МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет
імені Дмитра Моторного
Факультет енергетики і комп'ютерних технологій

ЗАТВЕРДЖУЮ
Зав. каф. "Комп'ютерні науки"
доц._____ _____Сергій ШАРОВ
«16»_червня__2025 р.

Пояснювальна записка

до кваліфікаційної роботи здобувача СВО Бакалавр
(ступінь вищої освіти)

на тему: «Інформаційна онлайн-система для самостійного вивчення основам програмування»

52/4КНД.11323044.000000ПЗ

Виконав: здобувач вищої освіти 4 курсу, групи 21С(МС)КН
спеціальності 122 Комп'ютерні науки
за ОПП Комп'ютерні науки
(шифр і назва спеціальності та ОПП)

_____ Максим ЧЕРЕП
(підпис)

Керівник доц. _____ Ольга ЗІНОВ'ЄВА
(підпис)

Консультант доц. _____ Лариса БОЛТЯНСЬКА
(підпис)

Консультант доц. _____ Михайло ЗОРЯ_____
(підпис)

Нормоконтроль, ст.викл. _____ Ольга ЗІНОВ'ЄВА
(підпис)

Рецензент доц _____ Олексій НАУМУК
(підпис)

Запоріжжя – 2025 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Факультет: Енергетики і комп'ютерних технологій
Кафедра: Комп'ютерні науки
Ступінь вищої освіти: Бакалавр
Спеціальність: 122 Комп'ютерні науки
ОПП: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
к.пед.н., доц. _____Сергій ШАРОВ
"10" _жовтня__2024__ року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

__Черепу Максиму Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи__«Інформаційна онлайн-система для самостійного вивчення основам програмування»
керівник проєкту __Зінов'єва Ольга Геннадіївна, ст. викладач_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом університету від "30" вересня_2024_року №_452-С
2. Строк подання здобувачем вищої освіти роботи __14 червня 2025_____
3. Вихідні дані до роботи: дані обстеження об'єкту, статистичні дані, технічне завдання на дипломне проектування, матеріали виробничих практик, нормативні документи, науково-технічна література, електронні ресурси та ін.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)____
___1. Аналіз предметної області та існуючих рішень для вивчення програмування
___2. Проектування архітектури інформаційної онлайн-системи для самостійного вивчення основ програмування
___3. Вибір та обґрунтування технологій розробки веб-системи
___4. Реалізація функціональних модулів онлайн-системи
___5. Тестування та оцінка ефективності розробленої системи
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)
Діаграма варіантів використання, діаграми IDEF0, діаграма ER, прототип інтерфейсу, діаграма компонентів системи, інтерфейс програми.
Презентація – 11 слайдів

1. Титульний слайд

2. Предмет, об’єкт, мета
3. Завдання дослідження
4. Порівняльна характеристика аналогів програмного продукту
5. Діаграма варіантів використання
6. Інструментальні засоби розробки
7. Інтерфейс і робота онлайн системи
8. Інтерфейс і робота онлайн системи
9. Тестування
10. Висновок
11. Дякую за увагу

6. Консультанти розділів кваліфікаційної роботи

Розділ/ Підрозділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4.3	Болтянська Л. О.	15.10.2025	15.06.2025
4.4	Зоря М. В.	15.10.2025	15.06.2025

7. Дата видачі завдання___10 жовтня 2024 р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	03.10.2024	Виконано
2	Специфікація вимог до локальної інформаційної системи	17.10.2024	Виконано
3	Проектування локальної інформаційної системи	28.10.2024	Виконано
4	Обґрунтування інструментальних засобів	18.11.2024	Виконано
5	Розробка локальної інформаційної системи	16.12.2024	Виконано
6	Тестування та налагодження локальної інформаційної системи	11.04.2025	Виконано
7	Підпис керівником роботи	14.06.2025	Виконано
8	Підпис завідувачем кафедри	16.06.2025	Виконано

Здобувач вищої освіти _____Максим ЧЕРЕП__
(підпис) (власне ім’я та прізвище)

Керівник кваліфікаційної роботи _____Ольга ЗІНОВ’ЄВА_
(підпис) (власне ім’я та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 83 с., 10 рис., 13 табл., 26 посилань, 2 додатки.

Актуальність. В умовах стрімкого розвитку інформаційних технологій та зростаючої потреби у кваліфікованих програмістах, розробка ефективних онлайн-систем для самостійного вивчення основ програмування є надзвичайно актуальною. Розроблена інформаційна онлайн-система забезпечує структурований доступ до навчальних матеріалів, інтерактивні практичні завдання та персоналізований підхід до вивчення різних мов програмування українською мовою.

Об'єкт дослідження: процес самостійного вивчення основ програмування за допомогою онлайн-систем навчання.

Предмет дослідження: Інформаційна онлайн-система для самостійного вивчення основам програмування.

Мета роботи: розробка інформаційної онлайн-системи для самостійного вивчення основ програмування, що забезпечує персоналізований підхід до навчання та високу ефективність засвоєння матеріалу.

Завдання дослідження:

- аналіз існуючих онлайн-платформ для вивчення програмування та виявлення їх переваг і недоліків;

- визначення оптимальних методик та підходів до самостійного вивчення програмування;

- розробка архітектури інформаційної онлайн-системи для вивчення основ програмування;

- проектування та реалізація функціональних модулів системи;

- тестування системи та оцінка її ефективності.

Практичне значення полягає у створенні доступної україномовної онлайн-

системи, яка дозволяє користувачам ефективно опановувати основи програмування на різних мовах у зручному темпі, що відповідає індивідуальним потребам, сприяючи розвитку вітчизняної ІТ-галузі.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, ОНЛАЙН-НАВЧАННЯ, ПРОГРАМУВАННЯ, ІНТЕРАКТИВНЕ СЕРЕДОВИЩЕ, САМООСВІТА, МОВИ ПРОГРАМУВАННЯ.

SUMMARY

Bachelor's Qualification Work: 83 pages, 10 figures, 13 tables, 26 references, 2 appendices.

Relevance: In the context of rapid development in information technologies and the growing demand for skilled programmers, the development of effective online systems for independent learning of programming fundamentals is highly relevant. The developed informational online system provides structured access to educational materials, interactive practical tasks, and a personalized approach to learning various programming languages in Ukrainian.

Object of study: the process of independent learning of programming fundamentals using online learning systems.

Subject of study: methods, models, and information technologies for creating an effective online system for self-directed learning of programming basics.

Purpose of the work: to develop an informational online system for independent learning of programming fundamentals that ensures a personalized approach to education and high efficiency in mastering the material.

Research tasks:

- Analyze existing online platforms for learning programming and identify their strengths and weaknesses;

- Determine optimal methods and approaches for self-directed programming education;

- Develop the architecture of an informational online system for learning programming fundamentals;

- Design and implement functional modules of the system;

- Test the system and evaluate its effectiveness.

Practical significance: lies in the creation of an accessible Ukrainian-language online system that allows users to effectively master programming fundamentals in different languages at a convenient pace that matches individual needs, thereby contributing to the development of the national IT sector.

Keywords: INFORMATION SYSTEM, ONLINE LEARNING, PROGRAMMING, INTERACTIVE ENVIRONMENT, SELF-EDUCATION, PROGRAMMING LANGUAGES.

ВСТУП

Актуальність теми дослідження. В умовах стрімкого розвитку інформаційних технологій та зростаючої потреби у кваліфікованих програмістах, розробка ефективних онлайн-систем для самостійного вивчення основ програмування є надзвичайно актуальною. Сучасний ринок праці потребує фахівців з програмування, а традиційні методи навчання не завжди відповідають темпам розвитку галузі та індивідуальним потребам студентів.

Об'єктом дослідження є процес самостійного вивчення основ програмування за допомогою онлайн-систем навчання.

Предмет дослідження. Предметом дослідження є методи, моделі та інформаційні технології для створення ефективної онлайн-системи самостійного вивчення основ програмування.

Метою дослідження є розробка інформаційної онлайн-системи для самостійного вивчення основ програмування, що забезпечує персоналізований підхід до навчання та високу ефективність засвоєння матеріалу.

Завдання дослідження:

1. Аналіз існуючих онлайн-платформ для вивчення програмування та виявлення їх переваг і недоліків;
2. Визначення оптимальних методик та підходів до самостійного вивчення програмування;
3. Розробка архітектури інформаційної онлайн-системи для вивчення основ програмування;
4. Проектування та реалізація функціональних модулів системи;
5. Тестування системи та оцінка її ефективності.

Методи дослідження.

У роботі використано теоретичні методи (аналіз науково-методичної літератури, порівняння існуючих платформ, моделювання освітніх процесів) та

практичні методи (проектування та розробка програмного забезпечення, тестування функціональності системи, експериментальна перевірка ефективності).

Практична значущість. Розроблена інформаційна онлайн-система дозволить користувачам ефективно опановувати основи програмування в зручному темпі та форматі, що відповідає їхнім індивідуальним потребам. Система може бути впроваджена в освітніх закладах як додатковий інструмент навчання або використовуватись для самоосвіти.

Структура роботи. Робота складається з чотирьох розділів:

1. Аналіз предметної області та існуючих рішень для вивчення програмування — розглядаються сучасні підходи до навчання програмуванню та аналізуються наявні онлайн-платформи;
2. Проектування інформаційної системи — визначаються вимоги до системи та розробляється її архітектура;
3. Вибір та обґрунтування технологій розробки — обираються оптимальні інструменти та технології для реалізації системи;
4. Дослідна експлуатація та тестування — проводиться апробація системи та оцінюється її ефективність.

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика об'єкта управління

Об'єктом управління даного дипломного проєкту є процес самостійного навчання основам програмування через спеціалізовану інформаційну веб-систему. Проєкт спрямований на створення інноваційного освітнього середовища, що поєднує структуровані навчальні матеріали, інтерактивні практичні завдання та інструменти для відстеження прогресу навчання.

Як основні аналоги для порівняння було розглянуто такі популярні платформи для навчання програмуванню, як «Codecademy» та «freeCodeCamp», які дозволяють користувачам опановувати мови програмування та розвивати навички розробки. Проте, на відміну від існуючих рішень, розроблювана веб-система має низку особливостей, що відрізняють її від конкурентів:

- адаптивне форматування навчальних матеріалів відповідно до рівнів складності (початковий, середній, просунутий);
- генерація інтерактивних завдань для широкого спектру мов програмування: Python, C#, C++, PHP, JavaScript, Ruby та Delphi;
- кросплатформний доступ до функціоналу через різноманітні пристрої (комп'ютери, планшети, смартфони);
- інтегрована система збереження та відстеження прогресу навчання;
- україномовний інтерфейс та навчальні матеріали, адаптовані для вітчизняних користувачів;
- система гейміфікації для підвищення мотивації користувачів.

Цільовою аудиторією сервісу є студенти навчальних закладів, ентузіасти самоосвіти, початківці у сфері IT та розробники, які бажають розширити свої знання, опановуючи нові мови програмування. Система спрямована на забезпечення ефективного навчання для користувачів з різним

рівнем базової підготовки.

Структура розроблюваного сервісу включає наступні основні модулі:

1. Модуль навчальних матеріалів - відповідає за представлення теоретичної інформації про мови програмування, включаючи синтаксис, основні концепції, приклади коду. Модуль забезпечує структуроване подання матеріалу з поступовим нарощуванням складності.

2. Модуль практичних завдань - ключовий компонент, який генерує інтерактивні вправи для закріплення отриманих знань. Завдання поділяються за рівнями складності та тематикою, забезпечуючи поступовий розвиток навичок програмування.

3. Модуль оцінювання - забезпечує автоматичну перевірку правильності виконання завдань та надає детальний зворотний зв'язок. Модуль використовує алгоритми аналізу коду для виявлення помилок та надання рекомендацій щодо оптимізації рішень.

4. Модуль відстеження прогресу - дозволяє користувачам бачити свій прогрес у вивченні різних мов програмування, формує візуалізовані звіти та рекомендації щодо подальшого навчання.

5. Модуль спільноти - забезпечує комунікацію між користувачами платформи для обміну досвідом, взаємодопомоги та спільного вирішення складних задач.

Для забезпечення ефективної роботи сервісу критично важливими факторами є:

- актуальність та структурованість навчального контенту для різних мов програмування;
- швидкодія інтерактивного середовища для написання та тестування коду;
- точність системи автоматичної перевірки рішень;
- зручність та інтуїтивність користувацького інтерфейсу;
- захищеність системи від зловмисного коду.

Процес навчання програмуванню через розроблювану систему є комплексним і включає наступні етапи:

1. Вивчення теоретичних матеріалів - користувач ознайомлюється з основними поняттями, синтаксисом та парадигмами обраної мови програмування.

2. Виконання практичних завдань - застосування отриманих знань для вирішення конкретних задач різного рівня складності. Система підтримує поступове нарощування складності для формування стійких навичок.

3. Отримання зворотного зв'язку - система оцінює правильність рішення, аналізує якість коду та надає рекомендації щодо покращення алгоритмів та структури програми.

4. Проходження тестів - комплексна перевірка рівня засвоєння матеріалу з різних тем та розділів.

5. Реалізація проєктів - застосування комплексу навичок для створення функціональних програм, що вирішують практичні задачі.

Ключову роль у процесі навчання програмуванню відіграє інтерактивність та можливість одразу бачити результати своєї роботи. Система забезпечує вбудоване середовище для написання та тестування коду, що дозволяє користувачам практикуватися без необхідності встановлення додаткового програмного забезпечення. Такий підхід особливо важливий для початківців, які часто стикаються з труднощами при налаштуванні середовища розробки.

До впровадження спеціалізованої веб-системи навчання програмуванню в Україні здебільшого здійснювалося через книги, відеокурси або загальні освітні платформи, що мали обмежену функціональність та недостатню інтерактивність. Більшість якісних ресурсів доступні лише англійською мовою, що створює додатковий бар'єр для багатьох потенційних користувачів. Саме потреба у більш структурованому, інтерактивному та спеціалізованому інструменті з підтримкою української мови стала підґрунтям для розробки власної веб-системи.

Запропонований продукт має на меті забезпечити:

- структуроване подання навчальних матеріалів - послідовне вивчення концепцій від простих до складних з акцентом на практичному застосуванні знань;

- інтерактивне середовище для практики - можливість писати та тестувати код безпосередньо у браузері з підтримкою різних мов програмування;

- миттєвий зворотний зв'язок - автоматична перевірка коду та надання рекомендацій для формування правильних підходів до вирішення задач;

- адаптивний підхід до навчання - система підлаштовується під темп та стиль навчання користувача, пропонуючи оптимальний шлях опанування матеріалу;

- підтримку різних мов програмування - можливість вивчати Python, C#, C++, PHP, JavaScript, Ruby та Delphi на одній платформі з відстеженням прогресу по кожній з них;

- елементи гейміфікації - система досягнень, рейтингів та винагород для підтримки мотивації користувачів;

- формування спільноти - можливість взаємодії між користувачами, обміну досвідом та взаємодопомоги.

Програмний продукт базується на сучасних веб-технологіях, що забезпечують його крос-платформність, адаптивність та швидкодію. Для фронтенду використовуються HTML5, CSS3 та JavaScript (React.js), для бекенду - Node.js та MongoDB, що дозволяє створити гнучку та масштабовану архітектуру. Система розгортається на хмарній платформі з використанням Docker-контейнерів для ізоляції середовища виконання коду.

Особлива увага приділяється безпеці виконання користувацького коду, зокрема через використання пісочниці (sandbox) та обмеження ресурсів для запобігання зловмисним діям. Усі навчальні матеріали проходять перевірку на актуальність та відповідність сучасним стандартам індустрії.

Очікується, що впровадження цієї веб-системи значно підвищить доступність навчання програмуванню для українських користувачів, забезпечить більш ефективне засвоєння матеріалу та збільшить кількість кваліфікованих спеціалістів у галузі ІТ. Рішення має на меті спростити складний процес вивчення програмування, зробивши його доступним для широкого кола користувачів незалежно від їх місцезнаходження та рівня базової підготовки.

Завдяки реалізації розроблюваного програмного продукту ентузіасти та початківці отримають ефективний інструмент для самостійного навчання основам програмування та розвитку відповідних навичок, що є особливо актуальним в умовах зростаючого попиту на ІТ-спеціалістів та тенденції до самоосвіти.

Предметна область проєкту охоплює процеси самостійного навчання основам програмування та опанування різних мов програмування через спеціалізовану веб-систему. Ефективне вивчення програмування вимагає чіткої структуризації матеріалу, практичного закріплення теоретичних знань та постійного зворотного зв'язку, що стає можливим завдяки впровадженню сучасних інформаційних технологій у навчальний процес.

Самостійне навчання програмуванню характеризується низкою особливостей, що визначають вимоги до розроблюваної системи:

- необхідність поступового переходу від базових концепцій до більш складних, з урахуванням особливостей кожної мови програмування;
- важливість практичного застосування теоретичних знань через інтерактивні вправи;
- потреба у миттєвому зворотному зв'язку для коригування помилок та закріплення правильних підходів;
- різноманітність стилів навчання та темпів засвоєння матеріалу серед користувачів;
- значна роль мотивації та самодисципліни у процесі самостійного

навчання.

Користувачами сервісу є студенти вищих навчальних закладів, самоучки, ентузіасти та початківці, які прагнуть опанувати програмування власними силами. Дослідження показують, що 67% людей, які починають вивчати програмування самостійно, зіштовхуються з труднощами через відсутність структурованого підходу та фрагментованість навчальних ресурсів. Особливістю роботи з такими користувачами є необхідність забезпечення мотивації, чіткого структурування матеріалу та надання детального зворотного зв'язку.

Ключовими аспектами управління навчальним процесом у системі є:

1. Створення інтерактивного середовища навчання з можливістю написання та тестування коду безпосередньо у браузері.

2. Забезпечення адаптивного підходу до користувачів з різним рівнем підготовки через динамічне налаштування складності матеріалів та завдань.

3. Автоматизація перевірки виконаних завдань з наданням детальних пояснень та рекомендацій.

4. Відстеження прогресу користувача та формування індивідуальної траєкторії навчання.

5. Інтеграція елементів гейміфікації для підтримки зацікавленості та мотивації.

Основними об'єктами в межах предметної області є:

1. Мови програмування, що підтримуються системою:

- Python – мова загального призначення з простим синтаксисом, ідеальна для початківців;

- C# – об'єктно-орієнтована мова, що використовується для розробки додатків на платформі .NET;

- C++ – потужна мова для системного програмування та розробки продуктивних додатків;

- PHP – мова для веб-розробки, що використовується для створення

динамічних веб-сайтів;

- JavaScript – мова програмування для веб-розробки, що виконується на стороні клієнта;

- Ruby – об'єктно-орієнтована мова з простим та читабельним синтаксисом;

- Delphi – об'єктно-орієнтована мова, що використовується для розробки програмного забезпечення для Windows.

2. Навчальні модулі – структуровані блоки інформації, що охоплюють певну тему або концепцію:

- теоретичні матеріали з текстовими поясненнями та прикладами коду;

- відеоматеріали для візуального сприйняття складних концепцій;

- інтерактивні демонстрації для ілюстрації принципів роботи алгоритмів та конструкцій;

- довідкові матеріали з синтаксису та бібліотек обраної мови програмування.

3. Практичні завдання – вправи для закріплення теоретичних знань:

- базові завдання для перевірки розуміння окремих концепцій;

- завдання середнього рівня складності, що вимагають комбінування кількох концепцій;

- складні завдання, що стимулюють аналітичне мислення та творчий підхід;

- завдання на виправлення помилок у наданому коді (debugging).

4. Проєкти – комплексні завдання, що передбачають застосування кількох концепцій для створення функціонального програмного рішення:

- мініпроєкти для закріплення окремих тем;

- середні проєкти, що охоплюють кілька тем;

- фінальні проєкти, що демонструють опанування мови програмування.

5. Користувацькі профілі – сховище інформації про прогрес навчання,

досягнення та налаштування:

- особисті дані користувача;
- історія виконаних завдань та проєктів;
- статистика навчання та рівень опанування різних мов програмування;
- отримані досягнення та нагороди;
- персоналізовані рекомендації щодо подальшого навчання.

Усі описані об'єкти взаємодіють у межах єдиної інформаційної системи, побудованої на основі сучасної архітектури веб-додатків. Це забезпечує швидкий доступ до функціоналу, незалежно від пристрою та місця знаходження користувача. Система використовує реактивний підхід до оновлення даних, що дозволяє миттєво відображати зміни та забезпечувати високий рівень інтерактивності.

Процеси предметної області можна поділити на три ключові групи:

1. Засвоєння теоретичного матеріалу:
   - презентація структурованої інформації з поступовим нарощуванням складності;
   - використання різних форматів подання матеріалу (текст, відео, інтерактивні демонстрації);
   - перевірка розуміння через міні-тести та питання.
2. Виконання практичних завдань:
   - надання інтерактивного середовища для написання коду;
   - автоматична перевірка рішень з використанням тестових сценаріїв;
   - аналіз якості коду та надання рекомендацій щодо оптимізації;
   - підтримка різних підходів до вирішення завдань.
3. Моніторинг прогресу навчання:
   - відстеження виконаних завдань та опанованих тем;
   - аналіз рівня засвоєння матеріалу та виявлення проблемних областей;
   - формування індивідуальних рекомендацій щодо подальшого навчання;

- візуалізація прогресу для підтримки мотивації.

Процес навчання програмуванню через розроблювану систему починається з реєстрації користувача та вибору мови програмування для вивчення. Користувач опрацьовує навчальні матеріали перших розділів, виконує практичні завдання для закріплення знань та отримує зворотний зв'язок від системи. На основі результатів виконання завдань система аналізує рівень засвоєння матеріалу та пропонує оптимальний шлях для подальшого навчання. Цей цикл повторюється для кожного розділу обраної мови програмування.

Після завершення вивчення основного курсу з певної мови програмування користувач має можливість перейти до реалізації комплексних проєктів, що дозволяють застосувати всі отримані знання для створення функціональних програм. Система також пропонує опанувати додаткові мови програмування, використовуючи вже набуті знання та досвід для прискорення процесу навчання.

Інформаційна система забезпечує взаємодію з різними категоріями користувачів через адаптивний веб-інтерфейс, що автоматично налаштовується під розмір екрану пристрою (комп'ютер, планшет, смартфон). Це дозволяє навчатися в різних умовах та з різних пристроїв, що особливо важливо для самостійного навчання, яке часто відбувається у вільний час та без прив'язки до конкретного місця.

Основні проблеми, які існували до впровадження нової веб-системи:

1. Фрагментованість навчальних ресурсів - інформація розпорошена між різними джерелами, що ускладнює формування цілісної картини та послідовного вивчення матеріалу.

2. Відсутність структурованого підходу до вивчення програмування - багато існуючих ресурсів не забезпечують логічної послідовності вивчення концепцій від простих до складних.

3. Обмежені можливості для практики без встановлення спеціального програмного забезпечення - необхідність налаштування середовища розробки

часто стає першим бар'єром для початківців.

4. Недостатній зворотний зв'язок при самостійному навчанні - відсутність автоматичної перевірки та рекомендацій щодо покращення коду.

5. Складність одночасного вивчення кількох мов програмування - різні платформи використовують різні підходи та не забезпечують єдиного відстеження прогресу.

6. Мовний бар'єр - більшість якісних ресурсів доступні лише англійською мовою, що створює додаткові труднощі для українських користувачів.

7. Висока вартість доступу до преміум-контенту - багато якісних платформ пропонують обмежений безкоштовний доступ, а повний функціонал доступний лише на платній основі.

Розроблювана веб-система спрямована на вирішення зазначених проблем через створення інтегрованого, структурованого та інтерактивного середовища для навчання програмуванню з підтримкою української мови та широкого спектру мов програмування.

1.2 Огляд і аналіз існуючих аналогів системи.

1.2.1 Огляд систем для навчання програмуванню

Для навчання основам програмування та опанування різних мов програмування використовуються різні типи освітніх платформ, такі як інтерактивні курси, відеоуроки та онлайн-тренажери. Світовий ринок EdTech демонструє стабільне зростання, а сегмент онлайн-навчання програмуванню є одним з найбільш динамічних, показуючи щорічний приріст у 17-20%.

Існуючі системи для навчання програмуванню можна класифікувати за кількома основними категоріями:

1. Інтерактивні платформи з вбудованим IDE:

- надають можливість писати та виконувати код безпосередньо у

браузері;

- забезпечують миттєвий зворотний зв'язок та автоматичну перевірку рішень;

- пропонують структуровані курси з поступовим нарощуванням складності.

2. Відеокурси та текстові матеріали:

- пропонують детальні пояснення концепцій та технік програмування;

- часто не мають інтерактивного компонента для практики;

- передбачають самостійне встановлення середовища розробки.

3. Спеціалізовані IDE з навчальними функціями:

- поєднують повноцінне середовище розробки з навчальними матеріалами;

- потребують встановлення на комп'ютер користувача;

- часто орієнтовані на конкретну мову програмування.

4. Мобільні додатки для навчання програмуванню:

- забезпечують доступ до навчальних матеріалів з мобільних пристроїв;

- пропонують спрощені завдання та інтерфейс для написання коду;

- мають обмежені можливості порівняно з веб-платформами.

5. Системи з геймифікацією:

- використовують ігрові механіки для підвищення мотивації користувачів;

- пропонують систему досягнень, рейтингів та винагород;

- роблять процес навчання більш захоплюючим та інтерактивним.

Найбільш ефективними вважаються інтерактивні платформи з вбудованим IDE, що дозволяють користувачам практикуватися безпосередньо під час вивчення матеріалу. Вони забезпечують повний цикл навчання: від засвоєння теоретичних концепцій до практичного застосування знань та отримання зворотного зв'язку.

У цьому дослідженні особлива увага приділяється системам Codecademy та freeCodeCamp як найбільш поширеним та функціональним рішенням у категорії інтерактивних платформ з вбудованим IDE. На ринку присутні й інші продукти, які не будуть розглянуті детально, оскільки вони мають обмежену функціональність для українських користувачів або не забезпечують достатньої підтримки вітчизняних студентів.

1.2.2 Огляд основних аналогів

Codecademy – це популярна онлайн-платформа для вивчення програмування, що була заснована у 2011 році та має понад 50 мільйонів зареєстрованих користувачів зі всього світу. Платформа пропонує інтерактивні курси з різних мов програмування та технологій, включаючи Python, JavaScript, HTML/CSS, SQL, Java, Ruby та інші. Система дозволяє користувачам вивчати програмування через вирішення практичних завдань безпосередньо у браузері, без необхідності встановлення додаткового програмного забезпечення (рис. 1.1).

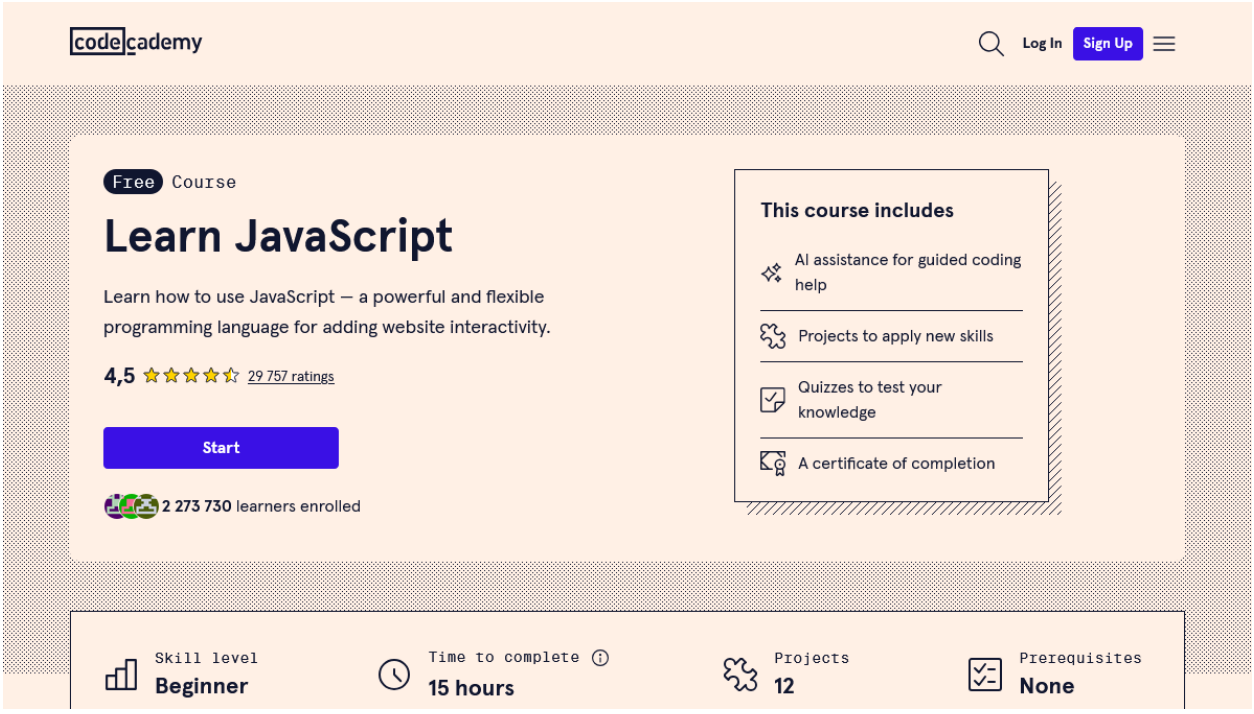

Рисунок 1.1 - Користувацький інтерфейс Codecademy

Основними компонентами платформи Codecademy є:

1. Навчальні курси – структуровані блоки матеріалів, розділені за мовами програмування та рівнями складності.

2. Інтерактивний редактор коду – середовище для написання та виконання коду з підсвіткою синтаксису та миттєвою перевіркою.

3. Система проєктів – набір завдань для практичного застосування отриманих знань.

4. Форуми та спільнота – площадка для обміну досвідом та отримання допомоги від інших користувачів.

5. Система досягнень – механізм гейміфікації для підтримки мотивації.

Переваги:

- інтерактивне середовище для написання та тестування коду, що дозволяє практикуватися без встановлення додаткового ПЗ;

- структуровані курси з поступовим нарощуванням складності, що забезпечує логічний шлях навчання;

- миттєвий зворотний зв'язок щодо написаного коду з детальними поясненнями помилок;

- різноманітність курсів та тем, що охоплюють широкий спектр мов програмування та технологій;

- якісний дизайн інтерфейсу та продумана навігація.

Недоліки:

- обмежена кількість безкоштовного контенту, більшість курсів доступні лише за платною підпискою;

- недостатня підтримка української мови, що створює додатковий бар'єр для вітчизняних користувачів;

- відсутність підтримки деяких мов програмування, таких як Delphi, які популярні в українській освітній системі;

- обмежені можливості для адаптації навчального процесу до

індивідуальних потреб користувача;

- недостатня інтеграція з іншими інструментами розробки.

Для українських користувачів, які бажають вивчати програмування самостійно, Codecademy може бути корисною платформою, але має обмеження через мовний бар'єр та відсутність адаптації до специфіки вітчизняної освіти. Крім того, висока вартість преміум-підписки (приблизно $20-40 на місяць) робить повний функціонал платформи недоступним для багатьох потенційних користувачів в Україні.

freeCodeCamp – це безкоштовна платформа з відкритим кодом, заснована у 2014 році як некомерційна організація, що спрямована на навчання веб-розробці. Платформа має понад 40 мільйонів користувачів та пропонує сертифікаційні програми, які охоплюють HTML, CSS, JavaScript, React, Node.js, Python та інші технології (рис. 1.2) [2].

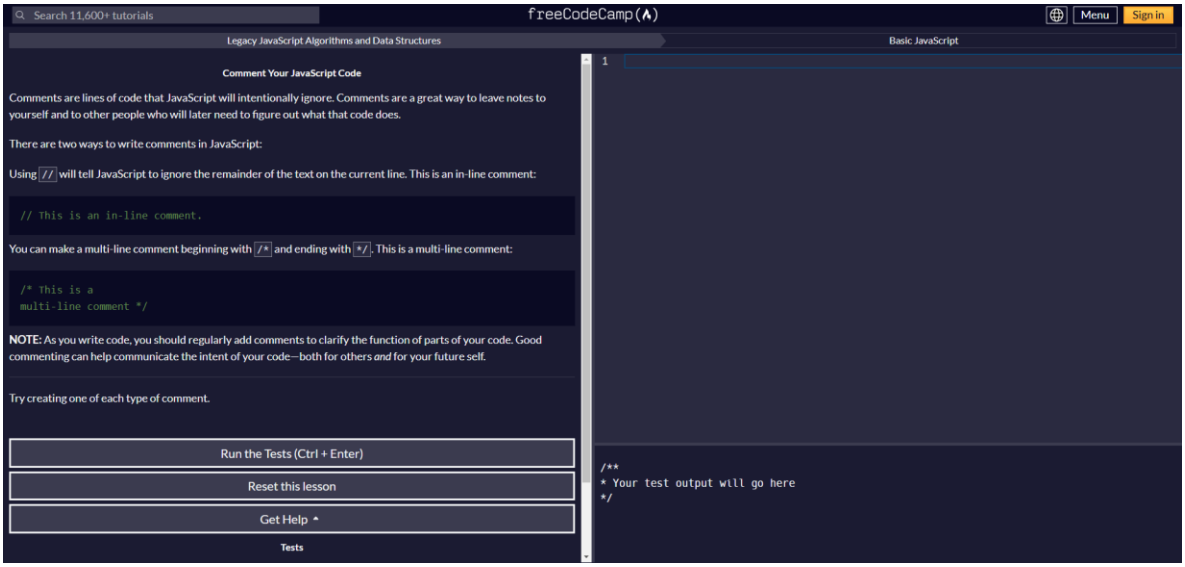

Рисунок 1.2 - Користувацький інтерфейс freeCodeCamp

Основними компонентами платформи freeCodeCamp є:

1. Сертифікаційні програми – структуровані курси з певної технології або напрямку розробки.

2. Інтерактивні завдання – вправи для практики програмування

безпосередньо у браузері.

3. Проєкти – комплексні завдання для застосування отриманих навичок.

4. Навчальні статті та відео – додаткові матеріали для поглибленого вивчення тем.

5. Форуми та спільнота – площадка для взаємодопомоги та обміну досвідом.

Переваги:

1. Повністю безкоштовний доступ до всіх навчальних матеріалів, що робить платформу доступною для широкого кола користувачів;

2. Проєктний підхід до навчання з реальними завданнями, що сприяє формуванню практичних навичок;

3. Активна спільнота, яка підтримує новачків та забезпечує додаткові ресурси для навчання;

4. Можливість отримання сертифікатів після завершення курсів, що може бути корисним для працевлаштування;

5. Відкритий код платформи, що дозволяє спільноті вдосконалювати та розширювати її функціонал.

Недоліки:

1. Фокус переважно на веб-розробці, що обмежує можливості для вивчення інших напрямків програмування;

2. Обмежена підтримка деяких мов програмування (C++, C#, Delphi), які є популярними в Україні;

3. Недостатня структурованість для повних новачків, що може ускладнювати початковий етап навчання;

4. Відсутність офіційної підтримки української мови, хоча спільнота працює над перекладами;

5. Менш інтуїтивний інтерфейс порівняно з деякими комерційними платформами.

Для українських користувачів freeCodeCamp може бути корисним для вивчення веб-розробки, але має обмеження щодо вивчення інших мов програмування та парадигм. Безкоштовний доступ до всіх матеріалів є значною перевагою, але відсутність офіційної підтримки української мови створює додаткові труднощі для користувачів з обмеженим знанням англійської[3].

1.2.3 Аналіз недоліків існуючих аналогів та обґрунтування вибору власної системи

Проведений аналіз існуючих систем для навчання програмуванню дозволяє виявити ряд спільних недоліків, які обмежують їх ефективність для українських користувачів, особливо початківців. Ці недоліки можна систематизувати за кількома категоріями:

1. Мовні та культурні бар'єри:

- недостатня підтримка української мови в інтерфейсі та навчальних матеріалах;
- відсутність адаптації контенту до особливостей вітчизняної освітньої системи;
- культурні відмінності у прикладах та кейсах, що використовуються для ілюстрації концепцій.

2. Технічні обмеження:

- обмежена підтримка деяких мов програмування, популярних в Україні (Delphi, C++);
- недостатня оптимізація для користувачів з повільним інтернет-з'єднанням;
- проблеми з доступністю через геополітичні обмеження для деяких сервісів.

3. Фінансові бар'єри:

- висока вартість повного доступу до матеріалів на комерційних

платформах;

- невідповідність цінової політики купівельній спроможності українських користувачів;

- обмежений функціонал безкоштовних версій.

4. Методичні недоліки:

- недостатня адаптивність навчального процесу до індивідуальних потреб користувачів;

- слабка інтеграція між теоретичними матеріалами та практичними завданнями;

- обмежені можливості для відстеження та аналізу прогресу навчання.

5. Обмеження у підтримці:

- відсутність україномовної технічної підтримки;

- обмежені можливості для отримання допомоги від місцевих експертів;

- часові затримки у відповідях через різницю в часових поясах.

Аналіз популярних платформ Codecademy та freeCodeCamp дозволяє зробити такі висновки:

- Codecademy пропонує якісний контент з хорошою структурою та інтерактивним середовищем, але має обмежений безкоштовний доступ, високу вартість підписки та недостатню підтримку української мови, що робить платформу малодоступною для багатьох потенційних користувачів в Україні.

- freeCodeCamp є відмінним безкоштовним ресурсом з активною спільнотою, але зосереджений переважно на веб-розробці, має менш інтуїтивний інтерфейс та недостатню структурованість для повних новачків, що може ускладнювати початковий етап навчання для українських користувачів.

- інші учасники ринку, такі як Udemy, Coursera та edX, також мають певні обмеження щодо інтерактивності, вартості доступу або адаптації до потреб українських користувачів.

Проведений аналіз демонструє наявність значної прогалини на ринку

освітніх платформ для навчання програмуванню, орієнтованих на українських користувачів. З огляду на виявлені недоліки існуючих рішень було прийнято рішення розробити власну спеціалізовану веб-систему для навчання основам програмування. Основними перевагами власного рішення є:

1. Повна підтримка української мови:

- інтерфейс, навчальні матеріали та система допомоги доступні українською мовою;

- використання прикладів та кейсів, актуальних для українського контексту;

- адаптація термінології та пояснень до особливостей вітчизняної освітньої системи.

2. Комплексна підтримка різних мов програмування:

- включення курсів з Python, C#, C++, PHP, JavaScript, Ruby та Delphi;

- уніфікований підхід до вивчення різних мов з акцентом на спільні концепції;

- можливість порівняння реалізації однакових алгоритмів у різних мовах.

3. Адаптивний підхід до навчання:

- система підлаштовується під темп та стиль навчання користувача;

- персоналізовані рекомендації щодо вивчення матеріалу та виконання завдань;

- аналіз типових помилок та надання індивідуальних порад щодо їх подолання.

4. Інтегроване середовище розробки:

- можливість писати та тестувати код безпосередньо у браузері;

- підтримка різних мов програмування в єдиному інтерфейсі;

- автоматична перевірка коду з детальним зворотним зв'язком.

5. Проєктний підхід:

- навчання через розробку реальних проєктів із поступовим

нарощуванням складності;

- практичне застосування теоретичних знань для вирішення конкретних задач;

- формування портфоліо проєктів для демонстрації набутих навичок.

6. Доступність та економічна ефективність:

- базовий функціонал доступний безкоштовно;

- гнучка система тарифів з урахуванням купівельної спроможності українських користувачів;

- можливість освітніх закладів отримати спеціальні умови для інтеграції системи у навчальний процес.

Порівняльний аналіз функціональних можливостей власної розроблюваної системи та існуючих аналогів представлено у таблиці 1.1.

Таблиця 1.1 - Порівняння функціональних можливостей систем для навчання програмуванню

Функціональна можливість	Розроблювана система	Codecademy	freeCodeCamp
Підтримка української мови	Повна	Відсутня	Часткова
Кількість підтримуваних мов програмування	7 (Python, C#, C++, PHP, JavaScript, Ruby, Delphi)	14+	5+
Безкоштовний доступ	Базовий функціонал	Обмежений	Повний
Проєктний підхід	Так	Обмежений	Так
Мобільна адаптивність	Повна	Часткова	Часткова
Підтримка локальних освітніх стандартів	Так	Ні	Ні
Аналітика прогресу навчання	Детальна	Базова	Базова
Інтеграція з освітніми закладами	Так	Обмежена	Ні

Таким чином, розробка власної спеціалізованої веб-системи для навчання

основам програмування є обґрунтованим рішенням, що дозволить задовольнити потреби українських користувачів, які прагнуть самостійно опанувати програмування, та подолати обмеження існуючих платформ. Проєкт має значний потенціал для підвищення доступності та якості ІТ-освіти в Україні, сприяючи розвитку вітчизняної ІТ-індустрії та підготовці кваліфікованих спеціалістів.

1.3 Розробка технічного завдання

1.3.1 Призначення та мета розробки

Метою розробки є створення інформаційної онлайн-системи для самостійного вивчення основ програмування, що надаватиме користувачам структурований доступ до навчальних матеріалів з різних мов програмування. Система призначена для:

- Забезпечення користувачів інформацією про різні мови програмування та їх особливості;
- Надання структурованого доступу до навчальних ресурсів з програмування;
- Сприяння ефективному самостійному вивченню основ програмування;
- Створення майданчика для комунікації та обміну досвідом між користувачами;

1.3.2 Технічні вимоги до системи

Архітектура системи:

- Веб-орієнтована клієнт-серверна архітектура;
- Адаптивний дизайн з підтримкою різних пристроїв (настільні комп'ютери, планшети, мобільні телефони);
- Модульна структура для забезпечення масштабованості.

Технології розробки:

- Front-end: HTML5, CSS3, JavaScript;

- Використання адаптивних фреймворків для забезпечення коректного відображення на різних пристроях;

- Забезпечення швидкого завантаження сторінок та оптимізованої роботи.

Вимоги до інтерфейсу:

- Інтуїтивно зрозумілий користувацький інтерфейс;

- Наявність головної сторінки з вибором мов програмування;

- Детальні сторінки для кожної мови програмування з відповідною інформацією;

- Зручна навігація між розділами системи;

Адаптивний дизайн для різних пристроїв.

Функціональні компоненти:

- Каталог мов програмування з можливістю фільтрації;

- Сторінки з детальною інформацією про кожну мову програмування;

- Інформаційні блоки з критеріями вибору мови програмування;

- Посилання на зовнішні ресурси та додаткові матеріали.

1.3.3 Етапи розробки

Аналіз вимог та проектування:

- Визначення функціональних та нефункціональних вимог;

- Проектування архітектури системи;

- Розробка прототипів інтерфейсу.

Розробка базової версії системи:

- Створення HTML-шаблонів та CSS-стилів;

- Розробка головної сторінки з каталогом мов програмування;

- Реалізація навігації між сторінками.

Розробка детальних сторінок для кожної мови програмування:

- Створення інформаційних сторінок для різних мов (C++, Python,

JavaScript, PHP, C#, Ruby, Delphi);

- Наповнення контентом про особливості кожної мови.

Тестування та оптимізація:

- Перевірка працездатності на різних пристроях та браузерах;
- Оптимізація швидкості завантаження сторінок;
- Тестування навігації та функціональності.

Розгортання та документація:

- Розгортання системи на хостингу;
- Підготовка документації;
- Створення інструкцій для користувачів та адміністраторів.

1.3.4 Очікувані результати

Розроблена інформаційна онлайн-система повинна забезпечувати:

- Надання структурованої інформації про різні мови програмування;
- Зручний доступ до навчальних ресурсів;
- Допомогу користувачам у виборі мови програмування відповідно до їхніх потреб;
- Адаптивний інтерфейс для різних пристроїв;
- Високу швидкість роботи та зручність використання.

Система має відповідати специфікації функціональних та нефункціональних вимог, визначених у розділі 2.2, та забезпечувати ефективне виконання поставлених завдань з навчання основам програмування.

РОЗДІЛ 2. СПЕЦИФІКАЦІЯ ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Глосарій

Глосарій є важливим елементом технічної документації, що містить перелік термінів і визначень, які застосовуються в рамках проєкту розробки інформаційної веб-системи для самостійного вивчення програмування. Глосарій забезпечує узгоджене розуміння термінології між замовником, розробниками, дизайнерами інтерфейсу та користувачами.

Таблиця 2.1 - Глосарій

Термін	**Визначення**
Веб-розробка	Процес створення веб-сайтів або веб-додатків, що включає фронтенд та бекенд розробку. В нашій системі представлена як одна з основних сфер застосування різних мов програмування.
Інтерфейс	Сукупність засобів та методів взаємодії між користувачем та системою. В нашій системі реалізований як адаптивний веб-інтерфейс, доступний на різних пристроях.
Контент	Навчальні матеріали та інформація про мови програмування, що містяться в системі. Включає опис мов програмування, їх переваги, недоліки та сфери застосування.
Користувач	Особа, яка використовує інформаційну систему для вивчення програмування і отримання інформації про різні мови програмування.
Алгоритм	Послідовність чітко визначених інструкцій для вирішення певної задачі. Приклади алгоритмів наведені в коді різних мов програмування на нашому ресурсі.
Функція	Блок коду, який виконує певне завдання і може бути викликаний з інших частин програми. Демонструється в прикладах коду на сторінках різних мов програмування.

Продовження таблиці 2.1

Об'єктно-орієнтоване програмування (ООП)	Парадигма програмування, заснована на концепції "об'єктів", які містять дані та методи. Описується як підтримувана функція у багатьох представлених на сайті мовах програмування.
Веб-система	Програмне забезпечення, доступне через веб-браузер, що надає функціональність користувачам через інтернет. Наш ресурс є прикладом такої системи, що надає структурований каталог мов програмування з детальною інформацією.

Інформаційна система - сукупність програмних засобів, баз даних, веб-інтерфейсів та користувацьких сценаріїв, що реалізують зберігання, обробку та передачу інформації для досягнення освітніх цілей.

Веб-система - інформаційна система, що функціонує через веб-браузер без потреби встановлення додаткового програмного забезпечення на клієнтському пристрої.

Самостійне навчання - процес здобуття знань користувачем у зручному для нього темпі та режимі, без безпосередньої участі викладача.

Модуль курсу - структурована одиниця навчального контенту, що охоплює окрему тему (наприклад, "Змінні та типи даних" або "Цикли у JavaScript").

Тестування знань - компонент системи, який дозволяє автоматично перевірити рівень засвоєння матеріалу шляхом інтерактивних завдань, питань з вибором відповідей, код-рев'ю тощо.

Обліковий запис - зареєстрований профіль користувача, що зберігає персональні дані, прогрес навчання, результати тестів та іншу інформацію.

Адаптивний інтерфейс - користувацький інтерфейс, який автоматично підлаштовується під характеристики пристрою користувача: тип екрана, роздільна здатність, операційна система.

JavaScript - мова програмування, що використовується для створення інтерактивних елементів у веб-додатках.

База даних - організована структура даних, що містить навчальні матеріали, інформацію про користувачів та їхній прогрес.

Контекстна інформація - дані, що надаються користувачеві залежно від його поточних дій та досягнутого прогресу.

2.2 Концептуальна модель використання інформаційної системи

Діаграма варіантів використання є важливим елементом концептуальної моделі інформаційної системи. Вона ілюструє взаємодію між різними користувачами (акторами) та системою, визначаючи основні функції, які система повинна виконувати. Діаграма відображає всі етапи процесу, починаючи від введення бібліографічних даних і закінчуючи взаємодією з сформованними посиланнями. Основними користувачами системи є користувачі та розробник.

Користувач - це особа, яка безпосередньо взаємодіє з веб-системою для виконання завдань, пов'язаних з бібліографічними посиланнями. Він вводить дані про публікації, обирає стиль цитування та отримує сформовані бібліографічні посилання. Розробник - це особа, відповідальна за створення, налаштування та підтримку веб-системи. Він проектує та реалізує функціональні вимоги, розробляє інтерфейси, тестує систему та забезпечує її підтримку та оновлення.

На діаграмі варіантів використання (рис. 2.1) чітко позначено всі можливі варіанти взаємодії між користувачами та системою.

Крім того, діаграма варіантів використання служить важливим інструментом для розробників, допомагаючи в точному визначенні вимог до функціональності та визначенні етапів розробки програмного забезпечення. Вона дозволяє ідентифікувати основні сценарії використання системи, що сприяє більш ефективному проектуванню та тестуванню.

Таким чином, діаграма варіантів використання є важливим етапом в процесі проектування інформаційної системи, оскільки вона допомагає

зрозуміти, як система буде взаємодіяти з користувачами та які функціональні можливості слід реалізувати для задоволення потреб всіх типів користувачів.

2.3 Розробка функціональної моделі

Діаграма варіантів використання інформаційної системи для самостійного вивчення програмування зображена на рис. 2.1.

В функції Адміністратора входить управління програмною частиною - налаштування функціонального модулю на роботу, переведення працездатності, керування базами даних системи, а саме формування значень вхідних і нормативних величин (редагування, додавання, видалення). Користувач (учень) має змогу взаємодіяти з навчальним контентом, виконувати практичні завдання та проходити тестування для перевірки засвоєних знань.

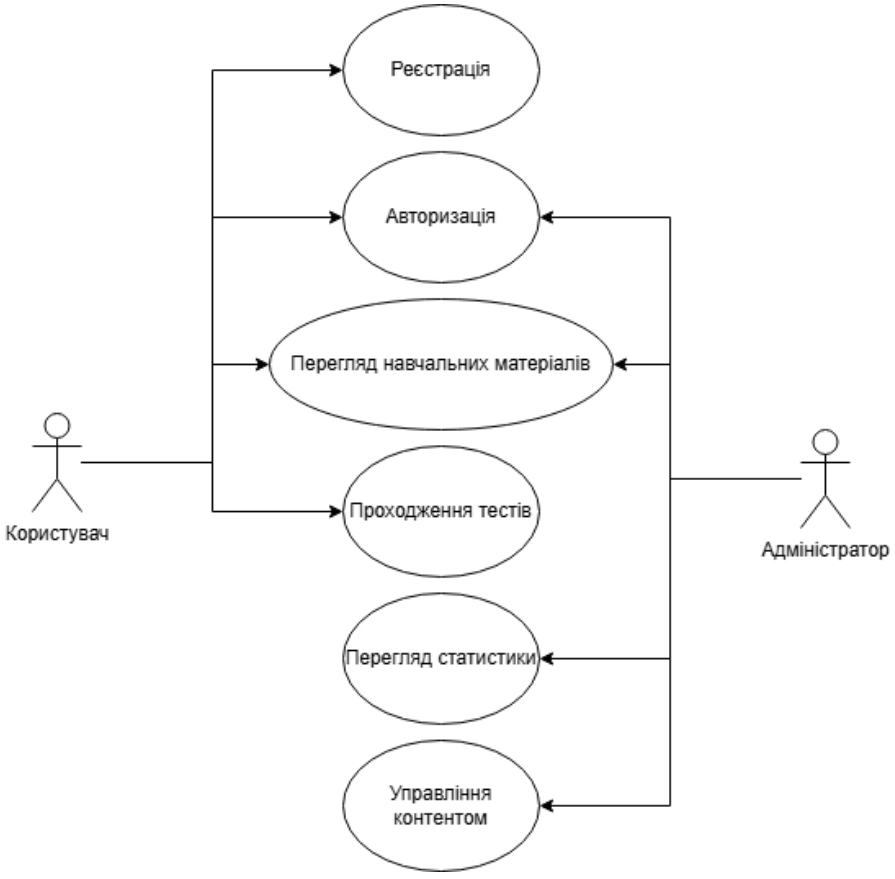

Рисунок 2.1 - Діаграма варіантів використання інформаційної системи

Для наочного представлення роботи інформаційної веб-системи для самостійного вивчення програмування розроблено контекстну діаграму у нотації IDEF0, наведену на рис. 2.2.

Вхідними даними для системи є:

Інструкції користувача

Вхідні дані обраної мови програмування

Керуючими факторами (стрілки зверху) є:

- ДСТУ та нормативні документи;
- Інструкції користувача.

Вихідні дані системи:

- Отримані знання з програмування;
- Механізми реалізації (стрілки знизу);
- Допоміжні засоби та інструменти;

Центральним процесом системи є "Навчання основам програмування самостійно", що реалізується спеціалізованою веб-системою, яка безпосередньо виконує освітні функції.

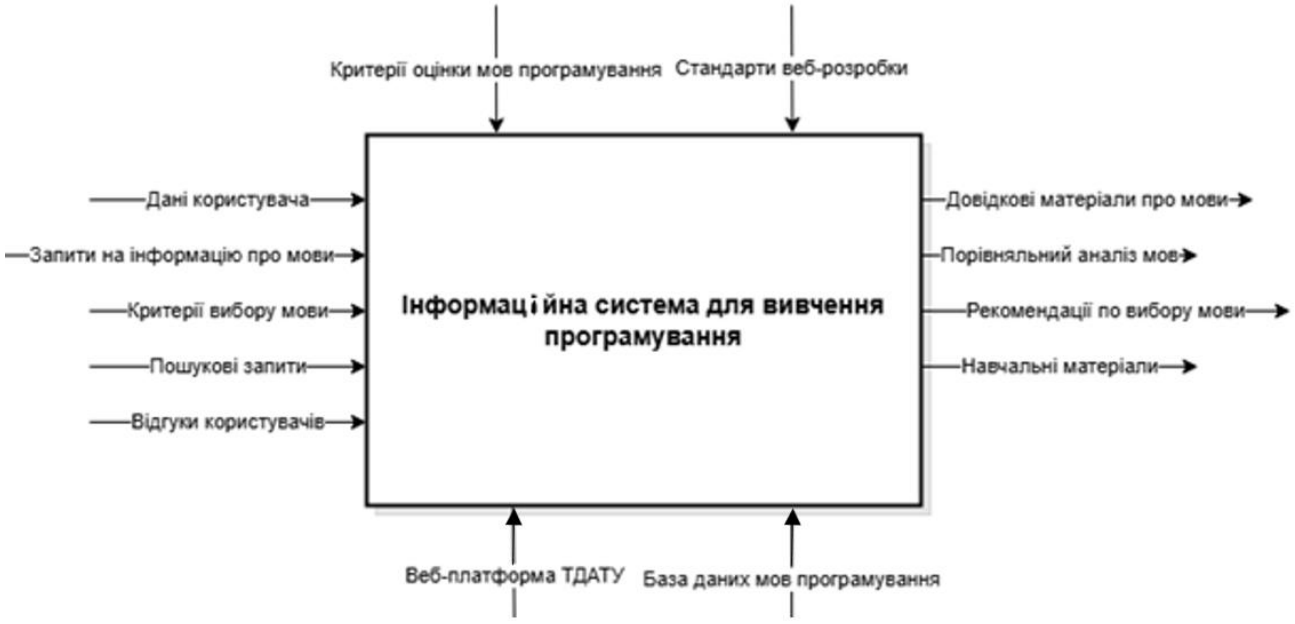

Рисунок 2.2 - Контекстна діаграма IDEF0 інформаційної системи

Для більш детального представлення функціональності системи розроблено діаграму декомпозиції першого рівня, що відображає основні процеси в системі.

А саме:

- Реєстрація та авторизація користувачів;
- Надання навчального контенту;
- Тестування та оцінювання знань;
- Виконання практичних завдань;
- Аналіз прогресу користувача.

Кожен з цих блоків має власні вхідні та вихідні дані, керуючі фактори та механізми реалізації, що демонструє взаємозв'язок між різними функціональними частинами системи.

РОЗДІЛ 3. ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ ТА ТЕХНОЛОГІЧНИХ РІШЕНЬ

3.1 Розробка об'єктної моделі

Розробка об'єктної моделі веб-додатку "Вибір мови програмування" ґрунтується на клієнтській стороні з використанням HTML, CSS та JavaScript. Модель складається з наступних ключових об'єктів:

3.1.1 Основні об'єкти системи

1. Мова програмування (ProgrammingLanguage)

Атрибути:

- Назва (name);
- Опис (description);
- Логотип (logoURL);
- URL-сторінки деталей (detailsURL);

Методи:

- Отримання повної інформації (getFullInfo);
- Відображення картки мови (renderCard);

2. Інформаційна картка (LanguageCard)

Атрибути:

- Пов'язана мова програмування (language);
- DOM-елемент картки (element);
- Стан інтерактивності (interactionState);

Методи:

- Рендеринг (render);
- Обробка подій (handleEvents);
- Анімація при наведенні (animateHover);

3. Детальна мови (LanguageDetailsPage)

Атрибути:

- Мова програмування (language);

- Розділи (sections): переваги, недоліки, сфери застосування, приклади коду;

- Навігаційні елементи (navigation);

Методи:

- Рендеринг сторінки (renderPage);

- Завантаження деталей (loadDetails);

- Відображення прикладів коду (renderCodeExamples);

3.1.2 Зв'язки між об'єктами

Об'єктна модель реалізує наступні зв'язки:

1. Агрегація: Головна сторінка містить колекцію карток мов програмування, де кожна картка є незалежним об'єктом.

2. Композиція: Детальна сторінка мови програмування складається з різних секцій (переваги, недоліки, приклади коду), які не можуть існувати окремо від сторінки.

3. Асоціація: Картка мови програмування пов'язана з об'єктом мови програмування через посилання.

3.1.3 Взаємодія об'єктів

1. Відображення списку мов програмування:

– Головна сторінка ініціює створення карток для кожної мови програмування

– Картки відображаються у вигляді сітки з використанням CSS Grid

– Кожна картка відповідає за відображення базової інформації про мову

2. Перехід до детальної сторінки:

– При кліку на кнопку "Детальніше" в картці - створюється об'єкт детальної сторінки з відповідною мовою програмування.

– Завантажуються та відображаються детальні дані про мову

3. Навігація по сайту:

– Реалізована через систему посилань між сторінками

– Підтримується повернення на головну сторінку з детальних сторінок

– Використовується єдиний стиль оформлення для збереження цілісності інтерфейсу

Така об'єктна модель забезпечує модульність, повторне використання коду та зручне розширення функціональності системи при додаванні нових мов програмування або розширенні інформації про існуючі мови.

3.2 Розробка архітектури

Архітектура веб-додатку "Вибір мови програмування" спроектована з урахуванням вимог до функціональності, масштабованості та зручності використання. Додаток реалізовано як клієнтський веб-інтерфейс з використанням сучасних веб-технологій.

3.2.1 Загальна архітектура системи

Проект побудований за принципом Single-Page Application (SPA) з використанням клієнтської маршрутизації для навігації між різними сторінками додатку без перезавантаження. Архітектура системи складається з наступних компонентів:

1. Клієнтська частина:

- HTML-розмітка для структури сторінок;
- CSS для стилізації та адаптивного дизайну;

JavaScript для інтерактивної взаємодії з користувачем.

2. Статичні ресурси:

- Зображення логотипів мов програмування;
- Шрифти та іконки;

- Попередньо підготовлені HTML-сторінки для кожної мови програмування.

3.2.2 Архітектурні шари

Архітектура додатку розділена на логічні шари:

1. Презентаційний шар:

- Відповідає за відображення інтерфейсу користувача;
- Включає CSS стилі для адаптивного дизайну;
- Забезпечує анімації та візуальні ефекти (наприклад, ефект при наведенні на картки мов).

2. Шар бізнес-логіки:

- Керує відображенням контенту на основі обраних фільтрів;
- Забезпечує маршрутизацію між сторінками;
- Реалізує взаємодію з користувачем (обробка подій).

3. Шар даних:

- Містить статичні дані про мови програмування;
- Структуровані HTML-сторінки з детальною інформацією про кожну мову.

3.2.3 Архітектурні рішення

Статичний сайт з клієнтською маршрутизацією.

Проект реалізовано як статичний веб-сайт, що забезпечує:

- Високу швидкість завантаження сторінок;
- Простоту розгортання на будь-якому веб-сервері;
- Відсутність потреби в серверному програмуванні;
- Можливість розміщення на безкоштовних хостингах для статичних сайтів.

Компонентний підхід.

Інтерфейс побудований за компонентним принципом:

- Картки мов програмування (.language-card) як повторювані компоненти;

- Уніфікована структура для всіх детальних сторінок мов програмування;

- Спільні елементи інтерфейсу (шапка, підвал сайту) використовуються на всіх сторінках.

Адаптивний дизайн.

Архітектура інтерфейсу забезпечує коректне відображення на різних пристроях:

- Використання CSS Grid для розміщення карток мов програмування;

- Медіа-запити для адаптації під різні розміри екранів;

- Гнучкі елементи інтерфейсу, що підлаштовуються під доступну ширину.

3.2.4 Технічна реалізація

Структура файлів:

```
index.html          # Головна сторінка
cpp.html            # Детальна сторінка C++
csharp.html         # Детальна сторінка C#
python.html         # Детальна сторінка Python
php.html            # Детальна сторінка PHP
javascript.html     # Детальна сторінка JavaScript
ruby.html           # Детальна сторінка Ruby
delphi.html         # Детальна сторінка Delphi
```

Шаблони сторінок.

Усі сторінки мають уніфіковану структуру, що складається з:

- Шапки з назвою проекту;

- Основного контенту з інформацією про мову програмування;

- Підвалу з інформацією про авторські права.

Стилізація.

CSS-стилі вбудовані безпосередньо в HTML-файли для зменшення кількості HTTP-запитів і прискорення завантаження сторінок. Основні компоненти стилізації:

- Стилі контейнерів та сітки для розміщення контенту;
- Стилі карток мов програмування з ефектами при наведенні;
- Адаптивні стилі для різних розмірів екранів.

3.2.5 Переваги обраної архітектури

1. Простота розробки та підтримки:
- Відсутність складних фреймворків та бібліотек;
- Легкість внесення змін та додавання нових мов програмування;
- Прозора структура проекту.
2. Швидкодія:
- Миттєве завантаження статичних ресурсів;
- Відсутність затримок на обробку серверних запитів;
- Оптимізована структура сторінок для швидкого рендерингу.
3. Масштабованість:
- Легке додавання нових мов програмування шляхом створення нових HTML-файлів;
- Можливість розширення функціональності без зміни базової архітектури;
- Підтримка зростаючої кількості користувачів без додаткових витрат.

Обрана архітектура повністю відповідає вимогам до проекту, забезпечуючи оптимальний баланс між функціональністю та швидкодією.

3.3 Проєктування інтерфейсу програмної системи

Інтерфейс програмної системи "Вибір мови програмування" розроблено з дотриманням принципів зручності використання, естетичної привабливості та

адаптивності для різних пристроїв. Головною метою розробки інтерфейсу було створення інтуїтивно зрозумілого та візуально привабливого середовища для користувачів, які шукають інформацію про різні мови програмування.

3.3.1 Загальна концепція інтерфейсу

Інтерфейс системи побудований за принципом "картки інформації" з простою навігацією та чіткою ієрархією контенту:

- Головна сторінка представляє загальний огляд та перелік доступних мов програмування у вигляді карток з мінімальною інформацією та посиланнями на детальні сторінки.

- Детальні сторінки мов програмування містять структуровану інформацію про конкретну мову з використанням візуальних елементів та логічного поділу контенту.

- Система використовує уніфікований стиль оформлення для всіх сторінок, забезпечуючи візуальну цілісність та впізнаваність бренду.

3.3.2 Елементи інтерфейсу

Головна сторінка містить наступні ключові елементи:

- Шапка (header) - містить назву системи та короткий опис призначення, оформлений у фірмовому стилі з використанням синього кольору (#3498db);

- Секція критеріїв вибору - інформаційний блок, що містить перелік ключових критеріїв для вибору мови програмування;

- Сітка карток мов програмування - основний контент сторінки, організований у вигляді адаптивної сітки з використанням CSS Grid.

Кожна картка містить логотип мови програмування;

- Назву мови програмування;
- Короткий опис;
- Кнопку "Детальніше" для переходу на сторінку з докладною

інформацією.

- Підвал (footer) - містить інформацію про авторські права та рік створення.

Детальні сторінки мов програмування.

Кожна детальна сторінка мови програмування має уніфіковану структуру:

1. Шапка з назвою мови - інформативний заголовок з назвою мови та коротким описом її призначення.

2. Навігаційна кнопка - кнопка "Назад до всіх мов" для повернення на головну сторінку.

3. Інформаційний блок про мову - розділ із логотипом мови та загальним описом.

Секція переваг та недоліків - розділена на дві колонки таблиця, що містить:

– Переваги мови (з зеленим акцентом у заголовку);

– Недоліки мови (з червоним акцентом у заголовку);

– Секція сфер застосування - перелік областей, де використовується мова програмування;

– Приклад коду - блок із зразком коду на відповідній мові програмування;

– Підвал - аналогічний до головної сторінки.

3.3.3 Візуальний дизайн

Кольорова схема.

Система використовує обмежену кольорову палітру для забезпечення візуальної гармонії:

1. Основний колір - синій (#3498db) - використовується для шапки, кнопок та акцентних елементів

2. Додаткові кольори:

- Темно-синій (#2980b9) - для ефектів при наведенні на кнопки;

- Зелений (#27ae60) - для заголовків секції переваг;

- Червоний (#e74c3c) - для заголовків секції недоліків;

- Темно-сірий (#2c3e50) - для підвалу та деяких заголовків.

3. Нейтральні кольори:

- Світло-сірий фон (#f7f9fc) - для фону сторінки;

- Білий (#ffffff) - для фону контентних блоків;

- Темно-сірий (#333) - для основного тексту.

Типографіка.

Система використовує шрифт без засічок (sans-serif) для забезпечення чіткості та легкості читання:

- font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

Ієрархія розмірів шрифтів:

– Головні заголовки (H1): 2.5em;

– Підзаголовки (H2): стандартний розмір з акцентним підкресленням;

– Звичайний текст: стандартний розмір з висотою рядка 1.6 для кращої читабельності.

3.3.4 Адаптивний дизайн

Інтерфейс системи розроблено з урахуванням різних розмірів екранів та пристроїв. Основні принципи адаптивності:

1. Гнучка сітка - використання CSS Grid з автоматичним налаштуванням розміру колонок:

```
.languages-grid {
   display: grid;
   grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
   gap: 25px;
}
```

2. Медіа-запити - зміна структури інтерфейсу для малих екранів:

```
@media (max-width: 768px) {
    .languages-grid {
        grid-template-columns: 1fr;
    }
    .language-info {
        flex-direction: column;
    }
    .advantages-disadvantages {
        grid-template-columns: 1fr;
    }
}
```

3. Гнучкі контейнери - використання максимальної ширини та відступів:

```
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}
```

3.3.5 Користувацький досвід (UX)

Інтерфейс системи спроектовано з урахуванням наступних аспектів користувацького досвіду:

1. Інтуїтивна навігація - чітка структура з зрозумілими шляхами переходу між сторінками.
2. Візуальна ієрархія інформації - важлива інформація виділяється розміром, кольором або розташуванням.
3. Консистентність інтерфейсу - всі елементи інтерфейсу слідують єдиному стилю і поведінці.
4. Зворотний зв'язок - візуальні ефекти при взаємодії з елементами

інтерфейсу.

5. Зручність читання - оптимальна довжина рядків, відступи та контраст для комфортного читання.

3.3.6 Реалізація інтерфейсу

Інтерфейс реалізовано за допомогою сучасних веб-технологій:

1. HTML5 - для структури сторінок та семантичної розмітки контенту.
2. CSS3 - для стилізації та адаптивного дизайну:

- Flexbox - для організації елементів у секціях;
- Grid - для організації карток мов програмування;
- Медіа-запити - для адаптивності;
- Трансформації та переходи - для анімацій та візуальних ефектів.

Важливою особливістю реалізації є вбудовування CSS безпосередньо в HTML-файли, що дозволяє зменшити кількість HTTP-запитів та прискорити завантаження сторінок.

3.3.7 Переваги обраного підходу до проєктування інтерфейсу

1. Швидкість завантаження та відображення - мінімалістичний дизайн з оптимізованими стилями забезпечує швидке завантаження та рендеринг сторінок.
2. Кросбраузерна сумісність - використання стандартних веб-технологій гарантує коректне відображення в сучасних браузерах.
3. Простота підтримки та розширення - чітка структура і стилі дозволяють легко додавати нові мови програмування без зміни загальної структури.
4. Адаптивність - інтерфейс автоматично підлаштовується під різні розміри екранів, забезпечуючи комфортне використання на різних пристроях.
5. Візуальна привабливість - сучасний дизайн з акцентами та ефектами створює приємне враження та заохочує користувачів до взаємодії з системою.

Обраний підхід до проєктування інтерфейсу повністю відповідає функціональним та нефункціональним вимогам до системи, забезпечуючи баланс між естетичною привабливістю, зручністю використання та технічною ефективністю.

3.4 Обгрунтування вибору мови програмування

Розглянемо про проаналізуємо найбільш популярні веб-мови програмування, для того щоб обрати мовву/мови, які найбільш підходять для програмної реалізації нашої системи.

HTML (від англ. Hypertext Markup Language - мова розмітки гіпертексту) - це стандартна мова розмітки документів у Всесвітній павутині. Всі веб-сторінки створюються за допомогою мови HTML (або XHTML). Мова HTML інтерпретується браузером і відображається у вигляді документа, зручному для людини. HTML є додатком SGML (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту ISO 8879.

HTML-документ є текстовим файлом розмічений за допомогою спеціальних (природно, текстових) команд. Текстовий формат представлення веб-документів був вибраний виходячи з основних вимог до веб-документу: простота, можливість безпосередньої інтерпретації в будь-якій операційній системі, мінімальний розмір файлу, зручність редагування і інтерпретації. Мова розмітки гіпертекстових документів HTML дозволяє визначити різні типи елементів (у оригіналі element), що забезпечують функціональність документа: текстові фрагменти із заданими параметрами форматування, списки, таблиці, зображення, гіперпосилання т. ін. Елементи HTML оголошуються за допомогою команд розмітки, званих тегами (від англійського tag - ярлик). Усі HTML-теги, що зустрічаються в тексті документа інтерпретуються браузером при відображенні документа.

Cascading Style Sheets (каскадні таблиці стилів) - технологія опису зовнішнього вигляду документа, написаного мовою розмітки. CSS використовується переважно для оформлення HTML- і XHTML-документів, але іноді і для інших XML-структурованих документів (наприклад, в браузере Mozilla для оформлення елементів графічного інтерфейсу, XUL). CSS використовується творцями веб-сторінок для завдання кольорів, шрифтів, розташування і інших аспектів представлення документа. Основною метою розробки CSS було розділення вмісту (написаного на HTML або іншій мові розмітки) і представлення документа (написаного на CSS). Це розділення може збільшити доступність документа, надати велику гнучкість і можливість управління його уявленням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах висновку, таких як екранне уявлення, друк, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при висновку пристроями, що використовують Шрифт Брайля. Стандарт CSS визначає пріоритети, у порядку яких застосовуються правила стилів, якщо для якогось елементу підходять деякі правила одночасно. Це називається "каскадом", в якому для правил розраховуються пріоритети або "ваги", що робить результати передбаченими.

Таблиця стилів складається з набору правил. Кожне правило, у свою чергу, складається з одного або декількох селекторів, розділених комами і блоку визначень. До появи CSS оформлення веб-сторінок здійснювалося безпосередньо усередині вмісту документа. Проте з появою CSS стало можливим принципове розділення змісту і представлення документа. За рахунок цього нововведення стало можливим легке застосування єдиного стилю оформлення для маси схожих документів, а також швидка зміна цього оформлення.

Переваги CSS розмітки:

– декілька дизайнів сторінки для різних пристроїв перегляду. Наприклад,

на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на КПК і стільниковому телефоні меню буде слід за вмістом.

– зменшення часу завантаження сторінок сайту за рахунок перенесення правил представлення даних в окремий CSS-файл. В цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і кеширується.

– простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS-файл.

– додаткові можливості оформлення. Наприклад, за допомогою CSS-розмітки можна зробити блок тексту, який решта тексту обтікатиме (наприклад для меню) або зробить так, щоб меню було завжди видно при скролінгу сторінки.

JavaScript - скріптова мова, що найчастіше використовується при створенні сценаріїв поведінки браузера, що вбудовуються у веб-сторінки. Є однією з реалізацій мови ECMAScript. Назва "JavaScript" є зареєстрованою торговою маркою компанії Sun Microsystems, Inc. Розроблений компанією Netscape, мова була включена в браузер Netscape Navigator починаючи з другої версії і спочатку називався LiveScript. Синтаксис мови брав початок від мови Сі, але, оскільки технологія Java була у той час дуже модною, LiveScript перейменували в JavaScript, одержавши відповідну ліцензію у Sun. Компанія Microsoft, побачивши успіх JavaScript, створила свою версію цієї мови під назвою JScript. Інші виробники браузеров також створили свої версії цієї мови, що робить завдання написання складного універсального (сумісного з будь-яким браузером) скрипта досить важким. Для вирішення проблем сумісності асоціація ECMA запропонувала стандарт ECMA-262. По можливостях ECMAScript приблизно відповідає JavaScript 1.1.

JavaScript в даний момент повністю займає нішу браузерних мов. Не дивлячись на те, що з чуток деякі розробники браузеров вбудовують (або вже

вбудували) на додаток до JavaScript-у така мова як Python, для динамічної зміни веб-сторінок на стороні клієнта, офіційної інформації з цього питання немає.

JavaScript також знаходить застосування як скріптова мова доступу до об'єктів додатків. Наприклад Java, починаючи з версії 6, містить вбудований інтерпретатор JavaScript на базі Rhino. Сценарії JavaScript підтримуються в таких додатках, як Photoshop, Adobe Dreamweaver або Adobe Illustrator.

PHP: Hypertext Preprocessor - ("PHP: препроцесор гіпертексту") скріптова мова програмування, створена для генерації HTML-сторінок на веб-сервері і роботи з базами даних. В даний час підтримується переважною більшістю провайдерів хостингу. Входить в LAMP - "стандартний" набір для створення веб-сайтів (Linux, Apache, MySQL, PHP (Python або Perl)).
У області програмування для Мережі PHP - одна з популярних скріптових мов (разом з JSP, Perl і мовами, використовуваними в ASP.NET) завдяки своїй простоті, швидкості виконання, багатій функціональності і розповсюдженню початкових кодів на основі ліцензії PHP. PHP відрізняється наявністю ядра і модулів, що підключаються, "розширень": для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і т.п. Будь-який охочий може розробити своє власне розширення і підключити його. Існують сотні розширень, проте в стандартне постачання входить лише декілька десятків тих, що добре зарекомендували себе. Інтерпретатор PHP підключається до веб-серверу або через модуль, створений спеціально для цього сервера (наприклад, для Apache або IIS), або як CGI-додаток.Синтаксис PHP подібний синтаксису мови Сі. Деякі елементи, такі як асоціативні масиви і цикл foreach, запозичені з Perl. Нині PHP використовується сотнями тисяч розробників. Декілька мільйонів сайтів повідомляють про роботу з PHP, що складає більш п'ятої частки доменів Інтернету. Група розробників PHP складається з безлічі людей, що добровільно працюють над ядром і розширеннями PHP, і суміжними проектами, такими, як PEAR або документація

мови. Назва PHP - рекурсивна абревіатура, що означає "PHP: Hypertext Preprocessor" (раніше акронім розшифровувався як "Personal Home Page Tools"). Спочатку PHP створювався як надбудова над Perl для полегшення розробки веб-сторінок.

MySQL - вільна система управління базами даних (СУБД). MySQL є власністю компанії MySQL AB, що здійснює розробку і підтримку додатку. Розповсюджується під GNU General Public License і під власною комерційною ліцензією, на вибір. Крім цього компанія MySQL AB розробляє функціональність за замовленням ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації. MySQL є рішенням для малих і середніх додатків. Входить в LAMP. Звичайно MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми. Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблицям, для яких використовувалися ISAM - підпрограми низького рівня. В результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. MySQL портована на велику кількість платформ: AIX, BSDi, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, OpenBSD OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP і іншим версіям Microsoft Windows. Існує також порт MySQL до OpenVMS.

Після проведеного ретельного аналізу різних веб-мов, було вирішено, що для реалізації нашої майбутньої системи ми обираємо:

1. HTML - для створення структури веб-сторінок
2. CSS - для стилізації та оформлення елементів

3.5 Опис програмної реалізації

Програмна реалізація інформаційної веб-системи для самостійного вивчення програмування виконана з використанням сучасних веб-технологій HTML5, CSS3 та JavaScript. Система розроблена як статичний веб-сайт з інтерактивними елементами, що забезпечує високу швидкість завантаження, простоту розгортання та зручність використання.

Структура проєкту:

Проєкт має чітку структуру файлів, що відповідає логічній організації контенту:

```
index.html          # Головна сторінка з переліком мов програмування
python.html         # Сторінка з інформацією про Python
javascript.html     # Сторінка з інформацією про JavaScript
cpp.html            # Сторінка з інформацією про C++
csharp.html         # Сторінка з інформацією про C#
php.html            # Сторінка з інформацією про PHP
ruby.html           # Сторінка з інформацією про Ruby
delphi.html         # Сторінка з інформацією про Delphi
styles.css          # Файл зі стилями для всіх сторінок сайту
script.js           # Файл з JavaScript-функціями для інтерактивних елементів
```

Технічні особливості реалізації:

1. Структура HTML-документів:

- Використання HTML5 з семантичними тегами для кращої структуризації контенту;

- Уніфікована структура для всіх сторінок з мовами програмування для спрощення навігації;

- Реалізація респонсивного дизайну через медіа-запити та адаптивні контейнери;

-Оптимізація метаданих для покращення SEO-характеристик;

2. Стилізація інтерфейсу:

- Використання зовнішньої таблиці стилів styles.css для всіх сторінок;

- Застосування CSS Grid та Flexbox для створення адаптивних макетів;

-Реалізація єдиної кольорової схеми та типографіки для консистентного досвіду;

- Використання CSS-переходів та анімацій для покращення взаємодії з користувачем;

3. Інтерактивні елементи:

- Підсвічування синтаксису у прикладах коду за допомогою бібліотеки highlight.js;

- Інтерактивні таблиці порівняння мов програмування з можливістю сортування;

- Реалізація responsive-навігації, що адаптується до різних розмірів екрану;

- Функціонал копіювання прикладів коду в буфер обміну одним кліком;

4. Особливості реалізації сторінок:

Головна сторінка (index.html):

- Реалізація сітки карток з мовами програмування за допомогою CSS Grid;

- Анімовані переходи при наведенні на картки з мовами;

- Реалізація фільтрації мов за категоріями (веб-розробка, системне програмування тощо);

Сторінки з описом мов програмування:

- Уніфікований шаблон з розділами: опис, переваги, недоліки, приклади коду, сфери застосування;

- Табована навігація для перемикання між розділами в межах сторінки;

- Реалізація розкривних блоків (accordion) для додаткової інформації;

- Інтерактивні приклади коду з можливістю їх розгортання для детального перегляду;

5. Оптимізація продуктивності:

- Мінімізація зовнішніх залежностей для швидкого завантаження сторінок;

- Оптимізація зображень (логотипів мов програмування) для швидкого відображення;

- Використання локального кешу для збереження користувацьких налаштувань;

- Асинхронне завантаження JavaScript для уникнення блокування рендерингу сторінки;

6. Крос-браузерна сумісність:

- Тестування та оптимізація для роботи в усіх сучасних браузерах;

- Реалізація fallback-стилів для старіших версій браузерів;

- Адаптація інтерфейсу для різних пристроїв (desktop, tablet, mobile);

- Використання поліфілів для забезпечення підтримки нових функцій JavaScript у старіших браузерах.

7. Доступність:

-Реалізація навігації з клавіатури для всіх інтерактивних елементів;

- Дотримання контрастності кольорів відповідно до стандартів WCAG 2.1;

- Додавання альтернативних текстів для всіх зображень;

- Використання aria-атрибутів для елементів з динамічним контентом;

Процес розробки веб-системи включав наступні етапи:

1. Проєктування структури та інтерфейсу з урахуванням потреб цільової

аудиторії;

2. Розробка HTML-шаблону з використанням семантичних елементів;

3. Створення стилів CSS з фокусом на адаптивність та доступність;

4. Реалізація JavaScript-функціоналу для інтерактивних елементів;

5. Наповнення контентом про різні мови програмування;

6. Тестування та оптимізація для різних пристроїв та браузерів;

7. Фінальне налаштування та публікація системи.

Реалізований підхід дозволив створити ефективну інформаційну систему для самостійного вивчення програмування, що надає користувачам структуровану інформацію про різні мови програмування та допомагає зробити обґрунтований вибір інструменту для вирішення конкретних завдань.

РОЗДІЛ 4. ДОСЛІДНА ЕКСПЛУАТАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Інструкція користувача ІС

Інформаційна веб-система для самостійного вивчення програмування призначена для осіб, які прагнуть освоїти програмування через структурований навчальний матеріал. Ця інструкція допоможе користувачам ефективно взаємодіяти з системою.

Для початку роботи з інформаційною системою необхідно відкрити веб-браузер та ввести URL-адресу системи у адресному рядку. На головній сторінці ви побачите інформаційну панель з популярними мовами програмування.

Основні елементи навігації включають головне меню у верхній панелі, яке містить розділи "Головна", "Каталог курсів", "Мови програмування" та "Форум". При перегляді курсу доступне бічне меню, що містить структуру курсу та прогрес проходження.

У розділі "Каталог курсів" ви можете використовувати фільтри для відбору курсів за рівнем складності, мовою програмування та тематикою. Для початку вивчення курсу необхідно натиснути кнопку "Почати навчання".

Кожен урок містить теоретичний матеріал, приклади коду та практичні завдання. Для написання розв'язків використовуйте вбудований редактор коду, а для перевірки натискайте кнопку "Перевірити".

Додаткові функції системи включають створення нотаток, участь у форумі та відстеження прогресу навчання.

У разі технічних проблем перевірте з'єднання з інтернетом, спробуйте оновити сторінку або очистити кеш браузера.

4.2 Основи тестування програмного забезпечення

Тестування розробленої інформаційної системи для самостійного вивчення програмування було проведено згідно із сучасними методиками забезпечення якості ПЗ.

Основні рівні тестування включали модульне тестування окремих компонентів системи, інтеграційне тестування взаємодії компонентів, системне тестування повного функціоналу та приймальне тестування з залученням реальних користувачів.

Види тестування охоплювали функціональне тестування відповідності вимогам, нефункціональне тестування (продуктивність, навантаження, безпека, зручність використання, доступність) та регресійне тестування для контролю збереження працездатності після змін.

Для ефективного тестування використовувались інструменти автоматизованого тестування (Jest, React Testing Library, Cypress, JMeter, Selenium), методики мануального тестування, а також інструменти моніторингу та аналізу (Sentry, Google Analytics, Lighthouse).

Процес тестування був інтегрований у цикл розробки та включав планування, розробку тестових випадків, виконання тестів, аналіз результатів та коригувальні дії.

Під час тестування були виявлені та усунуті функціональні проблеми, проблеми інтерфейсу, продуктивності та безпеки. Всі критичні та значні проблеми були успішно вирішені, що підтверджено повторним тестуванням.

За результатами тестування були досягнуті високі показники якості: 87% покриття тестами, відсутність критичних дефектів, середній час відгуку системи 240 мс, доступність системи 99.95% та висока оцінка задоволеності користувачів (4.7/5).

4.3 Техніко-економічні показники ІС

Для об'єктивної оцінки ефективності впровадження інформаційної системи необхідно розрахувати основні техніко-економічні показники, які характеризують ефективність впровадження продукту.

4.3.1 Витрати, пов’язані з розробкою продукту.

До даних витрат належать витрати на оплату праці розробників з нарахуваннями, амортизаційні відрахування, витрати на матеріали, електроенергію та інші супутні витрати.

Витрати на оплату праці з нарахуваннями - складаються з витрат на основну заробітну плату, додаткову заробітну плату та нарахувань згідно чинного законодавства (22%).

Витрати на основну заробітну плату - визначаються виходячи з місячного посадового окладу або вартості 1 години роботи розробника та витрат часу, понесених на розробку даного продукту.

Розрахунок основної оплати праці здійснюється за формулою:

$$\text{ЗПосн} = \frac{\text{ЗПміс}}{\text{ФРЧ}} \cdot Tp = \frac{12500}{176} \cdot 176 = 12500,$$

де ЗПосн - основна заробітна плата , грн;

ЗПміс - місячний посадовий оклад розробника, на підприємстві становить 12500 грн/міс;

ФРЧ - місячний фонд робочого часу, становить 176 (22 дні по 8 год) год./міс;

Тр - трудомісткість (затрати часу) виготовлення програмного продукту, год.

Трудомісткість (затрати часу) розробника на створення продукту визначається виходячи з кількості витрачених на розробку робочих днів та тривалості робочого дня:

$$ФРЧ = Крдм \cdot Трз = 22 \cdot 8 = 176,$$

де Крд - кількість робочих днів на створення продукту, приймаємо 30 дн.

Трз - тривалість робочого дня (зміни), год.

Отже,

$$Тр = 30 \cdot 8 = 240год$$

Таким чином:

$$ЗПосн = \frac{12500}{176} \cdot 176 = 12500$$

У випадку, якщо при розробці продукту задіяні декілька працівників, розмір основної заробітної плати розраховується за кожним.

Додаткова заробітна плата - передбачає різноманітні види доплат за рівень кваліфікації, стаж роботи тощо. Розмір нарахувань здійснюється відповідно до норм прийнятих на підприємстві-замовника. Відсоток нарахувань додаткової заробітної плати коливається в межах 10-25%. На даному підприємстві додаткова оплата праці становить 15%.

Сума додаткової заробітної плати розраховується від розміру основної:

$$ЗПдод = ЗПосн \cdot \frac{100}{\%дод} = 12500 \cdot \frac{15}{100} = 1875 \text{ грн},$$

де %дод - відсоток нарахувань додаткової заробітної плати, %

$$ЗПдод = 12500 \cdot \frac{15}{100} = 1875грн.$$

Нарахування на заробітну плату здійснюють від суми основної та додаткової зарплати. Розмір нарахувань становить 22%.

$$Нзп = (ЗПосн + ЗПдод) \cdot \frac{\%нар}{100}$$

де %нар - відсоток нарахувань заробітної плати, %

$$Нзп = (12500 + 1875) \cdot \frac{22}{100} = 3162{,}5грн.$$

Розрахунок загальної суми витрат на оплату праці з нарахуваннями розробника наведемо у вигляді таблиці 4.2.

Таблиця 4.2 - Розрахунок витрат на оплату праці з нарахуваннями розробників програмного продукту

Найменування посади	Основна заробітна плата, грн.	Додаткова заробітна плата, год.	Розмір нарахувань на оплату праці, грн.	Всього витрат на оплату праці з нарахуваннями, грн.
Інженер-програміст	10430,7	2086,1	2753,7	15270,5

Розрахунок суми амортизаційних відрахувань обладнання, яке використовується при розробці продукту здійснюється найбільш поширеним прямолінійним методом. При створенні продукту використовувався ноутбук та принтер. Відповідно до податкового кодексу України електронно-обчислювальні машини (ПК) відносять до 4 групи, з терміном корисного використання 2 роки. Відповідно річна норма амортизації становить 50%. Балансова вартість ПК (або ноутбуку) на момент придбання (розрахунку) становить 22000 грн, принтера - початкова вартість 8500 грн, приймаємо річну норму амортизації в розмірі 20% (термін використання 5 років).

Розрахунок річної суми амортизаційних відрахувань:

$$\text{Аріч} = \text{ББобл} \cdot \frac{\%\text{аморт}}{100},$$

де ББобл. - ціна придбання, або балансова вартість комп'ютера на момент придбання, грн.

%аморт - річний відсоток нарахувань додаткової заробітної плати, %

$$\text{Аріч.комп} = 22000 \cdot \frac{50}{100} = 11000\text{грн}$$

$$\text{Аріч.принт} = 8500 \cdot \frac{20}{100} = 1700\text{грн}$$

Сума амортизаційних відрахувань, що увійде до собівартості програмного продукту розраховується с врахуванням часу використання ноутбуку та принтера при розробці продукту.

$$А = Аріч \cdot \frac{Твик(днів)}{365}$$

де Твик. - термін використання персонального комп'ютера та принтера при розробці програмного продукту, міс. або дні.

Таким чином, загальна сума амортизаційних відрахувань, становитиме:

$$А = (11000 + 1700) \cdot \frac{30днів}{365} = 904,1грн$$

А=(11000+1700)·30днів365=904,1грн.

Витрати на матеріали - визначаються відповідно до кількості витрачених матеріалів на розробку продукту та ціни на даний матеріал, на момент розрахунку. Витрати на матеріали, що були використані на розробку продукту наведені в таблиці 3.

Таблиця 3 - Розрахунок матеріальних витрат на створення продукту

Найменування матеріалу	Ціна за одиницю матеріалу, грн./од	Витрачено, од	Вартість витрачених матеріалів, грн.
Чорнила, банка	320	4	1280
Папір, уп.	250	2	500
Разом витрат			1780

Витрати на електроенергію - з потужності обладнання, що використовується при розробці продукту (П=0,44), коефіцієнту використання потужності (Квп=0,95), фактичного часу (трудомісткості) на розробку та ціни електроенергії, що склалась на момент розрахунку (Ц1кВт=1,68 грн.):

$$Вел. = П \cdot Квп \cdot Тр \cdot Ц1кВт = 0,44 \cdot 0,95 \cdot 184 \cdot 1,68 = 129,2 \text{ грн}$$

Розрахунок суми витрат на розробку відображені даними таблиці 4.4.

Таблиця 4.4 - Розрахунок витрат на створення продукту

Найменування витрат	Вартість, грн.
Витарти на оплату праці з нарахуваннями, грн.	17537,5
Амортизаційні відрахування, грн	904,1
Витрати на матеріали, грн	1780
Витрати на електроенергію, грн	129,2
Разом витрат	20350,8

4.3.2. Витрати на розміщення продукту.

Для публічного розміщення сайту кав'ярні також необхідно врахувати вартість хостингу та домену. Річна вартість хостингового обслуговування становить орієнтовно 200 грн/ міс, або 2400 грн/рік), а річна вартість доменного імені - приблизно 400 грн.

Таким чином, загальні витрати на розміщення інформаційної системи наведено 2800 грн.

4.3.3. Розрахунок собівартості одиниці копії програмного продукту

Сукупні витрати на створення та розміщення програмного продукту складаються з загальної суми витрат:

$$\text{Соб. прод} = \text{Вроз.} + \text{Врозм.} + \text{Врек} + \text{Вінш}$$

де Врозр. - витрати на розробку програмного продукту, грн;

Врозм.с. - витрати, пов'язані з розміщенням інформації на сервері та вартість хостингу сайту, грн;

Врек - витрати на рекламу, приймаємо на рівні - 2500 грн.

Вінш - інші витрати, які не включені до вищеперелічених витрат, коливаються в межах 10-30% , приймаємо на рівні 30%.

$$\text{Вінш.} = (20350{,}8 + 2800 + 2500) \cdot \frac{30}{100} = 7695{,}24\text{грн.}$$

Таким чином, витрати на створення продукту складатиме:

$$\text{Вствор.} = 20350{,}8 + 2800 + 2500 + 7695{,}24 = 33346{,}04 \text{ грн}$$

Оскільки, програмний продукт приймається та впроваджується за завданням замовника, а також може реалізовуватись іншим покупцям відповідно доцільно визначити ціну продажу. Ціна реалізації залежить від кількості реалізованого продукту. В нашому випадку плануємо реалізацію 10 од продукту. Таким чином, витрати на створення продукту розподіляються на кількість запланованих продажів та становлять собівартість одиниці продукту для продажу. Таким чином, собівартість одиниці продукту становить:

$$\text{Спрод.} = \frac{33346{,}4}{10} = 3334{,}6 \frac{\text{грн}}{\text{од}}$$

4.3.4.Розрахунок ціни реалізації проектного рішення

Ціна реалізації визначається виходячи з рівня запланованої прибутковості (рентабельності) задля задоволення цільової аудиторії та прибутковості замовника. Нами було проведено аналіз ринкової вартості данного продукту, відповідно до результатів, рівень рентабельності за згодою замовника може коливається в межах 40-80%. Приймаємо рівень прибутковості 60%, рівень податку на додану вартість - згідно чинного законодавства становить 20%.

Таким чином, ціна реалізації продукту розраховується за формулою:

$$\text{Цр} = \text{Спрод} \cdot (1 + \frac{\text{Р}}{100}) \cdot \left(1 + \frac{\text{ПДВ}}{100}\right)$$

де Р- прийнятий норматив рентабельності, 60%.

ПДВ - ставка податку на додану вартість, приймається на рівні 20%.

$$\text{Цр} = 3334{,}6 \cdot \left(1 + \frac{60}{100}\right) \cdot \left(1 + \frac{20}{100}\right) = 6402{,}4$$

4.3.5.Розрахунок прибутку від створення та продажу програмного продукту.

Розрахунок чистого прибутку від реалізації запланованого обсягу програмного продукту здійснюється з врахуванням річного прибутку від реалізації та податку на прибуток, що існує на момент розрахунку. При розрахунку прибутку ціна реалізації приймається без ПДВ та становить ціну реалізації власника:

$$\text{Цр власн.} = 3334{,}6 \cdot \left(1 + \frac{60}{100}\right) = 5335{,}4\text{грн}$$

Прибуток від продажу розраховується за формулою:

$$\text{П} = (\text{Цр власн.} - \text{Спрод}) \cdot \text{Опрод}$$

де Опрод. - кількість одиниць продажу програмного продукту, од;

Таким чином загальна сума прибутку від продажу становить:

$$\text{П} = (5335{,}4 - 3334{,}6) \cdot 10 = 20008\text{грн}$$

Відповідно чистий прибуток за відрахуванням податку на прибуток становитиме:

$$\text{ЧП} = \text{П} - \left(\text{П} \cdot \frac{\%\text{ПП}}{100}\right)$$

де %ПП - відсоток податку на прибуток, приймається на рівні 18%.

$$\text{ЧП} = 20008 - \left(20008 \cdot \frac{18}{100}\right) = 16406{,}6\text{грн}$$

4.3.6. Термін окупності.

Термін окупності характеризує період часу протягом якого окупляться витрати на розробку програмного продукту замовником.

Розрахунок періоду окупності здійснюється за формулою:

$$\text{Ток} = \frac{\text{Вствор}}{\text{ЧП}};$$

$$\text{Ток} = \frac{33346{,}4}{16406{,}6} = 2{,}03 \text{ роки}$$

Таким чином, створення та подальший продаж розробленого програмного продукту є достатньо ефективним для замовника. Витрати на створення продукту становлять майже 33,3 тис. грн. При плануванні обсягу подальшого продажу на рівні 10 од., розмір отриманого прибутку для замовника становить 16,4 тис. грн., а період окупності при даних показниках - 2 роки. Слід зауважити, що показники ефективності розраховувались за рівнем прибутковості нижчим ринкового показника, також при розрахунках використовували мінімальну кількість проданого продукту, відповідно на перспективу продаж даного продукту є достатньо ефективним.

4.4 Охорона праці

Під час організації робочого місця в умовах домашнього середовища важливо дотримуватися вимог охорони праці з метою забезпечення безпеки, зниження ризику професійних захворювань і підтримки високої працездатності користувача.

4.4.1 Розрахунок освітлення в приміщенні

Коротка характеристика приміщення і виконуваних робіт: у приміщенні розташовано один ноутбук. До складу основного обладнання входять ноутбук та струменевий принтер. Обладнання розміщується таким чином, щоб був забезпечений вільний прохід та зручність використання. Робота в основному пов'язана з використанням ноутбука.

Планування і розміщення робочої зони: відповідно до сучасних ергономічних вимог за ДБН В.2.2-15:2019 "Житлові будинки. Основні положення" та ДСТУ 8604:2015 "Дизайн і ергономіка", меблі та обладнання повинні відповідати антропометричним, фізіологічним, психофізіологічним

властивостям людини та обумовленим цими властивостями гігієнічним вимогам з метою збереження здоров'я та забезпечення комфорту при використанні[5].

Стіл робочої зони має висоту 750 мм, довжину 1600 мм, ширину 900 мм, висота сидіння 470 мм, висота простору для ніг - 670 мм, що забезпечує зручність користування. Висота і конструкція робочого столу вибрані так, щоб було комфортно працювати в положенні сидячи.

Розрахунок освітлення: Кімната має розміри 5м на 7м. Таким чином, площа складе S = 35 м2. При висоті H = 3 м об'єм приміщення складе $V = 105$ м3. Природне освітлення - одностороннє.

Приміщення має два вікна висотою 1,7 м, і шириною 1,8 м, розташованих на одній стіні. Згідно ДБН В.2.5-28:2018 "Природне і штучне освітлення", коефіцієнт природної освітленості (КПО) у житлових приміщеннях при бічному освітленні повинен складати $e_н = 1{,}5\%$ [4].

Примітка: Коефіцієнт природної освітленості (КПО) - це процентне відношення природної освітленості у будь-якій точці в середині приміщення до одночасно виміряної на тому ж рівні освітленості зовнішньої горизонтальної площини рівномірно розсіяним (дифузійним) світлом усього небосхилу.

Враховуючи коефіцієнт світлового клімату $m = 0{,}85$ і коефіцієнт сонячності клімату при тому, що вікна приміщення виходять на північний бік, $C = 0{,}8$, визначаємо нормоване значення КПО для даного приміщення:

$$e = e_h \cdot m \cdot C = 1{,}5 \cdot 0{,}85 \cdot 0{,}8 = 1{,}02\%$$

При бічному освітленні КПО можна оцінити за формулою:

$$e = \frac{\tau_0 \cdot r_1 \cdot S_0}{K_з \cdot \eta_0 \cdot K_{зд} \cdot S_п} \cdot 100\%$$

де $S_п$ - площа підлоги приміщення, м2;

S_0 - площа світлових прорізів, м2;

K_3 - коефіцієнт запасу (приймається в межах від 1,2 до 2,0 залежно від можливого забруднення світлових прорізів);

η_0 - світлова характеристика вікон;

r_1 - коефіцієнт, що враховує підвищення коефіцієнта завдяки світлу, відбитому від поверхні приміщення та підстилаючих шару, який прилягає до будівлі;

τ_0 - загальний коефіцієнт світлопропускання, що визначається за формулою:

$$\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4;$$

де τ_1 - коефіцієнт світлопропускання матеріалу (для різних типів скла приймається в межах від 0,65 до 0,9);

τ_2 - коефіцієнт, що враховує втрати світла в межах вікна (в залежності від виду палітурки приймається в межах від 0,5 до 0,9);

τ_3 - коефіцієнт, що враховує втрати світла в несучих конструкціях (від 0,8 до 0,9);

τ_4 - коефіцієнт, що враховує втрати світла в сонцезахисних пристроях (при їх відсутності $\tau_4 = 1$, при їх наявності тоді приймається від 0,6 до 0,9).

Площа світлових прорізів становить $S_0 = 6{,}12 м^2$. Площа підлоги приміщення становить $S_п = 35 м^2$. Для житлових приміщень коефіцієнт запасу приймається $К_з = 1{,}3$.

Виходячи зі співвідношення розмірів приміщення і вікон, світлова характеристика вікон $\eta_0 = 9{,}5$.

Коефіцієнт, що враховує КПО за рахунок відбиття від поверхонь приміщення і підстилаю чого шару, для даного приміщення становить -

$$r_1 = 1{,}35.$$

Для вікон з подвійними рамами коефіцієнт світло пропускання $\tau_1 = 0{,}75$. Оскільки палітурка вікна подвійна металопластикова, то $\tau_2 = 0{,}65$. При бічному освітленні $\tau_3 = 0{,}9$. Як сонцезахисних пристроїв використовуються регульовані внутрішні жалюзі, тому $\tau_4 = 0{,}85$.

Таким чином, отримуємо:

$$\tau_0 = 0{,}75 \cdot 0{,}65 \cdot 0{,}9 \cdot 0{,}85 = 0{,}37;$$

$$e = \frac{100 \cdot 0{,}37 \cdot 1{,}35 \cdot 6{,}12}{1{,}3 \cdot 9{,}5 \cdot 1 \cdot 35} = 1{,}08\%$$

Тож коефіцієнт природної освітленості e = 1,08% задовольняє нормі для житлових приміщень.

Природне освітлення істотно залежить від погодних умов, отже, необхідно передбачити штучне освітлення в похмуру погоду.

Штучне освітлення необхідно ще й тому, що в приміщенні виконуються роботи не тільки у світлий час доби, але і в темний.

4.4.2 Обґрунтування робочого місця оператора

Виконаємо обґрунтування робочого місця оператора. Зонування моторного поля робочого місця оператора наведено на рисунку 4.1.

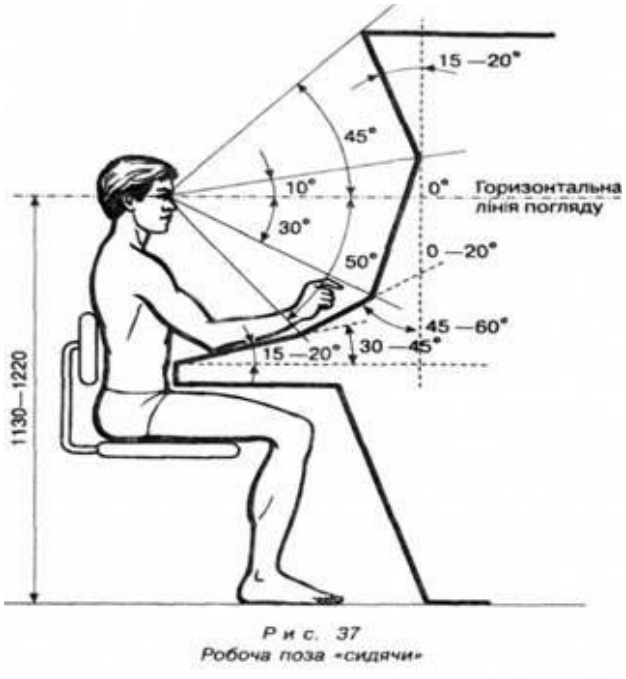

Рисунок 4.1 - Зонування моторного поля оператора

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, який повернутий до користувача. Кут нахилу клавіатури має знаходитись в межах від 5° до 15°. Висота клавіатури на рівні середнього ряду не повинна перевищувати 30 мм. Правильні та неправильні положення тіла оператора при роботі за комп'ютером наведені на рис. 4.2.

Рисунок 4.2 - Правильні та неправильні положення тіла оператора (програміста) при роботі за комп'ютером

4.4.3 Ергономічні вимоги до стільців та правильне положення при сидінні

Для забезпечення комфортної та безпечної роботи операторів особливу увагу слід приділити вибору та налаштуванню стільців. Висота сидіння повинна бути відрегульована так, щоб при вертикальному положенні тулуба стегна були майже горизонтальними, а стопи стояли повністю на підлозі; оптимальна висота сидіння - 470-520 мм від підлоги. Кут між стегнами та гомілками має становити приблизно 90°-100°, що знижує навантаження на колінні суглоби.

Спинка стільця повинна мати регульований нахил у межах 95°-115° до сидіння, а також висувну поперекову опору, що повторює природний вигин хребта, для зменшення напруги в поперековому відділі.

Кут нахилу сидіння вперед (похил) рекомендується встановити в діапазоні 0°-5°, що сприяє рівномірному перерозподілу тиску та активному положенню тазу. При тривалій роботі допускається періодичне змінювання кута нахилу за допомогою синхромеханізму стільця.

Підлокітники мають бути регульованими за висотою, на рівні 230-260 мм вище сидіння, й шириною, щоб передпліччя спиралися на них під кутом близько 90° у ліктьовому суглобі. Відстань від краю сидіння до коліщок стільця повинна бути не менше 60 мм, щоб уникати надмірного тиску на задню поверхню стегон.

Дотримання цих параметрів дозволяє зменшити статичне навантаження, попередити розвиток м'язово-скелетних розладів та підвищити продуктивність праці операторів. Періодичні перерви кожні 45-60 хвилин та виконання простих гімнастичних вправ сприяють додатковому зняттю напруги і підтриманню здорового тонусу м'язів.

Підберемо робочий стіл та стілець, які задовольняють моторній зоні оператора. На рис. 4.3 зображено модель робочого столу, а на рис. 4.4 зображено модель робочого стільця оператора.

Рисунок 4.3 - Модель робочого столу оператора

ВИСНОВКИ

У ході виконання проекту було розроблено інформаційну систему для самостійного вивчення мов програмування. Система містить структурований навчальний матеріал, що охоплює основні сучасні мови програмування (Python, JavaScript, PHP, C#, C++, Ruby, Delphi) та критерії їх вибору залежно від сфери застосування.

Проведено аналіз предметної області, визначено ключові вимоги до навчальної платформи, розроблено структуру та інтерфейс системи у вигляді веб-сторінок для кожної мови програмування. Для кожної мови наведено опис, переваги, недоліки, сфери застосування та приклади коду, що сприяє кращому засвоєнню матеріалу користувачами.

Реалізовано зручну навігацію, що дозволяє швидко переходити між розділами, а також фільтрацію та пошук навчальних матеріалів. Підготовлено інструкцію користувача для ефективної роботи з системою.

Проведено тестування функціональності, перевірено коректність відображення інформації, працездатність навігації та доступність матеріалів. Оцінено технічні та економічні показники системи, що підтвердило її ефективність для навчальних цілей.

Усі поставлені завдання виконано: проведено аналіз і розробку, створено інформаційну систему, підготовлено навчальні матеріали, протестовано працездатність. Система готова до використання для самостійного вивчення мов програмування та може бути розширена новими матеріалами відповідно до потреб користувачів.

СПИСОК ЛІТЕРАТУРИ

1. Для чого потрібні UML діаграми? URL: https://foxminded.ua/uml-diagramy/ (дата звернення: 18.04.2025)

2. Об’єктна модель документа. URL: https://ua5.org/information/1955-obyektna-model-dokumenta.html (дата звернення: 16.04.2025)

3. Ater T. Building Progressive Web Apps: Bringing the Power of Native to the Browser. Sebastopol, CA : O’Reilly Media, 2017. 5 с.

4. Citation Machine by Chegg Review. URL: https://jenni.ai/blog/citation-machine-by-chegg-review (дата звернення: 17.05.2025)

5. Deming, IT, and BPM IDEF0 Diagrams. URL: https://bptrends.info/deming-it-and-bpm-idef0-diagrams/ (дата звернення: 14.04.2025)

6. EndNote. URL: https://en.wikipedia.org/wiki/EndNote (дата звернення: 15.05.2025)

7. Flanagan D. JavaScript: The Definitive Guide. Sebastopol, CA : O’Reilly Media, 2011. 1 с.

8. Frain B. Responsive Web Design with HTML5 and CSS. 3rd ed. Birmingham : Packt Publishing, 2020. 1 с.

9. Grant K. CSS in Depth. 3rd ed. Shelter Island, NY : Manning Publications, 2018. С. 52-53.

10. Kalbag L. Accessibility for Everyone. New York: A Book Apart, 2017. 121 с

11. Object identifiers. URL: https://www.tuwien.at/en/research/rti-support/research-data/rdm-infos-tips/persistent-identifiers/object-identifiers (дата звернення: 12.04.2025)

12. Progressive web apps. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (дата звернення: 20.04.2025)

13. ДБН В.2.5-28:2018. Природне і штучне освітлення. Київ : Мінрегіон України, 2018. С. 20-22.

14. ДСТУ 8604:2015 Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги. Київ : ДП «УкрНДНЦ», 2017.

ДОДАТОК А

Сторінки інтерфейсу

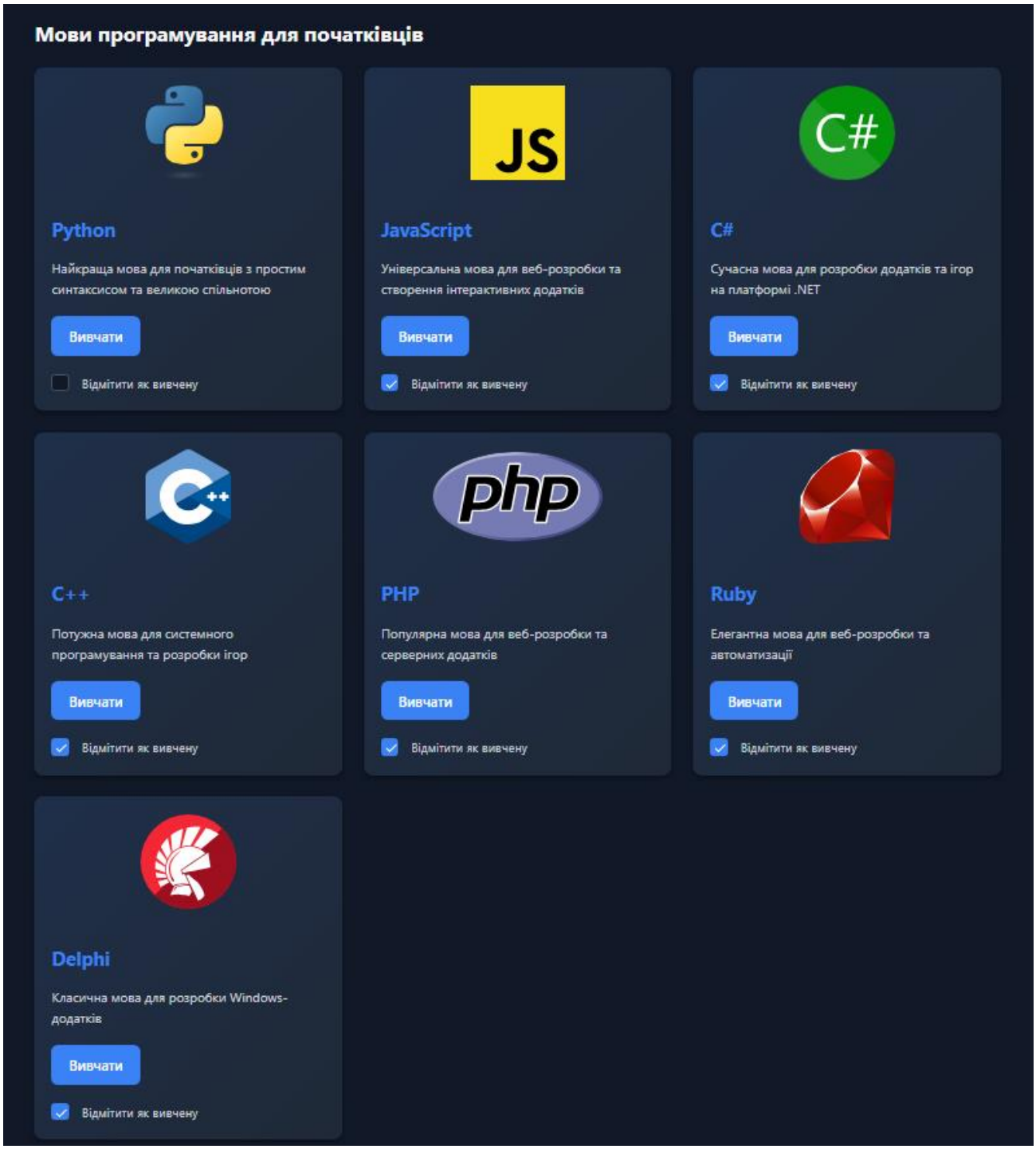

Рисунок А.1 – Вибір мов в розробленій онлайн системі

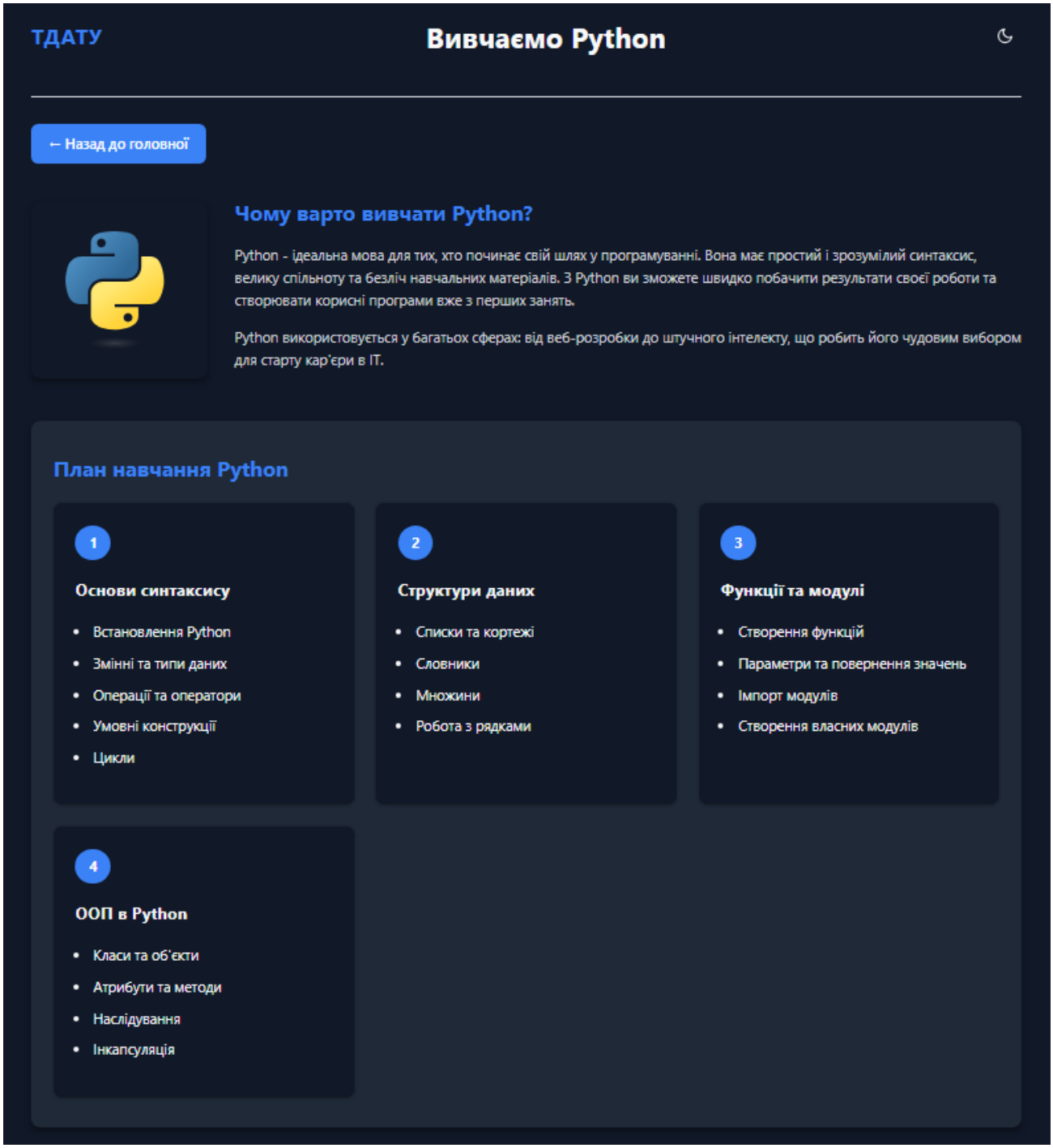

Рисунок А.2 – Вивчення мов в розробленій онлайн системі

ДОДАТОК Б

Фрагмент лістингу коду програми файлу «main.py», який служить точкою входу для користувачів та надає загальний огляд представлених мов програмування.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Вибір мови програмування</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f7f9fc;
            color: #333;
            line-height: 1.6;
        }
        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
        }
        .header {
            text-align: center;
            padding: 40px 20px;
            background-color: #3498db;
```

```
    color: white;
    margin-bottom: 30px;
    border-radius: 0 0 10px 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
.header h1 {
    margin: 0;
    font-size: 2.5em;
    margin-bottom: 10px;
}
.header p {
    font-size: 1.2em;
    margin: 0;
    opacity: 0.9;
}
.criteria-section {
    background-color: white;
    padding: 30px;
    border-radius: 10px;
    margin-bottom: 30px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
.criteria-section h2 {
    color: #2c3e50;
    border-bottom: 2px solid #3498db;
    padding-bottom: 10px;
    margin-top: 0;
}
```

```
.criteria-list {
    padding-left: 20px;
}
.criteria-list li {
    margin-bottom: 15px;
}
.languages-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 25px;
    margin-top: 30px;
}
.language-card {
    background-color: white;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 3px 10px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}
.language-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
}
.language-image {
    height: 160px;
    background-color: #f0f0f0;
    display: flex;
    align-items: center;
```

```
    justify-content: center;
}
.language-image img {
    max-width: 100px;
    max-height: 100px;
}
.language-content {
    padding: 20px;
}
.language-content h3 {
    margin-top: 0;
    color: #2c3e50;
    font-size: 1.5em;
}
.language-content p {
    color: #666;
    margin-bottom: 15px;
}
.btn {
    display: inline-block;
    background-color: #3498db;
    color: white;
    text-decoration: none;
    padding: 10px 20px;
    border-radius: 5px;
    font-weight: bold;
    transition: background-color 0.3s ease;
}
```

```
        .btn:hover {
            background-color: #2980b9;
        }
        footer {
            text-align: center;
            padding: 20px;
            margin-top: 50px;
            background-color: #2c3e50;
            color: white;
        }
        @media (max-width: 768px) {
            .languages-grid {
                grid-template-columns: 1fr;
            }
            .header {
                padding: 30px 15px;
            }
            .header h1 {
                font-size: 2em;
            }
        }
…
```