та Мельніков, А. (2011). Протокол WebSocket. RFC 6455.
11. Річардсон, Л., та Амундсен, М. (2013). RESTful веб-сервіси.
12. Codecademy. [Електронний ресурс]. – Режим доступу: https://www.codecademy.com/

a. Real Python. [Електронний ресурс]. – Режим доступу: https://realpython.com/

b. SoloLearn. [Електронний ресурс]. – Режим доступу: https://sololearn.com /

13. Що таке Node.js. [Електронний ресурс]. – Режим доступу: https://uk.theastrologypage.com/node-js
14. MySQL Workbench. [Електронний ресурс]. – Режим доступу:
15. https://www.mysql.com/products/workbench
16. ДСТУ OHSAS 18002:2015. Системи управління гігієною та безпекою праці. Основні принципи виконання вимог OHSAS 18001:2007 (OHSAS 18002:2008, IDT). К. : ГП «УкрНИУЦ», 2016. 21 с.
17. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями. URL:

http://sop.zp.ua/norm_npaop _0 _ 00-7_ 15-18_01_ua.php.

18. Гігієнічна класифікація праці (за показниками шкідливості і небезпеки факторів виробничого середовища від 12.08.1986 № 4137-86. - [Електронний ресурс] - Режим доступу: http://zakon4.rada.gov.ua/laws/show/v4137400-86

19. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. - [Електронний ресурс] - Режим доступу: http://mozdocs.kiev.ua/view.php?id=1972

20. ДБН В.2.5-28-20018 Природне і штучне освітлення - [Електронний ресурс] - Режим доступу: http://document.ua/prirodne-i-shtuchne-osvitlennja-nor8425.html

21. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. - [Електронний ресурс] - Режим доступу: http://document.ua/sanitarni-normi-virobnichogo-shumu-ultrazvuku-ta-infrazvuku-nor4878.html

22. ДСНіП3.3.6.096-2002. Державні санітарні норми і правила при роботі з джерелами електромагнітних полів. URL: http://zakon2.rada.gov.ua/laws/show/ z0203-03.

23. ДСанПіН 3.3.6-2002 Державні санітарні норми і правила при роботі з джерелами електромагнітних полів - [Електронний ресурс] - Режим доступу: http://zakon.nau.ua/doc/?code=z0203-03

24. НПАОП 0.00-7.11-12. Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників. URL: http://zakon.rada.gov.ua/laws/show/ z0226-12.

25. ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпек. URL: https://dbn.co.ua/load/normativy/dstu/dstu_b_v_1_1_36/5-1-0-1759.

26. ДБН В.1.1-7:2016 Пожежна безпека об`єктів будівництва. Загальні вимоги. URL: http://www.poliplast.ua/doc/dbn _v.1.1-7-2002.pdf.

27. Наказ Міністерства внутрішніх справ України «Про затвердження

Правил експлуатації та типових норм належності вогнегасників». URL: https://zakon.rada.gov.ua/laws/show/z0225-18#Text.

Додаток А
(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ВЕБ-ЗАСТОСУНОК ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ PYTHON НА ОСНОВІ ПРОТОКОЛУ HTTP
назва бакалаврської дипломної роботи

Таблиця 1 – Порівняння аналогів

Критерії оцінювання	realpython	sololearn	codecademy	learnpython
Наявність тестувань	-	+	+	+
Структуроване навчання	-	-	-	+
Збереження прогресу навчання	-	+	+	+
Можливість створення аккаунта	+	+	+	+
Редактор коду	-	-	+	+

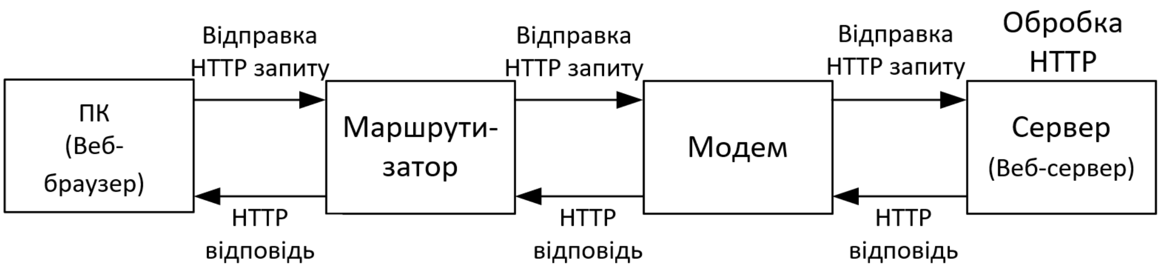

Рисунок 1 – Блок схема організації зв’язку

Рисунок 2 - Структура головної сторінки веб-системи

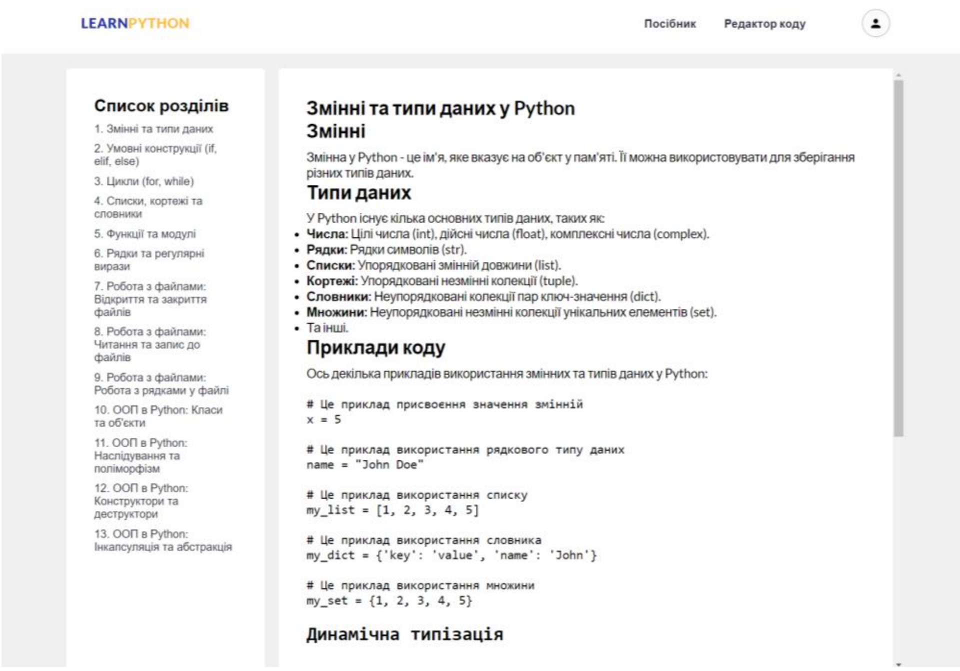

Рисунок 3 – Навчальна програма веб-системи

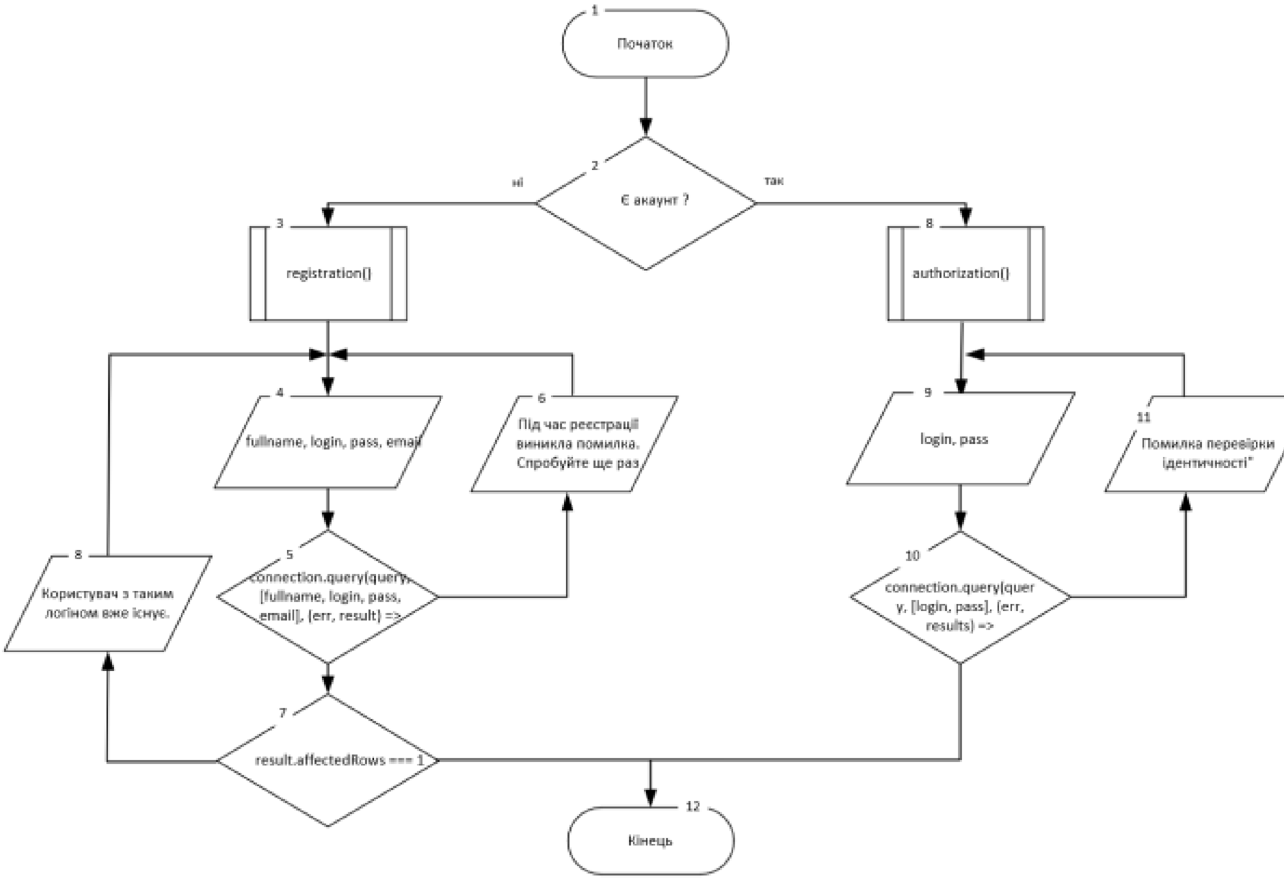

Рисунок 4 – Блок-схема алгоритму реєстрації/авторизації

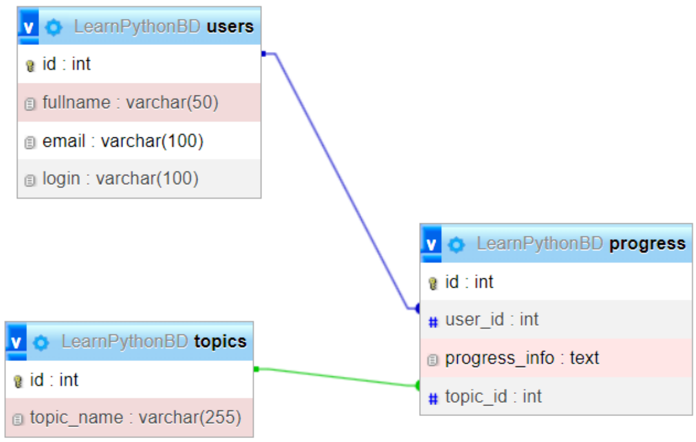

Рисунок 5 – ER-модель бази даних

Рисунок 6 – SQL-запит та вигляд сутності «user».

Рисунок 7 – «Дах» веб-сайту

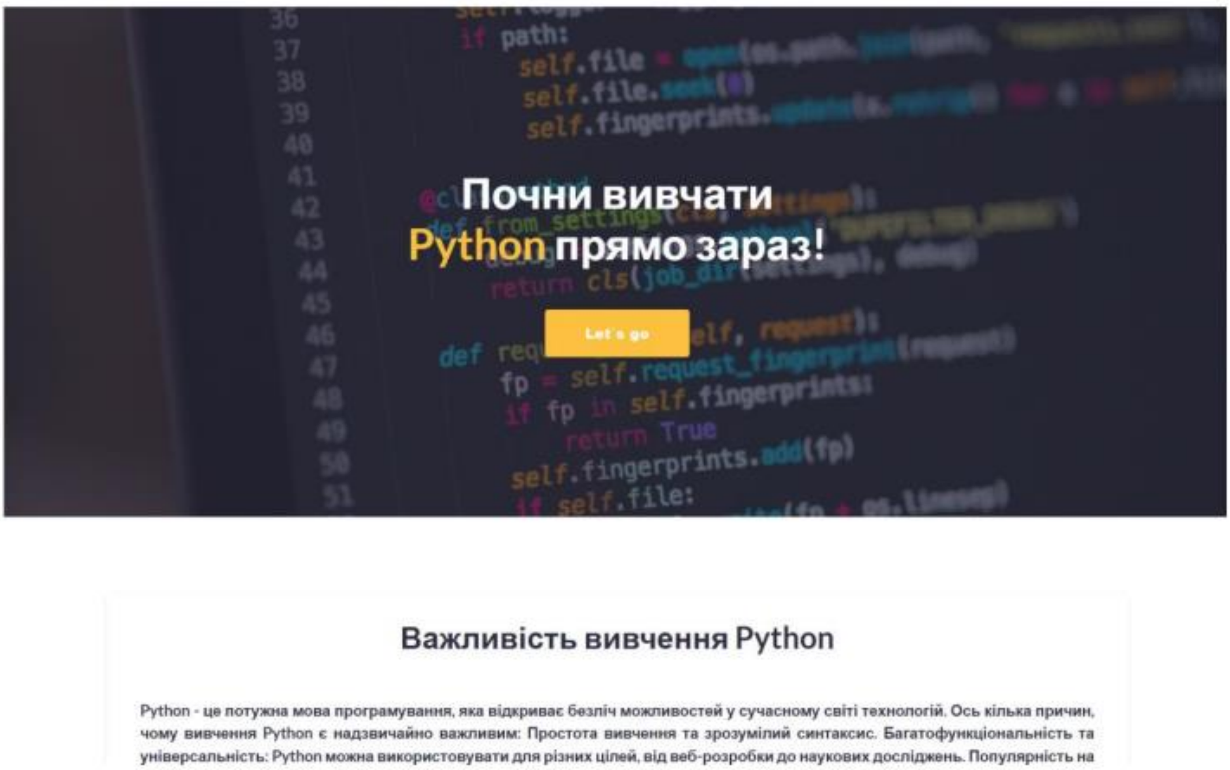

Рисунок 8 – Головна сторінка веб-сайту

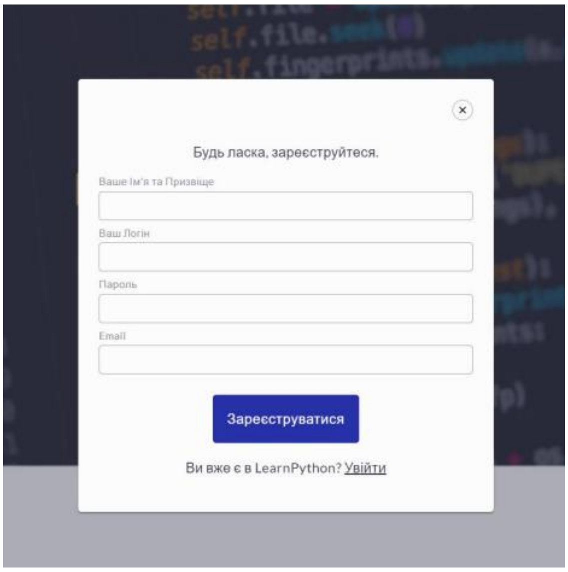

Рисунок 9 – Форма реєстрації користувача

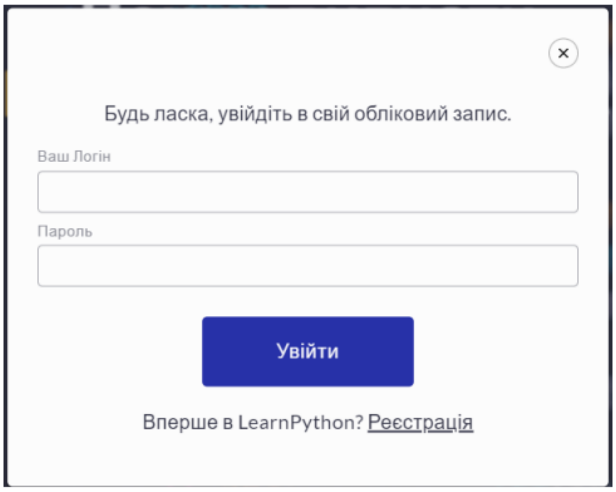

Рисунок 10 – Форма авторизації користувача

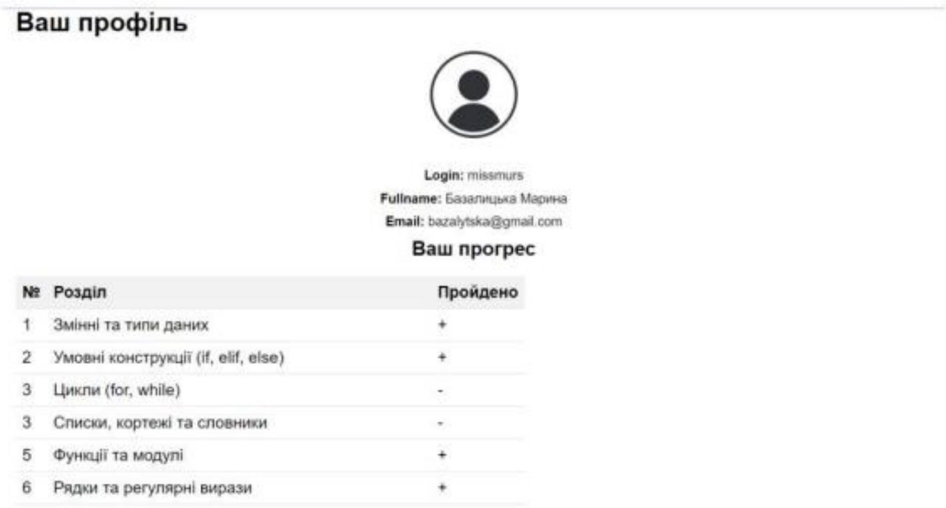

Рисунок 11 – Профіль користувача

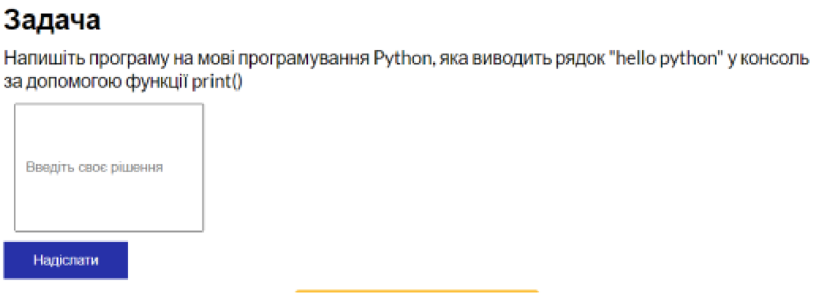

Рисунок 12 – Приклад задачі

Додаток Б
(обов`язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «веб-застосунок для вивчення мови програмування Python на основі протоколу http»

Тип роботи: Бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра інфокомунікаційних систем і технологій, факультет інформаційних електронних систем
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97,12 % Схожість 2,88 %

Аналіз звіту подібності (відмітити потрібне):

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

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. 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 ______ (підпис) Васильківський М.В. (прізвище, ініціали)

Ознайомлені з повним звітом, який був згенерований системою Unicheck щодо роботи.

Автор роботи ______ (підпис) Базалицька М. Р. (прізвище, ініціали)

Керівник роботи ______ (підпис) Воловик А. Ю. (прізвище, ініціали)

Додаток В – Лістинг програми

```
server.js
const mysql = require("mysql");
const express = require("express");
const ejs = require("ejs");

const app = express();
const topicsRouter = require("./topics");
app.use(topicsRouter);
const port = 7777;

app.set("view engine", "ejs");
app.use(express.static("public"));

app.get("/", (req, res) => {
  res.render("index");
});
const connection = mysql.createConnection({
  host: "127.0.0.1",
  user: "root",
  password: "",
  database: "LearnPython",
});

app.use(express.urlencoded({ extended: true }));
app.set("views", "./views");
app.set("view engine", "ejs");
app.use(express.static("public"));

connection.connect((err) => {
```

```
  if (err) {
    console.error("Error connecting to database:", err);
    return;
  }
  console.log("Connected to database successfully");
});

const register = require("./register");
app.use(register);
const login = require("./login");
app.use(login);
app.listen(port, () => {
  console.log(`Server started: http://localhost:${port}`);
});
```

login.js

```
const mysql = require("mysql");
module.exports = (req, res, next) => {
  if (req.method === "POST" && req.url === "/login") {
    const login = req.body.login;
    const pass = req.body.pass;

    const connection = mysql.createConnection({
      host: "127.0.0.1",
      user: "root",
      password: "",
      database: "LearnPython",
    });

    const query = "SELECT * FROM user WHERE login = ? AND pass = ?";
```

```
    connection.query(query, [login, pass], (err, results) => {
      if (err) {
        console.error("Помилка перевірки ідентичності:", err);
        res.status(500).send("Помилка перевірки ідентичності");
        return;
      }

      if (results.length > 0) {
        res.redirect("/profile");
      } else {
        res.send("Невірний логін або пароль");
      }
    });
  }
};
```

registration.js

```
const mysql = require("mysql");
module.exports = (req, res, next) => {
  if (req.method === "POST" && req.url === "/register") {
    const fullname = req.body.fullname;
    const login = req.body.login;
    const pass = req.body.pass;
    const email = req.body.email;

    const connection = mysql.createConnection({
      host: "127.0.0.1",
      user: "root",
      password: "",
      database: "LearnPython",
```

```
    });

    const query =
      "INSERT INTO user (fullname, login, pass, email) VALUES (?, ?, ?, ?)";

    connection.query(query, [fullname, login, pass, email], (err, result) =>
{
      if (err) {
        console.error(err);
        document.getElementById("message").innerText =
          "Під час реєстрації виникла помилка. Спробуйте ще раз.";
        return;
      }

      if (result.affectedRows === 1) {
        document.getElementById("message").innerText =
          "Ви успішно зареєструвалися!";
        res.redirect("/");
      } else {
        document.getElementById("message").innerText =
          "Користувач з таким логіном вже існує.";
      }
    });

    connection.end();
  } else {
    next();
  }
};
```

quest.js

```
document.getElementById("submitBtn").addEventListener("click", function ()
{
  var solution = document.getElementById("solution").value.trim();

  var correctSolution = "print('hello python')";

  if (solution === correctSolution) {
    document.getElementById("result").innerText = "Вірно!";
    alert("Вірно!");
    document.getElementById("result").style.color = "green";
  } else {
    document.getElementById("result").innerText = "Невірно!";
    alert("Невірно!");
    document.getElementById("result").style.color = "red";
  }
});
```

topics.js

```
const express = require("express");
const router = express.Router();
const mysql = require("mysql");
const ejs = require("ejs");
const connection = mysql.createConnection({
  host: "127.0.0.1",
  user: "root",
  password: "",
  database: "LearnPython",
});
// Маршрут для отримання списку тем
router.get("/manual", (req, res) => {
  connection.query("SELECT * FROM topics", (err, results) => {
```

```
    if (err) {
      console.error("Помилка запиту до бази даних:", err);
      res.status(500).send("Помилка сервера");
      return;
    }
    res.render("manual", { topics: results });
  });
});

router.get("/topics/:id", (req, res) => {
  const topicId = req.params.id;
  connection.query(
    "SELECT * FROM topics_info WHERE id = ?",
    [topicId],
    (err, results) => {
      if (err) {
        console.error("Помилка запиту до бази даних:", err);
        res.status(500).send("Помилка сервера");
        return;
      }
      if (results.length === 0) {
        res.status(404).send("Тема не знайдена");
        return;
      }
      res.render("block/topics_info", { topics_info: results[0] });
    }
  );
});

module.exports = router;
```

index.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LearnPython</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link

href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&display=swap"
        rel="stylesheet">
    <link rel="stylesheet" href="/main.css">
</head>



<body>
<header class="header">
    <div class="header-container container">
        <div>
            <a class="link logo" href="/">
                Learn <span class="python">Python</span>
            </a>
        </div>
        <div class="nav">
            <nav class="nav">
                <ul class="nav-list list">
                    <li><a class="link list-item " href="/manual">Посібник</a></li>
```

```
                    <li><a class="link list-item " href="/code">Редактор
кoду</a></li>
                </ul>
            </nav>
        </div>
        <div class="profile-btn-cont">
            <a class="link list-item " href="/profile">
                <button type="button" class="profile-btn">
                    <svg class="profile-btn-icon" width="20" height="20">
                        <use href="./images/icons.svg#icon-person"></use>
                    </svg>
                </button>
            </a>
        </div>
    </div>
</header>
<section class="hero-section section">
    <div class="hero-container container">
        <h1 class="hero-title">Почни вивчати
            <span class="python">
                Python
            </span> прямо зараз!
        </h1>
        <button type="button" class="hero-button" modal-singin-open-btn>
            Let`s go
        </button>
    </div>
</section>
<footer class="footer-section">
    <div class="container footer-container">
        <div class="logo-container">
```

```
            <a href="./index.html" class="link logo-footer">
                Learn<span class="studio-footer">Python</span>
            </a>
        </div>
        <div class="footer-form-box">
            <p class="form-title">Subscribe</p>
            <form class="footer-form">
                <label    for="email"    class="foter-form-label">    <input
type="email" class="footer-form-input"
                        name="user-email" id="email" placeholder="E-mail">
                </label>
                <button type="submit" class="btn-subscribe">Subscribe
                </button>
            </form>
        </div>
    </div>
</footer>

monual.ejs
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LearnPython</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
```

```
href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&display=swap"
        rel="stylesheet">
    <link rel="stylesheet" href="/main.css">
</head>


<body>
<section class="handbook-section">
    <div class="handbook-container">
        <div class="handbook-sidebar">
            <h2>Список розділів</h2>
            <ul>
                <% topics.forEach(function(item) { %>
                    <li>
                        <a class="link" href="/<%= item.id %>">
                            <p>
                                <%= item.id %>. <%= item.title %>
                            </p>
                        </a>
                    </li>
                    <% }); %>
            </ul>
        </div>
        <div class="handbook-content">
<ul>
        <% if (topicId) { %>
            <%- include(topicId) -%>
        <% } else { %>
            <li>No topic selected</li>
```

```
        <% } %>
    </ul>
            <button id="nextSectionButton" class="nextbtn">Наступний розділ</button>
        </div>


    </div>
</section>
</body>
</html>
```

styles.css

```
:root {
  --primary-brand: #2731a8;
  --pressed-state: #4552e3;
  --dark: #2e2f42;
  --success: #31d0aa;
  --text: #434455;
  --subtle-text: #8e8f99;
  --accent: #e7e9fc;
  --light: #f4f4fd;
  --modal-overlay: #2e2f4266;
  --hero-background: #2e2f42b2;
  --white: #ffffff;
  --modal-background: #fcfcfc;
  --python: #fcc13f;
}

* {
  padding: 0;
```

```
  margin: 0;
}

body {
  font-family: "Lato", sans-serif;
}

h1,
h2,
h3,
h4,
h5,
h6,
p {
  margin: 0;
  font-family: "Lato", sans-serif;
}
.link {
  text-decoration: none;
  color: var(--dark);
}
.list {
  list-style: none;
  padding: 0;
  margin: 0;
}
img {
  display: block;
  max-width: 100%;
  height: auto;
}
```

```
.link:hover,
.link:focus {
  color: var(--primary-brand);
}

.container {
  min-width: 1000px;
  max-width: 1158px;
  margin-left: auto;
  margin-right: auto;
  padding-left: 16px;
  padding-right: 16px;
}

.section-title {
  text-align: center;
  font-size: 36px;
  margin: 0 auto;
  line-height: 1.11;
  text-transform: capitalize;
  letter-spacing: 0.02em;
  color: var(--dark);
  margin-bottom: 72px;
}

.visually-hidden {
  position: absolute;
  width: 1px;
  height: 1px;
  margin: -1px;
```

```
  border: 0;
  padding: 0;
  white-space: nowrap;
  clip-path: inset(100%);
  clip: rect(0 0 0 0);
  overflow: hidden;
}
.header {
  border-bottom: 1px solid #e7e9fc;
  box-shadow: 0px 2px 1px rgba(46, 47, 66, 0.08),
    0px 1px 1px rgba(46, 47, 66, 0.16), 0px 1px 6px rgba(46, 47, 66, 0.08);
}

.header-container {
  display: flex;
  align-items: center;
  justify-content: center;
}
.logo {
  font-family: "Lato", sans-serif;
  font-size: 20px;
  font-weight: 800;
  display: flex;
  align-items: center;
  line-height: 1.17;
  letter-spacing: 0.03em;
  text-transform: uppercase;
  color: var(--primary-brand);
}

.python {
```

```
  color: var(--python);
}

.list-item {
  position: relative;
  display: block;
  padding-top: 24px;
  padding-bottom: 24px;
  font-weight: 600;
  font-size: 16px;
  line-height: 1.5;
  letter-spacing: 0.02em;
  transition: color 250ms cubic-bezier(0.4, 0, 0.2, 1);
  color: var(--dark);
}
.list-item::after {
  content: "";
  display: block;
  position: absolute;
  bottom: 0;
  left: 0;
  width: 100%;
  height: 4px;
  border-radius: 2px;
}

.nav {
  display: flex;
  align-items: center;
  margin-left: auto;
  margin-right: 40px;
```

```
}
.nav-list {
  display: flex;
  gap: 40px;
}

.profile-btn {
  top: 24px;
  right: 24px;
  width: 40px;
  height: 40px;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  border: 1px solid rgba(0, 0, 0, 0.3);
  padding: 0;
  cursor: pointer;
  background-color: var(--modal-background);
}

.profile-btn:hover,
.profile-btn:focus {
  background-color: var(--primary-brand);
  border: none;
  fill: var(--white);
}

.hero-container {
  display: flex;
  align-items: center;
```

```
  flex-direction: column;
}
.hero-title {
  color: var(--white);
  max-width: 496px;
  text-align: center;
  margin-bottom: 48px;
  font-size: 50px;
  font-style: normal;
  font-weight: 700;
  line-height: 60px; /* 107.143% */
}
.hero-button {
  min-width: 169px;
  height: 56px;
  background: var(--python);
  cursor: pointer;
  font-weight: 800;
  font-size: 16px;
  line-height: 1.5;
  letter-spacing: 0.04em;
  color: var(--white);
  display: block;
  margin: auto;
  padding: 16px 32px;
  height: 56px;
  border-radius: 4px;
  border: none;
  transition: background-color 250ms cubic-bezier(0.4, 0, 0.2, 1);
}
.hero-button:hover,
```

```
.hero-button:focus {
  background: var(--primary-brand);
}
.about-container {
  padding: 90px 0;
  margin: 0 auto;
  max-width: 1440px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-family: "Lato", sans-serif;
}
.about-title {
  text-align: center;
  padding: 30px;
  font-weight: 700;
}
.about-container-item {
  background: var(--white);
  color: var(--dark);
  margin: 0 120px;
  border-radius: 10px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  font-size: 18px;
}
.about-text {
  font-family: "Lato", sans-serif;
  font-weight: 600;
  text-align: justify;
  text-justify: inter-word;
  margin: 20px;
```

```
  line-height: 1.5;
  list-style: none;
}
.footer-section {
  background: var(--dark);
  padding-top: 100px;
  padding-bottom: 100px;
}
.footer-container {
  display: flex;
  justify-content: center;
  align-items: flex-start;
}
.logo-container {
  margin-right: 120px;
  background-color: #fff;
  padding: 5px;
}
.logo-footer {
  font-family: "Lato", sans-serif;
  font-size: 20px;
  font-weight: 800;
  display: flex;
  align-items: center;
  line-height: 1.17;
  letter-spacing: 0.03em;
  text-transform: uppercase;
  color: var(--primary-brand);
}

.studio-footer {
```

```
  color: var(--python);
}
.footer-text {
  color: var(--light);
  font-size: 16px;
  line-height: 24px;
  letter-spacing: 0.02em;
  width: 264px;
}

.form-title {
  margin-bottom: 16px;
  font-weight: 500;
  font-size: 16px;
  line-height: 1.5;
  letter-spacing: 0.02em;
  color: #ffffff;
}
.footer-form {
  display: flex;

  align-items: center;
  justify-content: center;
  gap: 24px;
}

.footer-form-input {
  width: 264px;
  height: 40px;
  border-radius: 4px;
  border: 1px solid var(--white);
```

```
  box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.15);
  outline: transparent;
  background-color: transparent;
  color: var(--white);
  border-radius: 4px;
  border: 1px solid var(--white, #fff);
  box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.15);
  font-size: 12px;
  line-height: 1.5;
  letter-spacing: 0.04em;
  padding-left: 16px;
}

.footer-form-input:hover,
.footer-form-input:focus {
  border-color: #4d5ae5;
}

.btn-subscribe {
  display: inline-flex;
  padding: 8px 24px;
  justify-content: center;
  align-items: center;
  gap: 16px;
  border-radius: 4px;
  background: var(--primary-brand);
  color: var(--white);
  text-align: center;
  border: none;
  font-family: "Roboto", sans-serif;
  font-size: 16px;
```

```
  font-style: normal;
  font-weight: 500;
  line-height: 24px; /* 150% */
  letter-spacing: 0.64px;
  transition: border-color 250ms cubic-bezier(0.4, 0, 0.2, 1),
    color 250ms cubic-bezier(0.4, 0, 0.2, 1);
  min-width: 165px;
  cursor: pointer;
}
```