ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА
Навчально-науковий інститут енергетичної, інформаційної
та транспортної інфраструктури

Кафедра комп'ютерних наук та інформаційних технологій

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Проєктування вебзастосунку для вивчення мов програмування з елементами гейміфікації»

Виконала: здобувачка вищої освіти 4 курсу,
групи КН 2022-1
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

Вероніка ЖУВАКО
(ім'я та прізвище)

Керівник: Наталія БРАТЕРСЬКА
(ім'я та прізвище)

Рецензент ас. Валерія ЧЕРКАСОВА
(ім'я та прізвище)

м. Харків – 2026 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ
Марина НОВОЖИЛОВА
«__23__» ___червня____2026 Року

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

Жувако Вероніки Григорівни
(прізвище, ім'я, по батькові)

1. Тема роботи Проєктування вебзастосунку для вивчення мов програмування з елементами гейміфікації

керівник роботи ас. Братерська Наталія Миколаївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «22» травня 2026 р. № 440-03
2. Термін подання студентом роботи 22.06.2026р.
3. Вихідні дані до роботи Проєктування та розробка вебзастосунку для вивчення мов програмування з елементами гейміфікації з використанням Python, HTML, CSS, JavaScript, сучасних засобів веброзробки та бази даних для збереження інформації про користувачів, навчальні матеріали, результати виконання завдань і прогрес навчання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та аналіз існуючих платформ для навчання програмування; проєктування архітектури вебзастосунку (діаграми варіантів використання, класів, компонентів, ER-діаграма бази даних, алгоритми роботи системи); розробка програмного забезпечення вебзастосунку (реєстрація та авторизація користувачів, проходження навчальних курсів, виконання практичних завдань, система оцінювання результатів, реалізація елементів гейміфікації, відстеження прогресу користувача та взаємодія з віртуальним персонажем); тестування системи та розробка керівництва користувача; аналіз питань охорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Презентація – 15 аркушів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Наталія БРАТЕРСЬКА	11.05.2026	15.05.2026
Розділ II	Наталія БРАТЕРСЬКА	15.05.2026	26.05.2026
Розділ III	Наталія БРАТЕРСЬКА	26.05.2026	01.06.2026
Розділ IV	Вікторія МАЛИШЕВА	02.06.2026	05.06.2026

7. Дата видачі завдання 11.05.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми кваліфікаційної роботи	11.05.2026	Виконано
2	Затвердження тем, наукових керівників, завдань та календарного плану підготовки дипломної роботи	13.05.2026	Виконано
3	Написання I розділу	15.05.2026	Виконано
4	Написання II розділу	26.05.2026	Виконано
5	Написання III розділу	01.06.2026	Виконано
6	Написання IV розділу	06.06.2026	Виконано
7	Подання дипломної роботи керівнику	07.06.2026	Виконано
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	14.06.2026	Виконано
9	Подання доопрацьованого варіанту роботи керівнику	15.06.2026	Виконано
10	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	24.06.2026	Виконано

Здобувач _____ (підпис) Вероніка ЖУВАКО (прізвище та ініціали)

Керівник роботи _____ (підпис) Наталія БРАТЕРСЬКА (прізвище та ініціали)

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи бакалавра студента групи КН 2022-1 спеціальності 122 Комп'ютерні науки Жувако Вероніки Григорівни за темою «Проєктування вебзастосунку для вивчення мов програмування з елементами гейміфікації» складається з 4 розділів, містить 46 рисунків, 5 таблиць, 22 джерел.

Кваліфікаційну роботу присвячено проєктуванню та розробці вебзастосунку для вивчення мов програмування із застосуванням елементів гейміфікації, спрямованих на підвищення мотивації користувачів та ефективності навчального процесу.

У розділі «Загальні положення» наведено опис предметної області, проаналізовано сучасні підходи до навчання програмування, досліджено існуючі освітні платформи та гейміфіковані аналоги, визначено мету і завдання дослідження.

У другому розділі виконано аналіз предметної області, визначено вхідні та вихідні дані системи, спроєктовано базу даних, побудовано об'єктно-орієнтовану модель, розроблено математичне й алгоритмічне забезпечення вебзастосунку.

У третьому розділі наведено опис використаних засобів розробки, вимог до технічного та програмного забезпечення, реалізації архітектури вебзастосунку, навчальних модулів, системи реєстрації користувачів, механізмів перевірки практичних завдань, елементів гейміфікації, користувацького інтерфейсу та керівництва користувача.

У розділі «Охорона праці» розглянуто питання охорони праці під час роботи з комп'ютерною технікою та запропоновано заходи щодо забезпечення безпечних умов праці.

Ключові слова: ВЕБЗАСТОСУНОК, МОВИ ПРОГРАМУВАННЯ, ГЕЙМІФІКАЦІЯ, PYTHON, НАВЧАЛЬНА ПЛАТФОРМА, ПРОГРЕС КОРИСТУВАЧА, ІНТЕРАКТИВНЕ НАВЧАННЯ.

ANNOTATION

Structure and scope of the thesis. The explanatory note of the bachelor's qualification thesis prepared by a student of group KN 2022-1, specialty 122 Computer Science, Zhuvako Veronika Hryhorivna entitled "Design and development of a web application for learning programming languages with gamification elements" consists of 4 chapters, contains 46 figures, 5 tables, and 22 references.

The bachelor's qualification thesis is devoted to the design and development of a web application for learning programming languages using gamification elements aimed at increasing user motivation and improving the effectiveness of the learning process.

The chapter "General Provisions" describes the subject area, analyzes modern approaches to programming education, reviews existing educational platforms and gamified solutions, and defines the purpose and objectives of the research.

The second chapter presents the analysis of the subject area, identifies the system input and output data, designs the database, develops the object-oriented model, and describes the mathematical and algorithmic support of the web application.

The third chapter describes the development tools used, the software and hardware requirements, the implementation of the web application architecture, learning modules, user registration system, mechanisms for assessing practical tasks, gamification elements, the user interface, and the user manual.

The chapter "Occupational Health and Safety" addresses occupational safety issues related to working with computer equipment and proposes measures to ensure safe working conditions.

Keywords: WEB APPLICATION, PROGRAMMING LANGUAGES, GAMIFICATION, PYTHON, EDUCATIONAL PLATFORM, VIRTUAL CHARACTER, USER PROGRESS, INTERACTIVE LEARNING.

ЗМІСТ

ВСТУП

У сучасному світі стрімкого розвитку інформаційних технологій та цифровізації освіти особливої актуальності набуває створення ефективних інструментів для навчання програмування, особливо серед дітей та підлітків. Зростання попиту на ІТ-фахівців зумовлює необхідність пошуку нових підходів до засвоєння знань, які не лише надають доступ до навчальних матеріалів, а й підтримують інтерес користувачів протягом усього процесу навчання.

Традиційні методи вивчення мов програмування часто характеризуються низьким рівнем залученості та складністю сприйняття абстрактного матеріалу. Це призводить до втрати інтересу на початкових етапах навчання. Одним із підходів до вирішення цієї проблеми є гейміфікація – використання ігрових механік у навчальному процесі, таких як рівні, винагороди, персонажі та система прогресу.

Актуальність роботи полягає у створенні вебзастосунку, який поєднує навчальний контент із елементами гейміфікації, що дозволяє підвищити мотивацію та залученість користувачів.

Метою кваліфікаційної роботи є розробка вебзастосунку для вивчення основ програмування з використанням елементів гейміфікації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області та існуючих програмних рішень;
- визначити функціональні та нефункціональні вимоги до вебзастосунку;
- спроєктувати архітектуру системи та її основні компоненти;
- обґрунтувати вибір інструментальних засобів розробки;
- реалізувати функціональний прототип вебзастосунку;
- провести тестування та оцінку працездатності системи.

Об’єктом дослідження є процес навчання мов програмування з використанням вебтехнологій.

Предметом дослідження є методи та засоби проєктування вебзастосунків з елементами гейміфікації.

У процесі виконання роботи використовуються такі методи дослідження: аналіз наукових джерел та існуючих аналогів, методи системного аналізу, об’єктно-орієнтованого проєктування, а також сучасні технології веброзробки.

Практичне значення отриманих результатів полягає у створенні готового функціонального прототипу вебзастосунку, який може використовуватися як самостійний навчальний інструмент або як основа для подальшої розробки повноцінної освітньої платформи.

Теоретична цінність роботи полягає у узагальненні підходів до застосування гейміфікації у навчальних вебзастосунках для вивчення програмування.

Результатом кваліфікаційної роботи є функціональний прототип вебзастосунку для вивчення основ програмування на прикладі мови Python з елементами гейміфікації, який включає систему авторизації користувачів, інтерактивні міні-уроки, вбудований редактор коду, систему рівнів та досвіду, інтерактивного віртуального персонажа, що еволюціонує залежно від прогресу, а також механізми відстеження навчального прогресу.

Отримані результати мають прикладне значення у контексті створення навчальних вебзастосунків, оскільки дозволяють підвищити ефективність засвоєння базових знань з програмування завдяки використанню інтерактивних та гейміфікованих підходів. Розроблений прототип може бути використаний як основа для подальшого розвитку більш комплексних освітніх систем.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Сучасний світ характеризується стрімким розвитком інформаційних технологій, що зумовлює високий попит на ІТ-фахівців. Володіння основами програмування сьогодні розглядається як одна з ключових компетенцій XXI століття. За даними OECD, computational thinking та базові навички програмування дедалі частіше інтегруються в освітні програми різних рівнів, починаючи зі шкільного навчання, оскільки вони сприяють розвитку логічного мислення, креативності та здатності вирішувати складні задачі [1].

Цільовою аудиторією розроблюваного вебзастосунку є початківці, які вперше стикаються з програмуванням. Особливої уваги потребують діти та підлітки шкільного віку, для яких важлива наочна, інтерактивна та мотивуюча форма подачі матеріалу. Водночас система може бути корисною і для інших категорій початківців, які бажають опанувати програмування у доступній формі.

На сьогодні вивчення програмування відбувається у кількох основних формах: у загальноосвітніх навчальних закладах, шляхом самостійного навчання (YouTube-канали, блоги, документація) та через спеціалізовані онлайн-платформи. Зростання популярності онлайн-освіти підтверджується сучасними статистичними даними, які фіксують стабільне збільшення частки дистанційних та гібридних форматів навчання [2].

Однак навіть сучасні цифрові платформи не завжди забезпечують високий рівень мотивації та залученості користувачів. Саме тому все більшої популярності набувають рішення, що поєднують навчальний контент з елементами гейміфікації. Такі системи дозволяють зробити процес навчання більш захопливим, динамічним та ефективним [4].

Отже, зростання ролі цифрових технологій в освіті та необхідність підвищення мотивації до вивчення програмування обумовлюють актуальність розробки вебзастосунків, які поєднують якісний навчальний матеріал з ігровими механіками.

1.1.1 Опис процесу діяльності

Традиційне вивчення програмування в закладах загальної середньої освіти здебільшого відбувається на уроках інформатики. Такий підхід забезпечує певну системність навчального процесу, проте має обмеження, пов’язані з великою кількістю учнів у групі та недостатньою можливістю індивідуалізації навчання. У результаті матеріал часто подається у теоретичній та абстрактній формі, що ускладнює його засвоєння.

Самостійне навчання програмуванню через відеоуроки, онлайн-курси та навчальні ресурси вимагає високого рівня самодисципліни та внутрішньої мотивації. Для значної частини учнівської аудиторії це є складним фактором, що призводить до нерегулярного навчання або його припинення. Онлайн-платформи частково вирішують проблему структуризації матеріалу, однак не завжди забезпечують достатній рівень залученості та емоційної взаємодії з користувачем.

У процесі навчання програмування можна умовно виділити типову послідовність дій користувача: ознайомлення з матеріалом, виконання практичних завдань, отримання зворотного зв’язку та закріплення навичок через повторення. На кожному з цих етапів можуть виникати труднощі, пов’язані зі складністю абстрактних понять, недостатньою мотивацією та відсутністю миттєвого результату навчання.

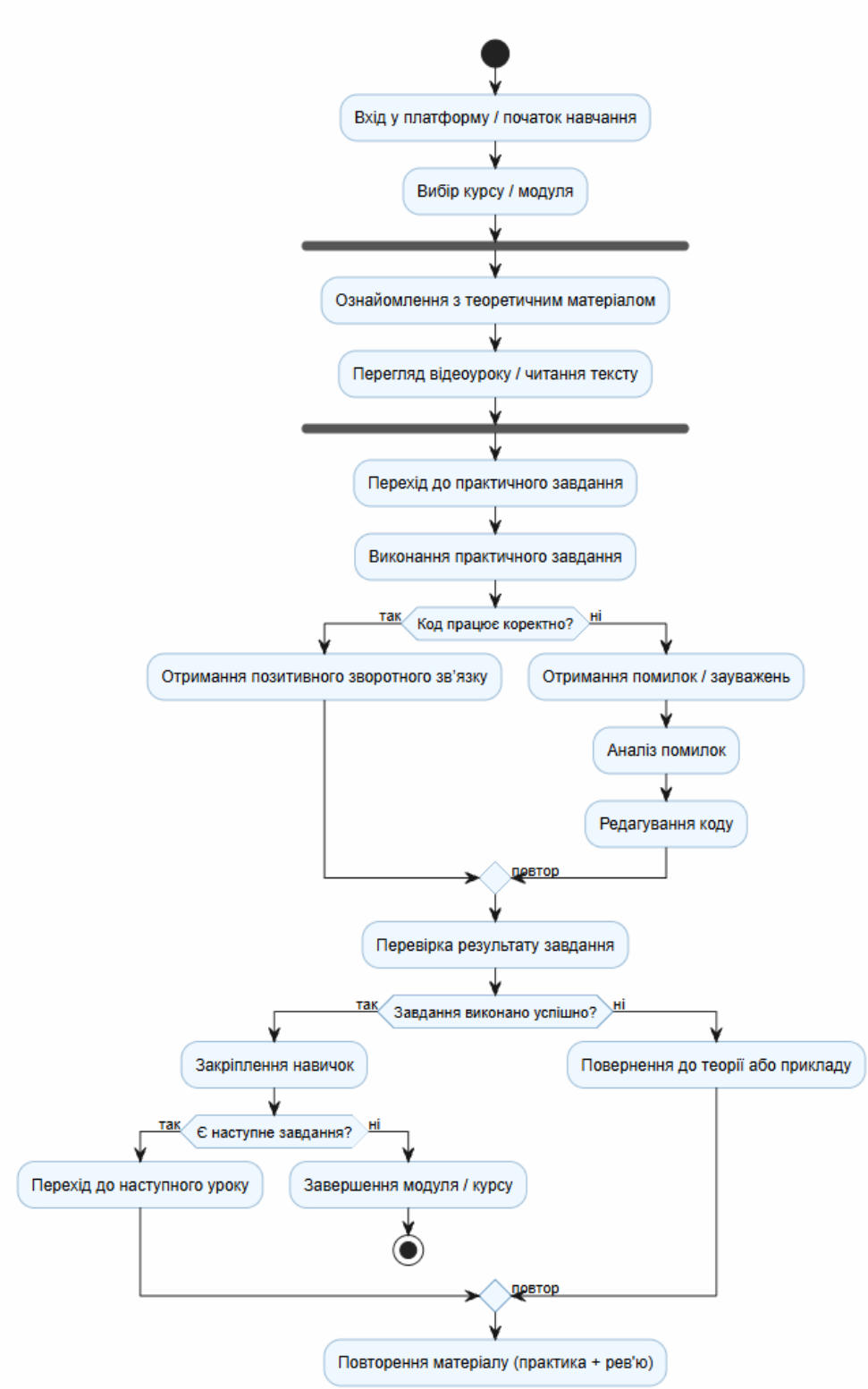

Рисунок 1.1 – Схема процесу вивчення програмування користувачем (UML блок-схема або Activity Diagram)

Основними труднощами при вивченні програмування є складність сприйняття абстрактних концепцій, недостатній рівень мотивації, обмежений

зворотний зв’язок у процесі навчання та перевантаження теоретичною інформацією. Дослідження показують, що значна частина студентів та учнів стикається з подібними проблемами, що часто призводить до втрати інтересу до подальшого навчання та відмови від вивчення програмування [3].

Рисунок 1.2 – Основні труднощі при вивченні програмування (за середнім балом) [3]

Як видно з рисунку 1.2, найбільш значущими проблемами є надмірна теоретична спрямованість навчальних курсів, недостатня практична складова та низький рівень мотивації. Отримані результати підтверджують необхідність використання сучасних підходів до організації навчального процесу, зокрема впровадження інтерактивних та гейміфікованих методів.

Одним із таких підходів є гейміфікація – використання ігрових елементів у неігрових процесах, зокрема в освіті. До таких елементів належать система балів, рівнів, нагород, прогресу та інтерактивних сценаріїв взаємодії з користувачем. Згідно з дослідженнями, гейміфікація позитивно впливає на мотивацію, залученість та тривалість навчальної активності користувачів [4].

У контексті розроблюваного вебзастосунку гейміфікація реалізується через систему навчальних завдань, прогресу користувача та віртуального персонажа, який супроводжує процес навчання.

1.1.2 Опис функціональної моделі

Функціональна модель вебзастосунку описує основні процеси взаємодії користувача із системою та визначає набір функцій, необхідних для реалізації навчального процесу. Основною метою розроблюваного вебзастосунку є створення інтерактивного середовища для вивчення мов програмування з використанням елементів гейміфікації, що сприяють підвищенню мотивації та залученості користувачів у процес навчання.

Сучасний підхід до навчання програмування передбачає поєднання коротких теоретичних блоків, практичних завдань та регулярного зворотного зв’язку. Така структура дозволяє користувачу поступово засвоювати матеріал і одразу застосовувати отримані знання на практиці. У розроблюваному вебзастосунку навчальний процес організований у вигляді невеликих уроків, що містять теоретичну частину, приклади коду та практичні вправи для самостійного виконання.

Особливістю системи є використання елементів гейміфікації, які реалізуються через систему прогресу, нарахування досвіду та візуалізацію досягнень користувача. У процесі проходження уроків користувач отримує зворотний зв’язок і може відстежувати власний прогрес у межах курсу, що робить навчання більш структурованим та наочним.

Окремим елементом системи є віртуальний персонаж, який виступає інтерактивним супутником користувача та відображає його навчальні досягнення у вигляді власного рівня розвитку. Прогрес проходження курсу трансформується у систему рівнів персонажа, що створює прямий зв’язок між виконанням навчальних завдань і його «еволюцією». Таким чином, загальний прогрес користувача в курсі відображається через систему візуалізації прогресу навчання, тоді як рівень персонажа виступає його ігровою інтерпретацією.

Функціональна модель системи включає такі основні можливості:

- реєстрація та авторизація користувача;
- вибір навчального курсу;

– проходження навчальних уроків;
– виконання практичних завдань;
– отримання автоматизованого зворотного зв’язку за результатами виконання завдань;
– перегляд прогресу навчання;
– взаємодія з віртуальним персонажем.

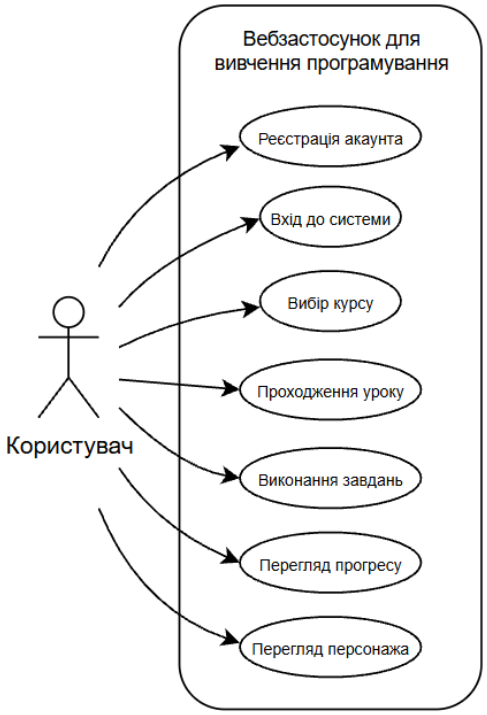

Рисунок 1.3 – Діаграма варіантів використання (Use Case Diagram) вебзастосунку для навчання програмування

Основні варіанти використання системи включають:

– реєстрація акаунта;
– вхід до системи;
– вибір курсу;
– проходження уроків;
– виконання завдань;
– перегляд прогресу;
– перегляд стану віртуального персонажа.

1.2 Огляд наявних аналогів

На сучасному етапі розвитку інформаційних технологій існує значна кількість вебзастосунків та онлайн-платформ, призначених для навчання програмування та суміжних дисциплін. Значна частина таких рішень орієнтована на дітей та підлітків і використовує елементи гейміфікації для підвищення мотивації до навчання.

Гейміфікація у навчальних системах передбачає використання ігрових механік, таких як система рівнів, бали досвіду, нагороди, персонажі та інтерактивні завдання. Такий підхід дозволяє зробити процес навчання більш залучаючим та зменшити втрату інтересу користувачів, що особливо актуально на початкових етапах вивчення програмування.

Водночас існуючі платформи суттєво відрізняються за рівнем інтерактивності, глибиною навчального контенту та ступенем гейміфікації. Умовно їх можна поділити на дві групи:

- класичні освітні платформи;
- гейміфіковані навчальні середовища.

1.2.1 Класичні освітні платформи

До цієї групи належать такі сервіси, як, наприклад, Coursera, Udemy та Prometheus.

Coursera та Udemy є масовими онлайн-платформами, що надають доступ до великої кількості курсів з програмування та інших IT-дисциплін. Навчання зазвичай побудоване у форматі відеолекцій із практичними завданнями. Основною перевагою таких систем є висока якість контенту та широкий вибір курсів.

Prometheus [5] є українською освітньою платформою, яка також пропонує безкоштовні та платні курси з програмування, алгоритмізації та суміжних напрямів. Особливістю є доступність навчання українською мовою та орієнтація на академічний формат подачі матеріалу.

Перевагами даної групи платформ є:

– структурованість навчальних курсів;
– висока якість навчальних матеріалів;
– широкий спектр тем.

Недоліками є:

– відсутність гейміфікації або її мінімальний рівень;
– низький рівень інтерактивності;
– слабка емоційна залученість користувача;
– високий рівень відсіву студентів на початкових етапах навчання.

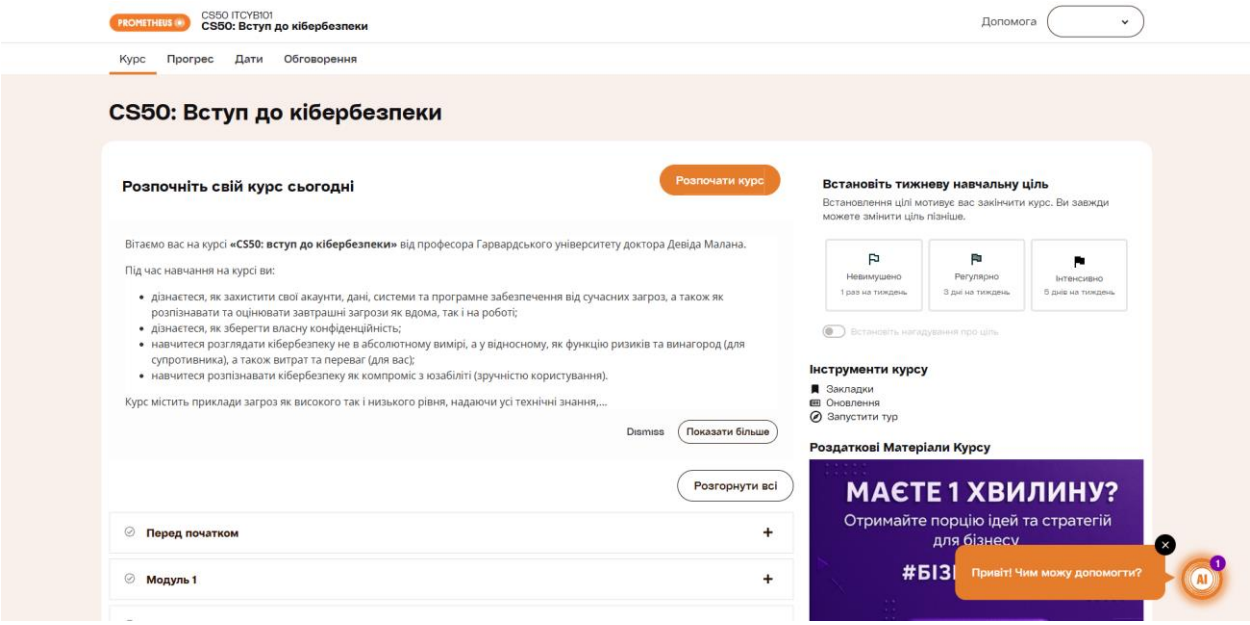

Рисунок 1.4 – Приклад сторінки курсу на Prometheus

1.2.2 Гейміфіковані навчальні платформи

До цієї категорії належать системи, що поєднують навчання з ігровими механіками.

Гейміфіковане навчання (Duolingo)

Duolingo [6] – популярна онлайн-платформа для вивчення іноземних мов, яка активно використовує елементи гейміфікації для підвищення залученості користувачів. Незважаючи на те, що платформа не орієнтована на

навчання програмуванню, вона є показовим прикладом ефективного застосування ігрових механік у навчальному процесі.

Навчання у Duolingo організоване у вигляді коротких інтерактивних уроків, які користувач проходить поступово. Система включає такі елементи, як бали досвіду, рівні, щоденні цілі, серії безперервного навчання (streak) та віртуальні персонажі.

Платформа забезпечує високий рівень залученості користувача завдяки поєднанню простих завдань, миттєвого зворотного зв’язку та візуального відображення прогресу.

Переваги:

- високий рівень мотивації користувачів;
- ефективна система прогресу та винагород;
- простий і зрозумілий інтерфейс;
- короткі інтерактивні уроки.

Недоліки:

- обмежена глибина опрацювання складних тем;
- спрощений формат завдань;
- елементи гейміфікації можуть створювати відчуття зобов’язаності (наприклад, підтримка серії занять – streak), що інколи викликає психологічний тиск;
- надмірна кількість нагадувань та повідомлень може сприйматися користувачами як нав’язлива взаємодія;
- відсутність орієнтації на навчання програмуванню.

Візуальне програмування (Scratch / Tynker)

Scratch та Tynker [7] орієнтовані на молодшу аудиторію та використовують візуальне (блочне) програмування. Користувач створює програми шляхом з’єднання блоків, що дозволяє легко засвоїти базові алгоритмічні конструкції.

Tynker додатково пропонує ігрові сценарії та персонажів, тоді як Scratch більше орієнтований на творчість і створення власних проєктів.

Переваги:

– простий старт навчання;
– візуальне представлення алгоритмів;
– розвиток творчих навичок;
– орієнтація на дітей.

Недоліки:

– відсутність або слабка підтримка текстового програмування на старті;
– обмежений рівень складності;
– недостатня система довготривалої мотивації.

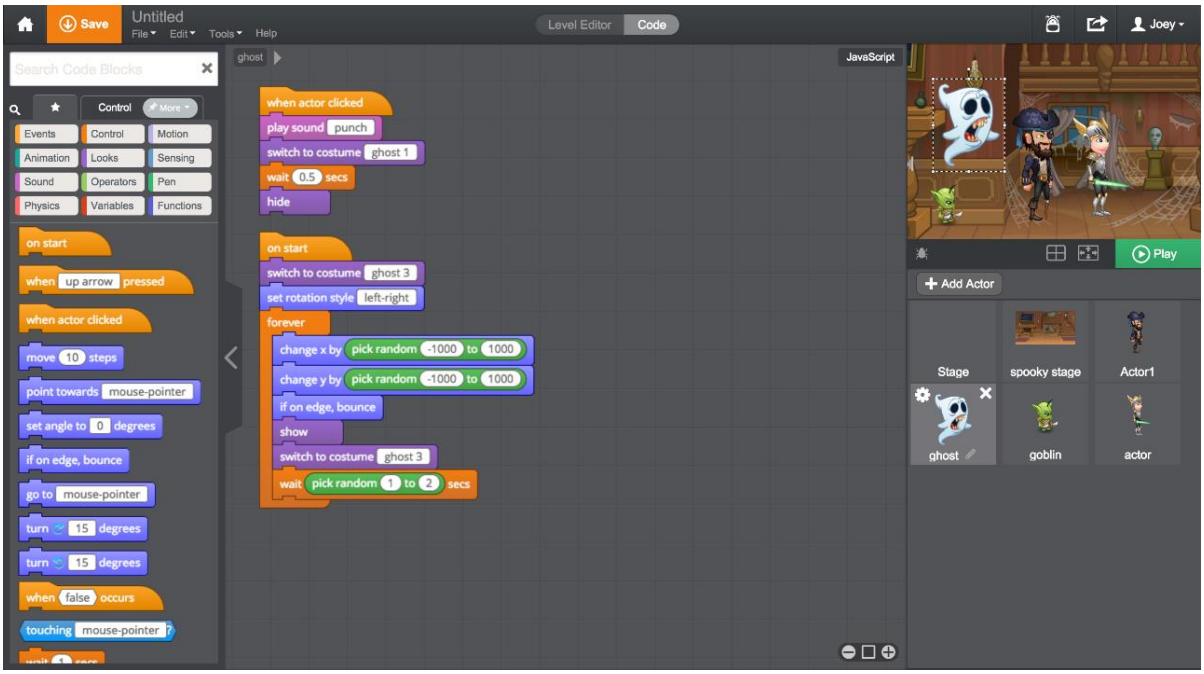

Рисунок 1.5 – Приклад створення блокового коду в Tynker [8]

Ігрове навчання (CodeCombat / CodeMonkey / ProdigyGame)

CodeCombat та CodeMonkey реалізують навчання у форматі гри, де користувач керує персонажем за допомогою коду. Основна увага приділяється алгоритмам, логіці та базовим конструкціям програмування.

ProdigyGame (Prodigy Math) – навчальна RPG-гра, орієнтована на математику для дітей молодшого та середнього шкільного віку. Демонструє

ефективний підхід до практики математичних навичок через ігрові механіки, систему прогресу та бойові завдання.

Переваги:

– висока залученість через ігровий процес;
– навчання через практичні задачі;
– розвиток алгоритмічного мислення;
– наявність системи прогресу.

Недоліки:

– іноді надмірна складність для початківців;
– обмежена гнучкість у роботі з реальним кодом;
– часткова платність функціоналу.

Рисунок 1.6 – Приклад ігрового рівня CodeCombat з кодом

Для більш наочного порівняння розглянутих платформ основні характеристики було узагальнено та систематизовано у вигляді таблиці 1.1. У таблиці представлено порівняння платформ за такими критеріями, як тип навчання, використовувані мови програмування, рівень гейміфікації, ключові особливості та основні недоліки.

Таблиця 1.1 – Порівняння платформ-аналогів

Платформа	Тип навчання	Мови	Гейміфікація	Основна особливість	Недоліки
Coursera	Академічні курси	Python, JS тощо	Низька	Висока якість курсів	Мало інтерактивності
Udemy	Відеокурси	Різні	Низька	Велика кількість курсів	Немає структури прогресу
Prometheus	Освітні курси	Python, ін.	Низька	Український контент	Слабка інтерактивність
Duolingo	Інтерактивне навчання	–	Висока	Сильна гейміфікація та мотивація	Не орієнтований на програмування
Scratch / Tynker	Візуальне програмування	Візуальна мова	Середня	Простота навчання	Обмежений текстовий код
CodeCombat / CodeMonkey	Ігрове програмування	Python, JS	Висока	Навчання через гру	Складність для новачків
ProdigyGame	Освітня RPG	–	Висока	Сильна гейміфікація	Не орієнтований на програмування

Аналіз даних, наведених у таблиці 1.1, дозволяє зробити висновок, що жодна з розглянутих платформ не забезпечує повного поєднання простоти навчання, інтерактивності та достатнього рівня гейміфікації.

Класичні освітні платформи, такі як Coursera, Udemy та Prometheus, забезпечують якісний теоретичний матеріал, проте мають низький рівень залучення користувача. Водночас гейміфіковані платформи, такі як Duolingo, CodeCombat, CodeMonkey та ProdigyGame, демонструють високий рівень залученості користувачів, проте або не орієнтовані на навчання програмуванню, або можуть бути складними для початківців.

Платформи візуального програмування (Scratch, Tynker) забезпечують простий вхід у програмування, проте мають обмеження у подальшому переході до текстових мов.

Таким чином, результати порівняльного аналізу підтверджують доцільність розробки вебзастосунку, який поєднуватиме простоту навчання, інтерактивність, систему прогресу та елементи гейміфікації, зокрема

використання віртуального персонажа як засобу підвищення мотивації користувача.

1.3 Постановка задачі

Метою розробки вебзастосунку є створення інтерактивного середовища для навчання основ програмування на мові Python із використанням елементів гейміфікації. Система спрямована на поєднання навчального контенту з ігровими механіками, що забезпечує підвищення мотивації та залученості користувачів у процесі навчання.

У межах реалізації передбачається вирішення проблеми низької мотивації та швидкої втрати інтересу під час вивчення програмування, що є характерним для початкового етапу навчання.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати існуючі підходи до навчання програмування та гейміфіковані платформи;
- визначити структуру навчального курсу з основ Python;
- розробити архітектуру вебзастосунку;
- реалізувати базовий функціонал системи, що включає навчальні уроки, практичні завдання та перевірку відповідей;
- реалізувати систему відстеження прогресу користувача;
- розробити елемент гейміфікації у вигляді віртуального персонажа, який супроводжує процес навчання;
- забезпечити базову інтерактивність та зручність користувацького інтерфейсу.

Для реалізації вебзастосунку передбачається використання мови програмування Python для серверної частини. Для побудови інтерфейсу користувача застосовуються стандартні вебтехнології, зокрема HTML, CSS та JavaScript.

Оскільки проєкт має характер прототипу, вибір конкретних технологічних рішень може уточнюватися в процесі розробки. Основна увага зосереджується на реалізації навчальної логіки та гейміфікованої взаємодії, а не на побудові повноцінного комерційного продукту.

Наукова новизна роботи полягає у поєднанні навчального процесу з елементами гейміфікації через використання віртуального персонажа як інтерактивного засобу мотивації користувача. Персонаж виконує функцію супровідного елементу навчання, відображаючи прогрес користувача та змінюючи свій стан залежно від рівня засвоєння навчального матеріалу. Це створює емоційний зв’язок між користувачем і системою та підсилює залученість у процес навчання. На відміну від існуючих рішень, акцент зроблено на інтеграції навчальної взаємодії з емоційно-мотиваційними механіками, що забезпечує більш глибоку персоналізацію та підвищення ефективності навчання.

Результатом роботи є функціональний прототип вебзастосунку для вивчення основ програмування, який поєднує навчальні матеріали, систему прогресу та елементи гейміфікації, спрямовані на підвищення ефективності засвоєння знань.

Висновки до розділу

У розділі було обґрунтовано актуальність розробки вебзастосунку для вивчення основ програмування з використанням елементів гейміфікації для дітей та підлітків (початківців). Встановлено, що традиційні підходи до навчання та більшість наявних освітніх платформ не забезпечують достатнього рівня мотивації та залученості користувачів через складність абстрактного матеріалу, недостатню інтерактивність та швидку втрату інтересу на початкових етапах.

Проведено аналіз предметної області, описано процес навчання програмування та визначено основні фактори, що впливають на зниження

ефективності навчального процесу. Окрему увагу приділено ролі гейміфікації як сучасного підходу до підвищення мотивації в освітніх системах.

Аналіз існуючих аналогів показав, що більшість платформ або недостатньо орієнтовані на інтерактивне текстове програмування, або не забезпечують повноцінного поєднання навчального процесу з ігровими механіками та емоційною взаємодією користувача.

Розглянуто функціональну модель вебзастосунку, яка включає навчальні модулі з практичними завданнями, систему прогресу та віртуального персонажа як мотиваційний елемент.

На основі проведеного аналізу сформульовано мету, завдання та постановку задачі, що створює основу для подальшого проєктування архітектури та реалізації функціонального прототипу вебзастосунку.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз предметної області

У сучасних умовах розвитку інформаційних технологій та цифровізації освіти особливої актуальності набуває питання ефективного навчання програмування, зокрема на початковому рівні. Вивчення мов програмування часто супроводжується труднощами, пов’язаними з абстрактністю матеріалу, відсутністю наочності та недостатнім рівнем залученості користувачів у навчальний процес.

Предметна область даної роботи охоплює процес організації навчання основ програмування у вебсередовищі, що передбачає взаємодію користувача з навчальним контентом, виконання практичних завдань та отримання результатів навчальної діяльності. Важливою складовою є поєднання теоретичного матеріалу з практичним застосуванням знань, що дозволяє формувати базові навички програмування.

Аналіз предметної області показує, що традиційні підходи до навчання програмування, такі як підручники, відеолекції або окремі практичні середовища, не завжди забезпечують достатній рівень інтерактивності та мотивації. Користувачі, особливо початківці, часто втрачають інтерес через складність матеріалу або відсутність швидкого зворотного зв’язку щодо результатів виконання завдань.

У межах предметної області можна виділити такі основні складові:

– користувач, який взаємодіє з системою та проходить навчальні матеріали;

– навчальний курс, що містить структуровані уроки з певної тематики;

– уроки, які включають теоретичну частину та практичні завдання;

– результати виконання завдань, що відображають рівень засвоєння матеріалу;

– прогрес навчання, який характеризує ступінь проходження курсу.

Окрему увагу в межах предметної області приділено використанню елементів гейміфікації як засобу підвищення зацікавленості користувача. Гейміфікація передбачає використання ігрових механік у неігровому середовищі, що дозволяє зробити процес навчання більш привабливим та емоційно насиченим. До таких механік належать система прогресу, візуалізація досягнень та інтерактивні елементи, що супроводжують користувача під час навчання.

Таким чином, аналіз предметної області демонструє необхідність створення інтерактивного вебзастосунку, що поєднує навчальні матеріали, практичні завдання й елементи мотивації користувача. Це сприяє підвищенню ефективності засвоєння знань і робить процес навчання більш цікавим, зрозумілим та структурованим.

2.1.1 Вхідні дані

Для забезпечення роботи вебзастосунку для вивчення мов програмування необхідне надходження різних категорій вхідних даних. Основним джерелом таких даних є користувач, який взаємодіє із системою під час реєстрації, проходження курсів та виконання практичних завдань.

До основних категорій вхідних даних належать:

– дані облікового запису користувача;

– навчальні дані курсів;

– дані практичних завдань;

– дані взаємодії з гейміфікаційною системою.

Під час реєстрації та авторизації користувач вводить персональні дані, необхідні для ідентифікації в системі. До них належать ім’я користувача, адреса електронної пошти та пароль.

Навчальні дані містять структуру курсів, модулів і уроків. Для кожного уроку задаються назва, тип навчального матеріалу (теорія або практичне завдання), опис завдання та початкові дані для виконання вправ.

Під час виконання практичних завдань користувач вводить програмний код, а також за потреби додаткові текстові значення, що використовуються у навчальних прикладах.

Окрему категорію становлять дані взаємодії з елементами гейміфікації, зокрема дії користувача щодо віртуального персонажа (наприклад, взаємодія, годування або інші доступні дії).

Структуру основних вхідних даних вебзастосунку наведено на рисунку 2.1.

Рисунок 2.1 – Структура вхідних даних системи

Основні категорії вхідних даних наведено в таблиці 2.1.

Таблиця 2.1 – Основні вхідні дані системи

Категорія даних	Призначення
Дані користувача	Реєстрація та авторизація
Навчальні дані	Формування структури курсів
Програмний код користувача	Виконання практичних завдань
Дані взаємодії	Робота з елементами гейміфікації

На відміну від вхідних даних, результати обробки інформації системою, зокрема прогрес навчання, нараховані бали досвіду та стан віртуального персонажа, належать до вихідних даних і розглядаються в підрозділі 2.1.2.

2.1.1 Вихідні дані

У результаті обробки вхідних даних система формує вихідну інформацію, яка відображає перебіг навчального процесу, результати діяльності користувача та його досягнення на платформі. Вихідні дані призначені для інформування користувача про стан проходження курсу, успішність виконання завдань і накопичений прогрес.

Основними категоріями вихідних даних є результати навчання, інформація про проходження курсу, результати виконання практичних завдань та елементи гейміфікації.

Після взаємодії з навчальними матеріалами система формує інформацію про поточний стан проходження курсу. Користувач отримує відомості про завершені уроки, доступні навчальні модулі та загальний відсоток опанування курсу. Це дозволяє контролювати власний прогрес та планувати подальше навчання.

Під час виконання практичних завдань система надає результат перевірки введеного розв’язку. Користувач отримує повідомлення про успішне виконання завдання або інформацію про виявлені помилки, що сприяє закріпленню отриманих знань і формуванню навичок програмування.

Важливою складовою вихідних даних є статистика навчальних досягнень. Система відображає рівень активності користувача, накопичений досвід, досягнутий рівень та інші показники, що характеризують успішність проходження курсу.

Для підвищення мотивації до навчання у платформі використовується гейміфікаційна складова. У результаті дій користувача система формує інформацію про стан віртуального персонажа, його розвиток та зміни характеристик. Такий підхід забезпечує додаткову зацікавленість у регулярному проходженні навчальних матеріалів.

Структуру основних вихідних даних вебплатформи наведено на рисунку 2.2.

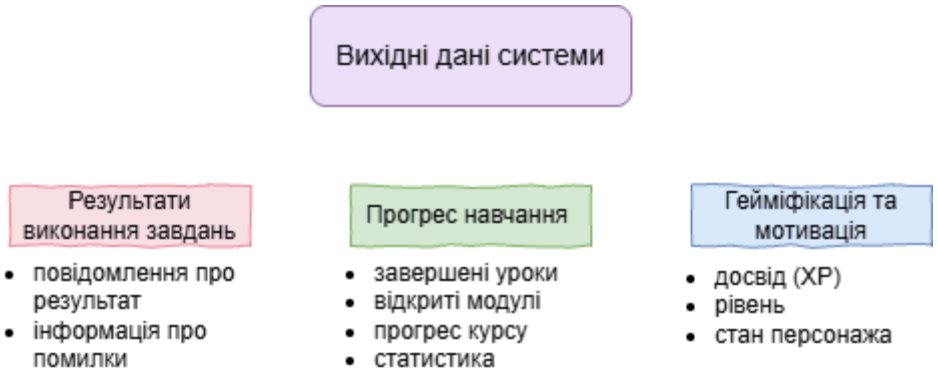

Рисунок 2.2 – Структура вихідних даних

Результатом роботи системи є надання користувачеві актуальної інформації про власні досягнення, успішність виконання завдань, прогрес проходження курсу та стан елементів гейміфікації, що забезпечує зручний контроль навчального процесу.

2.2 Проєктування системи

Проєктування системи є одним із ключових етапів розроблення вебплатформи для вивчення мов програмування. На цьому етапі визначаються основні компоненти системи, їх взаємозв’язки та принципи організації даних, що забезпечують подальшу реалізацію функціональних можливостей застосунку.

Основною метою проєктування є розробка цілісної, логічної та масштабовної структури системи, яка забезпечуватиме зручну взаємодію користувача з навчальним контентом, ефективне відстеження прогресу та реалізацію елементів гейміфікації. Особлива увага приділяється інтеграції навчальної складової з ігровою механікою, щоб віртуальний персонаж, система рівнів та прогрес навчання працювали як єдине ціле.

Оскільки розроблювана система належить до категорії навчальних вебзастосунків, під час проєктування особлива увага приділяється організації навчального контенту, структурі користувацьких даних та механізмам відстеження результатів навчання. Крім того, необхідно передбачити можливість подальшого розширення функціоналу шляхом додавання нових курсів, навчальних модулів та ігрових елементів.

У межах даного підрозділу розглядаються концептуальна модель зберігання даних та побудова об’єктно-орієнтованої моделі системи, які є основою для подальшої програмної реалізації вебплатформи.

2.2.1 Проєктування бази даних

Для забезпечення ефективної роботи навчальної вебплатформи необхідно організувати зберігання інформації про користувачів, навчальні курси, результати навчання та елементи гейміфікації. Незважаючи на те, що на поточному етапі розробки проєкт функціонує виключно на клієнтській стороні без використання окремої серверної бази даних, доцільно сформувати повноцінну логічну та концептуальну модель даних. Такий підхід дозволяє чітко визначити структуру інформації, встановити взаємозв’язки між основними сутностями системи, а також забезпечити можливість подальшого масштабування та спрощений перехід до серверної архітектури в майбутньому.

Основу інформаційної моделі складають шість основних сутностей: Користувач (User), Курс (Course), Модуль (Module), Урок (Lesson), Прогрес (Progress) та Віртуальний персонаж-звір (Pet). Взаємозв’язки між сутностями наведено на рисунку 2.3.

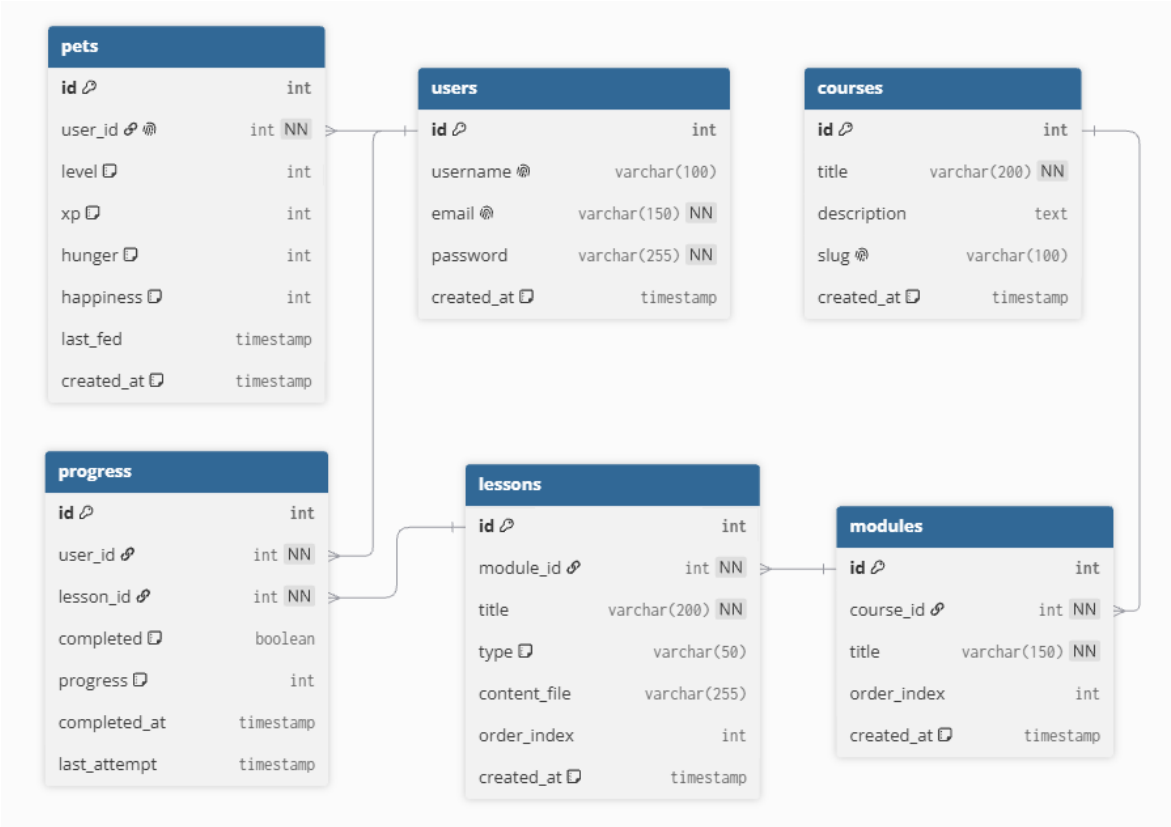

Рисунок 2.3 – Концептуальна модель даних навчальної платформи (ER-діаграма)

Сутність «Користувач» призначена для зберігання відомостей про зареєстрованих користувачів системи. До основних атрибутів належать:

– ідентифікатор користувача;
– ім’я користувача;
– адреса електронної пошти;
– пароль для автентифікації.

Кожен користувач може проходити один або декілька навчальних курсів. Для збереження інформації про прогрес навчання використовується проміжна сутність «Progress», яка реалізує зв’язок типу «багато-до-багатьох» між сутностями «User» та «Lesson» (або «Module»).

Сутність «Курс» містить інформацію про навчальні матеріали певного напряму. До її складу входять назва курсу, короткий опис та набір навчальних модулів.

Кожен курс складається з кількох модулів, тому між сутностями «Course» та «Module» встановлюється зв’язок типу «один-до-багатьох».

Сутність «Модуль» використовується для логічного групування навчального матеріалу. Кожен модуль містить набір уроків. Між сутностями Module та Lesson встановлюється зв’язок типу «один-до-багатьох».

Сутність «Урок» описує окремий елемент навчального контенту (теорія, практичне завдання тощо) та містить назву, тип, зміст і стан проходження.

Сутність «Прогрес» (Progress) зберігає дані про виконання користувачем конкретних уроків: відсоток виконання, статус завершення, дату проходження тощо.

Сутність «Віртуальний персонаж» використовується для реалізації елементів гейміфікації та містить показники розвитку персонажа (рівень, досвід, інші характеристики), які змінюються залежно від активності користувача. Між сутностями User та Pet встановлюється зв’язок типу «один-до-одного».

Запропонована структура даних є достатньо гнучкою та забезпечує зберігання всієї необхідної інформації для функціонування навчальної платформи, дозволяє ефективно відстежувати прогрес користувачів і підтримувати гейміфікацію, а також може бути легко розширена під час подальшого розвитку проєкту, зокрема шляхом додавання таблиць для тестів, сертифікатів, досягнень або підтримки кількох курсів одночасно.

2.2.2 Побудова об’єктно-орієнтованої моделі

Для спрощення розробки, підтримки та можливого майбутнього масштабування вебзастосунку було обрано об’єктно-орієнтований підхід до проєктування. Такий підхід дозволяє представити систему у вигляді взаємопов’язаних класів, кожен з яких відповідає за певну область функціональності.

Під час побудови об'єктно-орієнтованої моделі були визначені основні сутності системи, які відображають структуру навчальної платформи та логіку взаємодії користувача з навчальним контентом. До таких сутностей належать:

User, Course, Module, Lesson, Progress та Character. Взаємозв'язки між ними представлені на UML-діаграмі класів (рис. 2.4).

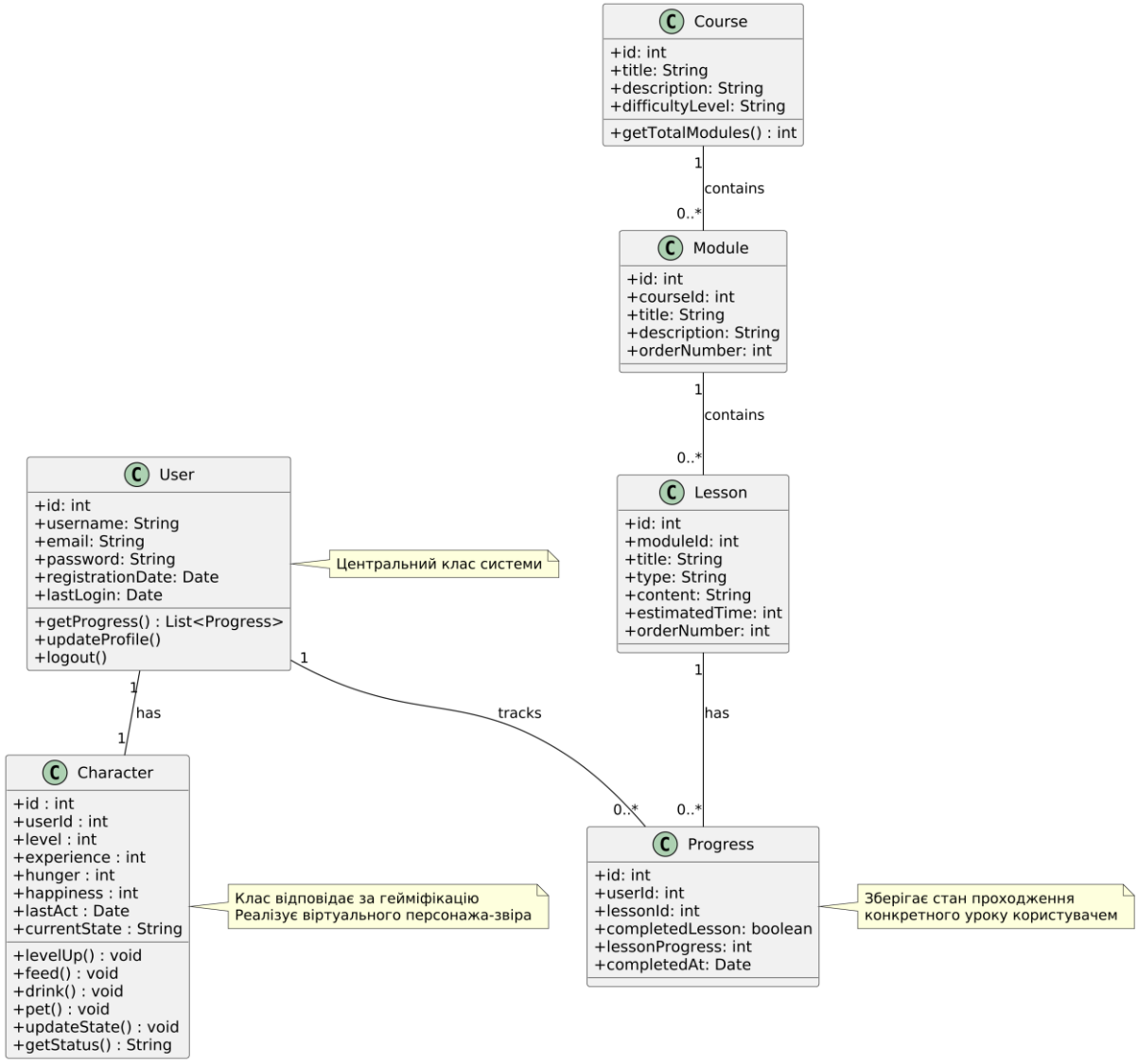

Рисунок 2.4 – UML-діаграма класів вебзастосунку

Центральним елементом системи є клас User, який представляє зареєстрованого користувача платформи. Даний клас використовується для зберігання інформації про обліковий запис та індивідуальний прогрес навчання.

Основними атрибутами класу є:

– id – унікальний ідентифікатор користувача;
– username – ім'я користувача;

- email – адреса електронної пошти;
- password – пароль доступу до системи;
- registrationDate – дата реєстрації;
- lastLogin – дата останнього входу.

Окрім атрибутів, клас User містить методи для взаємодії з даними користувача. Метод getProgress() використовується для отримання інформації про поточний стан проходження курсу. Метод updateProfile() призначений для оновлення особистих даних користувача. Метод logout() забезпечує завершення поточної сесії та вихід користувача із системи.

Оскільки кожен користувач проходить навчальний курс у власному темпі, об'єкт User пов'язаний із результатами проходження модулів, уроків та тестів.

Основною навчальною сутністю системи є клас Course, який представляє повний курс програмування.

Для дипломного проєкту передбачено реалізацію курсу з мови програмування Python. Надалі структура системи допускає додавання нових курсів без зміни загальної архітектури застосунку.

Клас Course містить такі атрибути:

- id – ідентифікатор курсу;
- title – назва курсу;
- description – короткий опис;
- difficultyLevel – рівень складності.

Крім того, клас містить метод getTotalModules(), який використовується для отримання кількості модулів курсу та повертає ціле числове значення.

Кожен курс складається з декількох навчальних модулів, тому між класами Course та Module існує зв'язок типу «один до багатьох».

Для структуризації навчального матеріалу використовується клас Module.

Модуль об'єднує декілька взаємопов'язаних тем та дозволяє логічно розділити навчальний курс на окремі етапи.

Атрибутами класу є:

- id – ідентифікатор модуля;
- courseId – посилання на курс;
- title – назва модуля;
- description – опис змісту;
- orderNumber – порядковий номер у курсі.

Кожен модуль містить набір уроків, тому між класами Module та Lesson реалізується зв'язок типу «один до багатьох».

Безпосередньо навчальний матеріал представлений класом Lesson.

Саме урок є основною одиницею навчального контенту, яку опрацьовує користувач під час проходження курсу.

До атрибутів Lesson належать:

- id – ідентифікатор уроку;
- moduleId – посилання на модуль;
- title – назва уроку;
- type – тип уроку (theory, code);
- content – навчальний матеріал;
- estimatedTime – орієнтовний час проходження;
- orderNumber – порядковий номер.

Для відстеження прогресу користувача введено клас Progress, який пов’язує користувача з конкретними уроками.

Основним призначенням цього класу є збереження інформації про стан проходження навчального матеріалу та забезпечення можливості відновлення навчання з місця останнього завершеного етапу. Кожен запис класу Progress відповідає одному уроку, який проходить конкретний користувач.

До атрибутів класу належать:

– id – унікальний ідентифікатор запису прогресу;

– userId – посилання на користувача;

– lessonId – посилання на урок;

– completedLesson – ознака завершення уроку;

– lessonProgress – відсоток виконання уроку;

– completedAt – дата та час завершення уроку.

Клас Progress забезпечує можливість визначення рівня засвоєння навчального матеріалу, контролю проходження курсу та формування статистики навчальної активності користувача.

Між класами User та Progress реалізується зв’язок типу «один до багатьох», оскільки один користувач може мати записи прогресу для багатьох уроків. Водночас кожен урок також може бути пов’язаний із великою кількістю записів Progress, що відповідають різним користувачам платформи.

Для візуального відображення прогресу користувача у системі передбачено клас Character.

Клас Character реалізує елементи гейміфікації платформи та представляє віртуального персонажа-звіра, який супроводжує користувача протягом навчання та змінює свій рівень залежно від накопиченого досвіду.

Основними атрибутами Character є:

- id – ідентифікатор персонажа;
- userId – ідентифікатор користувача;
- level – поточний рівень;
- experience – кількість накопиченого досвіду (XP);
- hunger – показник голоду персонажа (від 0 до 100);
- happiness – показник щастя/задоволення персонажа (від 0 до 100);
- lastAct – дата і час останньої взаємодії з персонажем;
- currentState – поточний стан або стадія розвитку персонажа (наприклад, «egg», «baby», «teen», «adult»).

Клас також містить методи, які забезпечують взаємодію з персонажем:

- levelUp() – підвищення рівня персонажа при накопиченні достатньої кількості досвіду;
- feed() – нагодувати персонажа: впливає на показники hunger та happiness;

- drink() – напоїти персонажа: зменшує показник hunger та підтримує нормальний стан персонажа (happiness);
- pet() – погладити персонажа: значно підвищує показник щастя (happiness);
- updateState() – оновлення візуального стану персонажа залежно від його характеристик.
- getStatus() – отримання поточного стану персонажа у вигляді текстової інформації.

Персонаж пов'язаний із користувачем зв'язком типу «один до одного», оскільки кожному користувачу відповідає власний віртуальний супутник.

Використання такого підходу дозволяє реалізувати механізми гейміфікації, що позитивно впливають на залучення користувачів до навчального процесу.

Таким чином, побудована об'єктно-орієнтована модель охоплює основні сутності навчальної платформи, їх атрибути та взаємозв'язки. Використання об'єктно-орієнтованого підходу дає можливість забезпечити модульність архітектури, спростити подальшу реалізацію програмного забезпечення та створити основу для розширення функціональних можливостей системи в майбутньому.

2.3 Математичне та алгоритмічне забезпечення

2.3.1 Математичне забезпечення

Математичне забезпечення вебзастосунку використовується для обчислення навчального прогресу користувача, визначення рівня віртуального персонажа та формування статистичної інформації, що відображається в особистому кабінеті.

Основним показником успішності проходження курсу є відсоток виконання навчального матеріалу. Значення прогресу визначається як відношення кількості завершених уроків до загальної кількості уроків курсу.

Формула розрахунку прогресу має вигляд:

$$P = \left(\frac{L_c}{L_t}\right) \cdot 100, \tag{2.1}$$

де

- P – відсоток проходження курсу, %;
- L_c – кількість завершених уроків;
- L_t – загальна кількість уроків курсу.

Значення прогресу використовується для побудови індикатора виконання курсу та відображення статистики навчання в профілі користувача.

Для визначення загальної кількості завершених уроків здійснюється послідовний перегляд усіх модулів курсу та перевірка стану кожного уроку. Кількість завершених уроків обчислюється як сума всіх уроків, для яких встановлено ознаку завершення.

Формула має вигляд:

$$P = \left(\frac{\sum_{i=1}^{n} l_i}{n}\right), \tag{2.2}$$

де:

- l_i – ознака завершення уроку ($l_i = 1$, якщо урок завершений, та $l_i = 0$, якщо ні);
- n – загальна кількість уроків.

Окремим елементом системи є механізм визначення рівня віртуального персонажа. У реалізованій версії застосунку рівень персонажа безпосередньо залежить від відсотка проходження курсу Python.

Рівень обчислюється за формулою:

де:

$$Level = max\left(1, \left\lfloor \frac{P}{2.5} \right\rfloor\right), \tag{2.3}$$

- *Level* – рівень персонажа;

- P – прогрес проходження курсу у відсотках.

Використання даної формули дозволяє поступово збільшувати рівень персонажа відповідно до успішності навчання користувача. При повному проходженні курсу максимальний рівень становить 40.

Крім основних показників, у системі виконуються допоміжні обчислення, необхідні для формування статистичної інформації про проходження курсу. Зокрема, визначається кількість завершених уроків шляхом підрахунку навчальних занять, для яких встановлено ознаку завершення. Також обчислюється загальна кількість уроків курсу та кількість навчальних модулів, що використовуються під час розрахунку відсотка прогресу користувача.

Кількість завершених уроків визначається як сума всіх уроків, що мають статус завершених. Загальна кількість уроків обчислюється шляхом підрахунку всіх уроків, які входять до складу навчальних модулів курсу. Кількість модулів визначається як число навчальних блоків, передбачених структурою курсу.

Зазначені обчислення не є самостійними математичними моделями, однак виступають вхідними даними для розрахунку прогресу навчання та рівня віртуального персонажа. Вони забезпечують коректне відображення статистики користувача та дозволяють оцінювати стан проходження навчального матеріалу.

Отримані математичні показники використовуються для формування прогрес-барів, статистичних блоків профілю користувача та визначення стану розвитку віртуального персонажа.

2.3.2 Алгоритмічне забезпечення

Алгоритмічне забезпечення вебзастосунку визначає порядок виконання основних функцій системи та забезпечує взаємодію користувача з навчальною платформою. Реалізовані алгоритми відповідають за реєстрацію та авторизацію користувачів, проходження навчальних курсів, обробку прогресу

навчання, роботу особистого профілю та функціонування елементів гейміфікації.

Одним із базових алгоритмів системи є алгоритм реєстрації користувача. Після введення користувачем імені, електронної пошти та пароля система виконує перевірку правильності заповнення полів. Далі здійснюється пошук користувача з аналогічним логіном або електронною адресою серед записів, що зберігаються у Local Storage. Якщо користувача не знайдено, створюється новий обліковий запис із унікальним ідентифікатором та початковими параметрами профілю. Після успішного створення облікового запису інформація зберігається у локальному сховищі браузера, а користувач автоматично авторизується в системі.

У випадку входу до системи використовується алгоритм авторизації. Користувач вводить логін або електронну адресу та пароль, після чого система здійснює пошук відповідного запису серед зареєстрованих користувачів. Якщо введені дані збігаються із збереженими значеннями, виконується авторизація та відкривається доступ до функціональних можливостей платформи. У разі невідповідності даних користувачу виводиться повідомлення про помилку.

Алгоритм реєстрації та авторизації користувача можна подати у вигляді UML-діаграми активності (рис. 2.5).

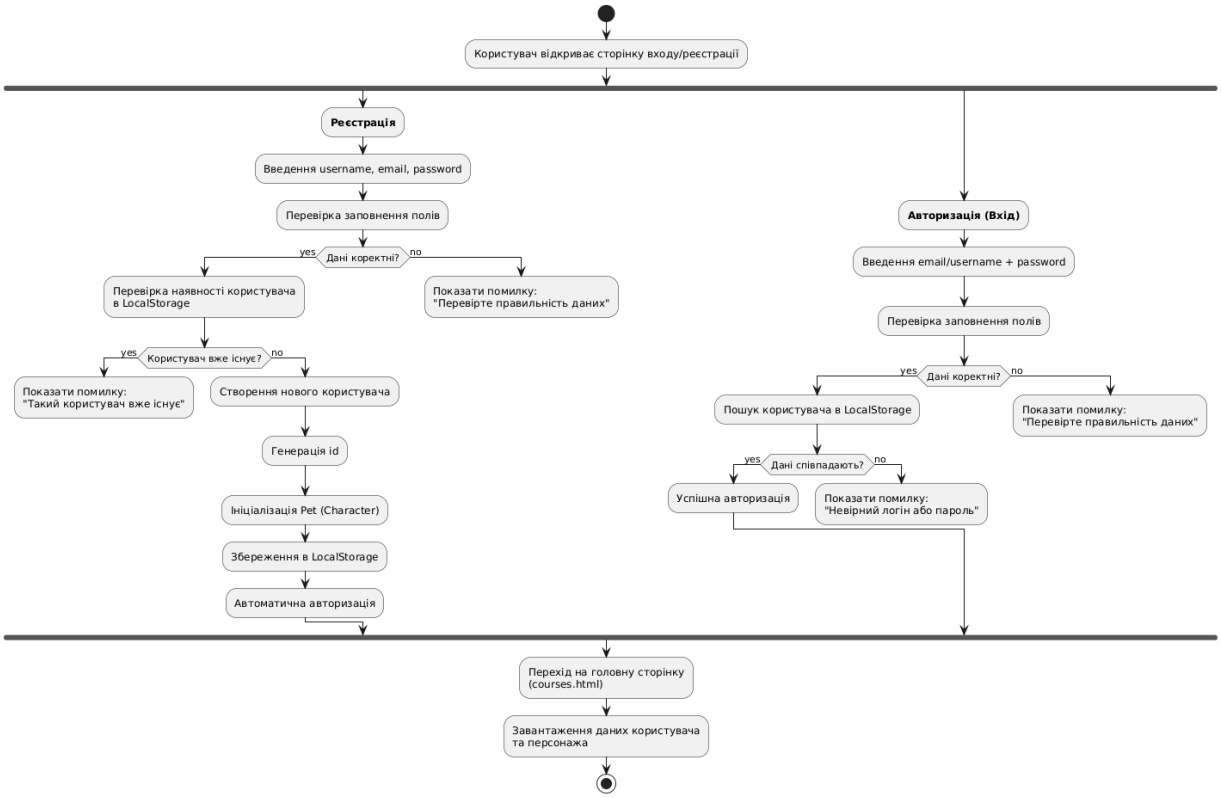

Рисунок 2.5 – UML-діаграма алгоритму реєстрації та авторизації користувача

Після входу до системи користувач отримує доступ до навчальних курсів. Навчальний контент організований у вигляді курсу «Основи Python», який складається з восьми навчальних модулів. Кожен модуль містить теоретичні уроки та практичні завдання з програмування.

Під час відкриття курсу система виконує завантаження структури навчальних модулів із JSON-файлу та формує список доступних уроків. Після вибору уроку користувач ознайомлюється з теоретичним матеріалом або виконує практичне завдання. Після завершення заняття система змінює статус уроку на виконаний та зберігає результат у Local Storage.

Алгоритм проходження навчального уроку включає такі етапи:

1. Вибір курсу користувачем.
2. Завантаження структури модулів.
3. Вибір навчального уроку.
4. Перегляд теоретичного матеріалу або виконання практичного завдання.
5. Позначення уроку як завершеного.

6. Збереження результату навчання.
7. Оновлення прогресу курсу.
8. Перехід до наступного уроку.

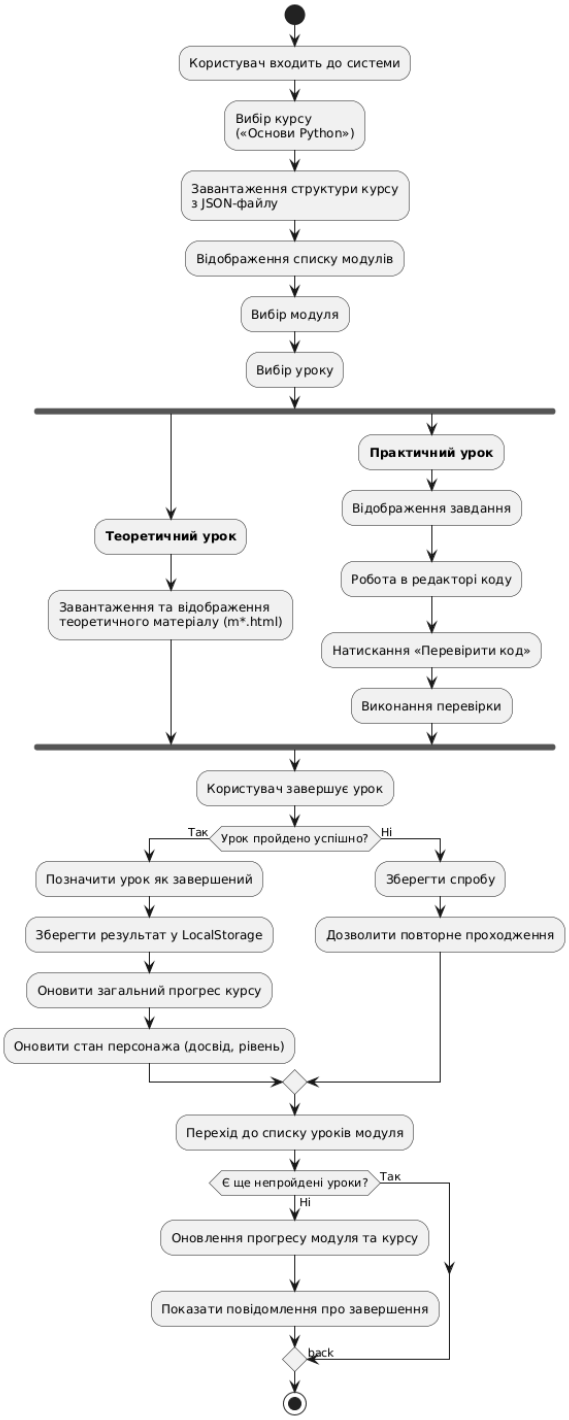

Рисунок 2.6 – UML-діаграма алгоритму проходження навчального уроку

Для контролю результатів навчання використовується алгоритм обробки прогресу. Після завершення уроку система зчитує всі модулі курсу та виконує підрахунок загальної кількості уроків і кількості завершених занять. Отримані значення використовуються для обчислення відсотка проходження курсу. Результат зберігається та відображається користувачеві у вигляді прогрес-бару на сторінці профілю.

Алгоритм обробки прогресу виконує такі дії:

1. Завантаження даних про проходження курсу.
2. Підрахунок загальної кількості уроків.
3. Підрахунок кількості завершених уроків.
4. Обчислення відсотка проходження.
5. Збереження оновленого результату.
6. Відображення прогресу користувачеві.

Відповідна схема наведена на рисунку 2.7.

Рисунок 2.7 – Блок-схема алгоритму обробки прогресу навчання

Окремим компонентом системи є особистий профіль користувача. Під час відкриття сторінки профілю система завантажує дані поточного користувача з Local Storage, після чого відображає персональну інформацію, список курсів та статистику проходження навчання. Також реалізовано можливість зміни імені користувача та завантаження власного аватара. Після внесення змін система оновлює дані профілю та повторно зберігає їх у локальному сховищі браузера.

Алгоритм роботи профілю включає:

1. Завантаження інформації про поточного користувача.
2. Відображення особистих даних.
3. Завантаження статистики проходження курсів.
4. Формування списку доступних курсів.
5. Завантаження аватара користувача.
6. Збереження змінених даних.
7. Оновлення відображення профілю.

Роботу профілю доцільно подати у вигляді блок-схеми-діаграми (рис. 2.8).

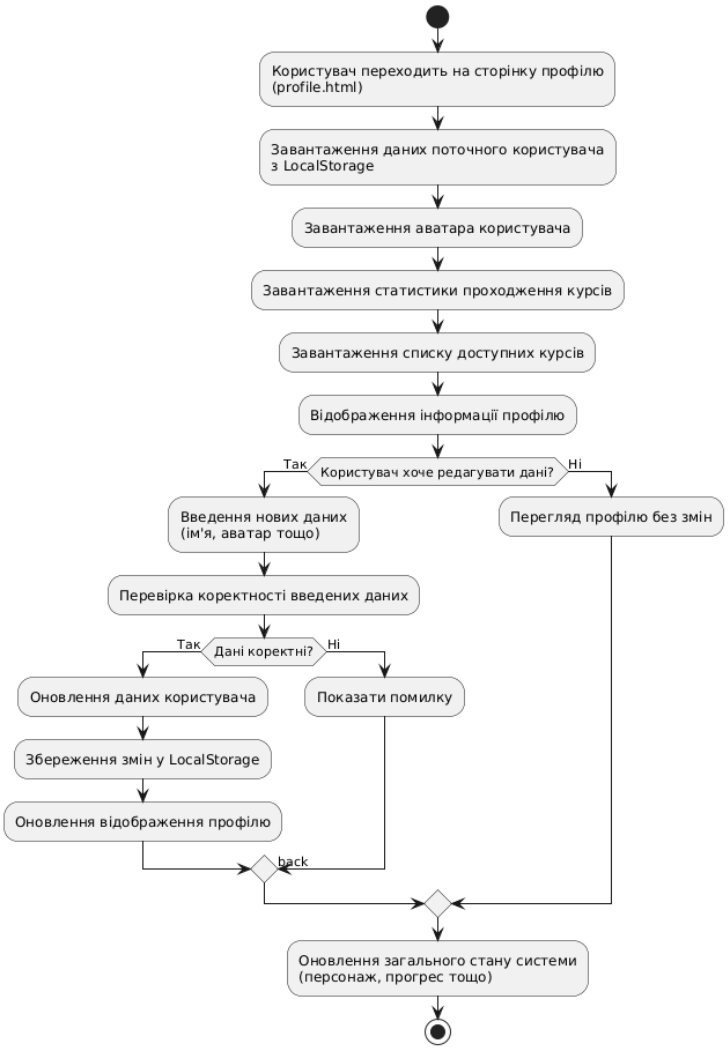

Рисунок 2.8 – Блок-схема алгоритму роботи особистого профілю користувача

Для реалізації елементів гейміфікації у системі використовується алгоритм керування віртуальним персонажем. Після відкриття профілю система перевіряє наявність розблокованого персонажа. Якщо персонаж доступний, виконується завантаження інформації про нього та визначення поточного рівня.

Рівень персонажа залежить від прогресу проходження курсу та визначається автоматично. Після обчислення рівня система відображає ім’я персонажа, його зображення та поточний рівень у профілі користувача. Якщо персонаж ще не відкритий, користувачеві відображається повідомлення про необхідність завершення навчання для його отримання.

Алгоритм роботи системи персонажа включає:

1. Перевірку наявності персонажа.
2. Завантаження інформації про персонажа.
3. Зчитування поточного прогресу курсу.
4. Обчислення рівня персонажа.
5. Відображення інформації на сторінці профілю.

Відповідний алгоритм доцільно подати у вигляді блок-схеми (рис. 2.9).

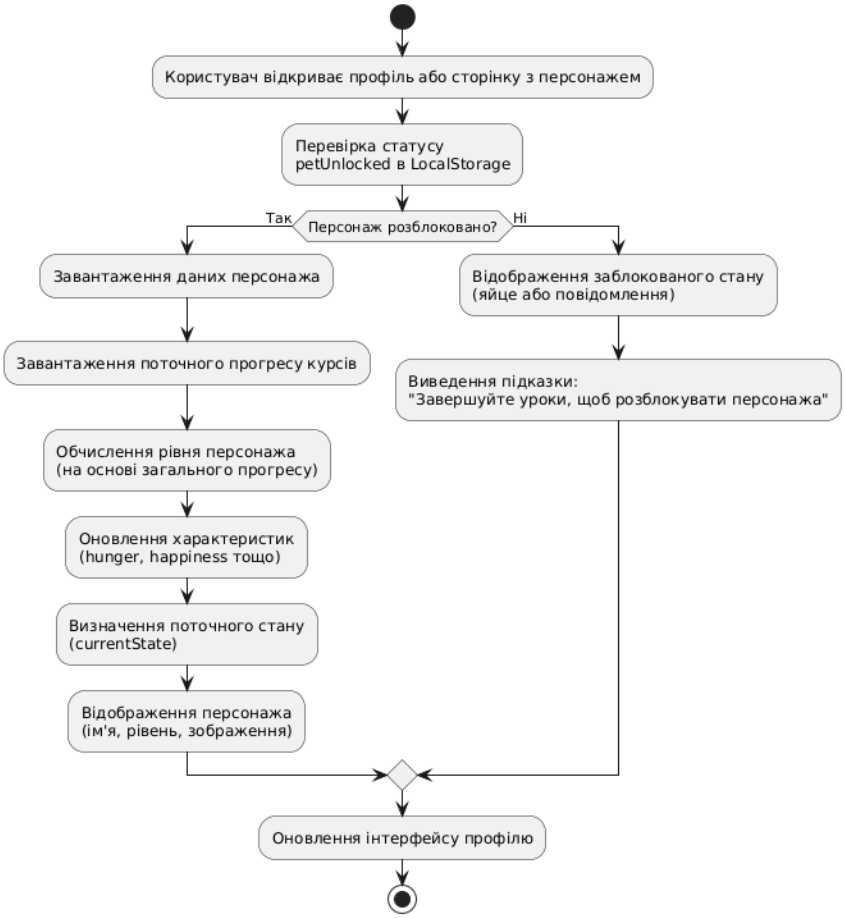

Рисунок 2.9 – Блок-схема алгоритму визначення рівня та відображення персонажа

Таким чином, алгоритмічне забезпечення розробленого вебзастосунку охоплює всі основні процеси функціонування системи: реєстрацію та авторизацію користувачів, проходження навчальних курсів, збереження та обробку прогресу, роботу особистого профілю та реалізацію елементів гейміфікації. Використання зазначених алгоритмів забезпечує коректне

функціонування навчальної платформи та створює умови для зручного проходження навчального курсу користувачем.

Висновки до розділу

У другому розділі було проведено аналіз предметної області та визначено основні вимоги до навчальної вебплатформи для вивчення мови програмування Python. Розглянуто склад вхідних і вихідних даних системи, а також визначено інформаційні потоки, що виникають під час взаємодії користувача з платформою.

У процесі проєктування системи визначено структуру зберігання даних користувачів, курсів, модулів, уроків і результатів навчання. Наведено опис основних сутностей системи, їх атрибутів та взаємозв’язків. Також побудовано об’єктно-орієнтовану модель, яка відображає логічну структуру вебзастосунку та взаємодію його основних компонентів.

У межах математичного забезпечення розглянуто підходи до обчислення прогресу проходження курсу, визначення відсотка завершення навчальних матеріалів та формування статистичних показників користувача. Наведено математичні залежності, що можуть бути використані для відображення результатів навчання та елементів гейміфікації.

Під час дослідження алгоритмічного забезпечення описано основні алгоритми функціонування системи, зокрема алгоритми реєстрації та авторизації користувачів, проходження навчальних курсів, обробки прогресу, формування статистики профілю та роботи елементів гейміфікації. Для візуалізації послідовності виконання процесів запропоновано використання відповідних схем та діаграм.

Отримані результати визначають концептуальні засади функціонування вебзастосунку та є основою для подальшої програмної реалізації системи, що буде розглянута в наступному розділі.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Для реалізації вебзастосунку для навчання програмуванню, який у процесі розробки отримав робочу назву «Codity», було використано сучасний набір вебтехнологій, орієнтований на клієнтську взаємодію, високу швидкість роботи та мінімальні вимоги до серверної частини. Обраний технологічний стек дозволив реалізувати інтерактивну освітню платформу з підтримкою навчальних модулів, практичних завдань та елементів гейміфікації.

Основою клієнтської частини вебзастосунку стали технології HTML5, JavaScript (ES6) та CSS. Мова розмітки HTML5 була використана для формування структури вебсторінок, розміщення навчального контенту, створення форм авторизації та реєстрації, а також інших елементів користувацького інтерфейсу. Мова програмування JavaScript була використана для реалізації динамічної поведінки застосунку, обробки подій користувача, навігації між сторінками та роботи з даними.

Для створення зовнішнього вигляду вебзастосунку було використано утилітарний CSS-фреймворк Tailwind CSS [9], який став основним засобом стилізації інтерфейсу. Використання готових класів оформлення дозволило швидко реалізувати адаптивний дизайн, систему відступів, типографіку, кольорову схему та інтерактивні елементи керування. Застосування Tailwind CSS також спростило підтримку стилів та забезпечило єдиний візуальний стиль усіх сторінок платформи.

Каскадні таблиці стилів CSS використовувалися додатково для створення окремих користувацьких стилів, анімацій та візуальних ефектів, які не були реалізовані стандартними засобами Tailwind CSS.

Для реалізації функціональних можливостей платформи використовувалися засоби JavaScript, зокрема для:

- керування навчальними модулями та уроками;
- відображення та оновлення прогресу користувача;
- реалізації реєстрації та авторизації;
- роботи з елементами гейміфікації;
- перевірки виконання практичних завдань;
- збереження та завантаження користувацьких даних.

Для забезпечення можливості виконання програм мовою Python безпосередньо у браузері було використано бібліотеку Pyodide [10]. Дана технологія являє собою порт інтерпретатора Python для середовища WebAssembly та дозволяє виконувати Python-код на стороні клієнта без використання серверної інфраструктури. Завдяки використанню Pyodide користувачі можуть запускати та перевіряти результати виконання програмних завдань без встановлення додаткового програмного забезпечення.

Збереження інформації про користувачів, прогрес навчання, проходження уроків та інші параметри роботи системи було реалізовано за допомогою механізму Local Storage браузера. Використання локального сховища дозволило відмовитися від застосування серверної бази даних, спростити архітектуру системи та забезпечити автономну роботу вебзастосунку.

Для реалізації навчального процесу було створено систему теоретичних та практичних занять. Навчальні матеріали, структура модулів і параметри завдань зберігаються у форматі JSON, що спрощує додавання нових курсів та масштабування платформи. Практичні завдання передбачають написання та запуск програмного коду мовою Python безпосередньо у вебінтерфейсі з подальшою автоматизованою перевіркою результатів виконання.

Розробка вебзастосунку здійснювалася у середовищі Visual Studio Code [11], яке забезпечує зручне написання, редагування та структурування програмного коду. Для прискорення процесу розробки використовувалися спеціалізовані розширення, зокрема Tailwind CSS IntelliSense, Live Server та інші засоби автоматизації роботи з вебтехнологіями.

Додатково під час розробки застосовувалися інструменти Google Chrome DevTools, за допомогою яких здійснювалася перевірка роботи клієнтської логіки застосунку, аналіз структури Local Storage, контроль коректності збереження та зчитування даних користувача, тестування механізмів авторизації, обліку прогресу та інших функціональних компонентів вебзастосунку.

Таким чином, розглянуті засоби розробки забезпечують реалізацію вебзастосунку для навчання програмуванню відповідно до поставлених вимог. Використання HTML, JavaScript, Tailwind CSS, Pyodide та Local Storage дозволяє створити інтерактивне навчальне середовище з підтримкою практичних завдань, збереженням прогресу користувача та елементами гейміфікації.

3.2 Вимоги до технічного та програмного забезпечення

Для роботи вебзастосунку з вивчення програмування не потрібне спеціалізоване обладнання або високопродуктивна комп’ютерна техніка. Оскільки застосунок реалізовано на основі клієнтських вебтехнологій, його використання можливе на більшості сучасних пристроїв, що підтримують роботу веббраузера.

Мінімальні вимоги до технічного забезпечення користувача:

- процесор із тактовою частотою від 1,5 ГГц;
- оперативна пам’ять не менше 4 ГБ;
- вільне місце на накопичувачі не менше 200 МБ;
- монітор із роздільною здатністю не нижче 1366×768;
- стабільне підключення до мережі Інтернет.

Оскільки застосунок працює безпосередньо у веббраузері, до програмного забезпечення користувача висуваються такі вимоги:

- операційна система Windows, Linux або macOS;

- сучасний веббраузер з підтримкою HTML5, CSS3 та JavaScript (Google Chrome, Mozilla Firefox, Microsoft Edge або інші аналоги);
- увімкнена підтримка Local Storage та JavaScript.

Для коректного функціонування інтерактивного редактора коду браузер повинен підтримувати технологію WebAssembly, яка використовується бібліотекою Pyodide для виконання програм мовою Python безпосередньо на стороні клієнта.

Під час розробки застосунку використовувалося таке програмне забезпечення:

- Visual Studio Code – середовище розробки;
- Google Chrome, Microsoft Edge – тестування та налагодження інтерфейсу;
- Tailwind CSS – оформлення інтерфейсу користувача;
- Pyodide – виконання програмного коду Python у браузері.

Вебзастосунок не потребує встановлення додаткового програмного забезпечення користувачем та може запускатися локально у веббраузері або бути розгорнутим на вебсервері для забезпечення віддаленого доступу користувачів.

3.3 Опис програмної реалізації

3.3.1 Загальна структура вебзастосунку

Розроблений вебзастосунок являє собою клієнтську освітню платформу для вивчення програмування, реалізовану з використанням технологій HTML5, CSS3, JavaScript та бібліотеки Pyodide. Архітектура застосунку побудована таким чином, щоб забезпечити роботу без окремої серверної частини та бази даних, використовуючи можливості сучасного веббраузера для збереження та обробки даних користувача.

Структура проєкту організована у вигляді набору HTML-сторінок, стилів, скриптів та графічних ресурсів. Основними складовими вебзастосунку є сторінки головного меню, каталогу курсів, навчальних модулів, уроків, особистого кабінету користувача та сторінки звіра. Кожна сторінка виконує окрему функцію та взаємодіє з іншими компонентами системи за допомогою JavaScript.

Для зберігання інформації про користувача використовується механізм Local Storage браузера. У локальному сховищі зберігаються дані облікового запису, прогрес проходження курсу, отриманий досвід (XP), рівень звіра користувача та інші параметри віртуального компаньйона. Такий підхід дозволяє забезпечити автономну роботу застосунку без використання серверної інфраструктури.

До складу вебзастосунку входять такі основні сторінки:

– головна сторінка, що містить загальну інформацію про платформу, навігацію курси та форми реєстрації і авторизації;

– сторінка курсів, на якій відображаються доступні навчальні матеріали та кількість модулів;

– сторінка уроків, що містить теоретичний матеріал, практичні завдання та інтерактивне середовище для виконання програмного коду;

– сторінка профілю користувача з трьома вкладками, де відображаються розпочаті та завершені курси, інформація про віртуальних тварин (їх рівень та кількість XP), а також вкладка досягнень, яка у поточній версії виконує переважно презентаційну функцію та закладає основу для подальшого розвитку системи нагород;

– сторінка самого віртуального звіра з можливістю взаємодії.

Особливістю розробленого вебзастосунку є наявність системи гейміфікації, центральним елементом якої виступає віртуальний компаньйон. Персонаж представлений у вигляді стилізованої композиції з двох змій, що відсилає до логотипу мови програмування Python. Стан компаньйона

змінюється залежно від активності користувача та прогресу проходження навчального курсу.

Загальна структура проєкту наведена на рисунку 3.1.

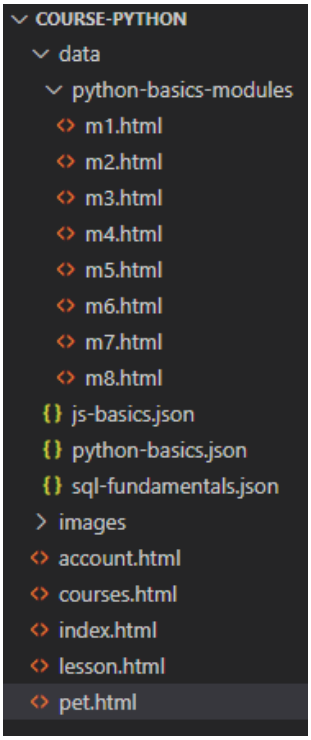

Рисунок 3.1 – Структура директорій вебзастосунку

3.3.2 Реалізація функціональних можливостей вебзастосунку

Однією з базових функціональних можливостей вебзастосунку є система реєстрації та авторизації користувачів, яка забезпечує створення персонального облікового запису, збереження індивідуальних даних користувача, а також підтримку сесії під час роботи з платформою.

Процес реєстрації передбачає заповнення користувачем відповідної форми, у якій необхідно вказати ім’я, адресу електронної пошти та пароль. Після проходження перевірки коректності введених даних інформація зберігається у локальному сховищі браузера (localStorage). Для кожного користувача створюється окремий запис, що містить персональні параметри профілю, дані про навчальний прогрес, а також характеристики віртуального компаньйона.

Форма авторизації користувача наведена на рисунку 3.2.

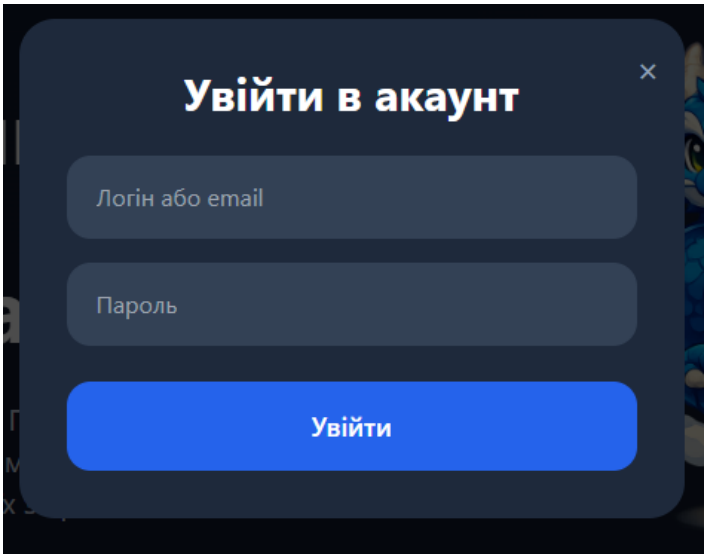

Рисунок 3.2 – Форма входу в акаунт

Збереження даних реалізовано за допомогою функції saveUser(user), яка додає нового користувача до масиву зареєстрованих користувачів та одночасно записує інформацію про поточного користувача для підтримки сесії між перезавантаженнями сторінки. Фрагмент програмного коду, що відповідає за цю функціональність, представлено на рисунку 3.3.

```
function saveUser(user) {

    let users =
        JSON.parse(localStorage.getItem('users')) || [];

    users.push(user);

    localStorage.setItem(
        'users',
        JSON.stringify(users)
    );

    localStorage.setItem(
        'currentUser',
        JSON.stringify(user)
    );
    }
```

Рисунок 3.3 – Програмна реалізація збереження даних користувача в localStorage

Авторизація в системі здійснюється шляхом введення електронної пошти та пароля. Для цього реалізовано функцію login(), яка виконує пошук користувача серед збережених записів і перевіряє відповідність введених облікових даних. У разі успішної перевірки інформація про користувача записується до localStorage, після чого відбувається перехід до сторінки навчальних курсів. Відповідний фрагмент програмної реалізації наведено на рисунку 3.4.

```
function login() {

    const input =
        document.getElementById('loginUsername')
        .value.trim();

    const password =
        document.getElementById('loginPass')
        .value;

    const users =
        JSON.parse(localStorage.getItem('users')) || [];

    const user =
        users.find(u =>
            (u.username === input ||
            u.email === input)
            &&
            u.password === password
        );

    if (user) {

        localStorage.setItem(
            'currentUser',
            JSON.stringify(user)
        );

        alert(`Вітаємо назад, ${user.username}!`);

        closeModals();

        window.location.href =
            "courses.html";

    } else {

        alert("Невірний логін або пароль");
```

Рисунок 3.4 – Програмна реалізація авторизації користувача

Для завершення роботи з обліковим записом передбачено функцію виходу із системи. Під час її виконання дані активної сесії видаляються з локального сховища браузера, що забезпечує коректне завершення сеансу користувача. Після цього здійснюється повернення на головну сторінку платформи. Програмна реалізація цієї функції представлена на рисунку 3.5.

```
function logout() {
    if (confirm("Вийти з акаунту?")) {
        localStorage.removeItem('currentUser');
        window.location.href = "index.html";
    }
}
```

Рисунок 3.5 – Програмна реалізація виходу користувача із системи

Основним елементом навчальної складової вебзастосунку є система курсів, яка забезпечує структуроване подання навчального матеріалу. Після авторизації користувач отримує доступ до сторінки курсів, де відображаються доступні навчальні напрями.

Кожен курс містить набір тематичних модулів, об'єднаних спільною темою та впорядкованих відповідно до логіки навчального процесу. На сторінці курсів відображаються назва курсу, короткий опис та інформація про прогрес його проходження.

Після вибору курсу користувач переходить до списку модулів, де може ознайомитися зі структурою навчального матеріалу. Кожен модуль складається з окремих уроків, що містять теоретичний матеріал, приклади програмного коду та практичні завдання.

```
function startCourse(courseId) {
    let user = getCurrentUser();
    if (!user.progress) user.progress = {};

    if (!user.progress[courseId]) {
        user.progress[courseId] = { progress: 0, status: "in-progress" };
    } else if (user.progress[courseId].status !== "completed") {
        user.progress[courseId].status = "in-progress";
    }

    user.progress.currentCourse = courseId;
    saveUser(user);

    window.location.href = `lesson.html?course=${courseId}`;
}
```

Рисунок 3.6 – Програмна реалізація відкриття навчального курсу

```
async function loadCourseData(course) {

    try {
        const res =
            await fetch(`data/${course.id}.json`);

        if (res.ok) {

            const data =
                await res.json();

            course.modules =
                data.modules
                    ? data.modules.length
                    : course.modules;

            course.progress =
                data.progress || 0;
        }

    } catch (e) {

        console.warn(
            "Не вдалося завантажити JSON для",
            course.id
        );

    }

    return course;
    }
```

Рисунок 3.7 – Завантаження структури курсу з JSON-файлу

Перехід між курсами, модулями та уроками реалізовано за допомогою JavaScript-логіки, яка забезпечує динамічне завантаження відповідного контенту. Структура навчальних матеріалів зберігається у вигляді окремих об'єктів даних, що містять інформацію про курси, модулі та уроки.

На рисунку 3.8 наведено сторінку курсів, на рисунку 3.9 – фрагмент структури даних, де описуються навчальні курси та їх модулі, а на рисунку 3.10 – внутрішня сторінка курсу.

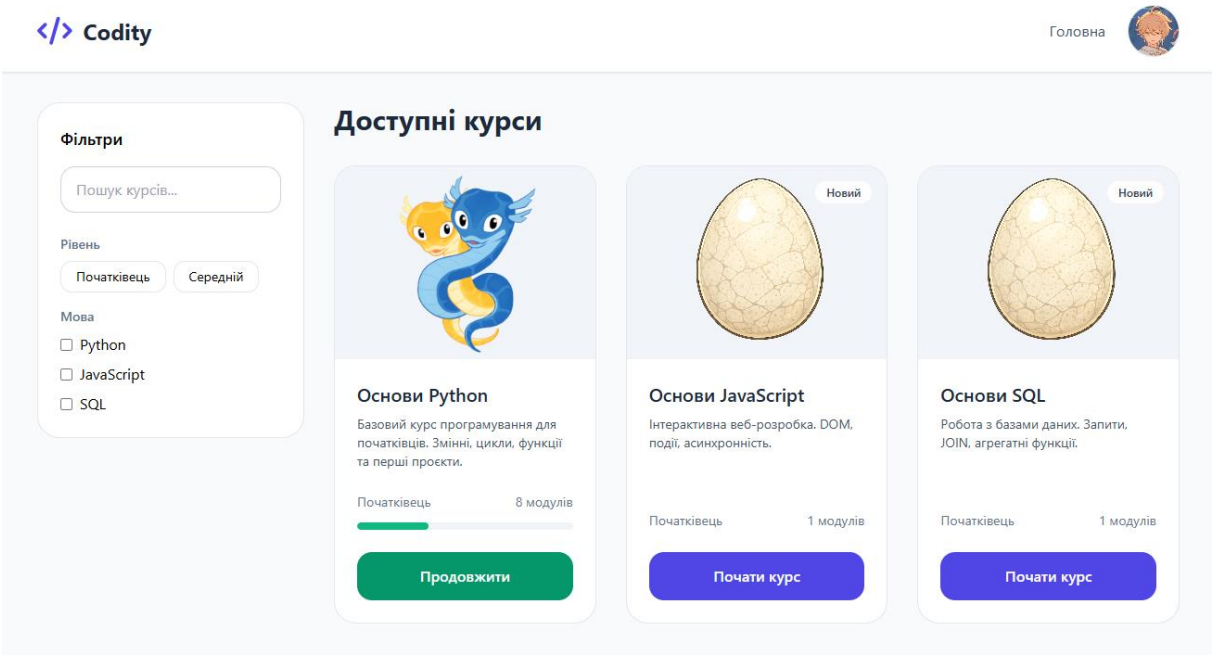

Рисунок 3.8 – Інтерфейс сторінки навчальних курсів

```
// Дані курсів
const courses = [
    {
        id: "python-basics",
        title: "Основи Python",
        description: "Базовий курс програмування для початківців. Змінні, цикли, функції та перші проєкти.",
        image: "images/egg.png",
        level: "Початківець",
        modules: 0,
        status: "not-started",
        progress: 0
    },
    {
        id: "js-basics",
        title: "Основи JavaScript",
        description: "Інтерактивна веб-розробка. DOM, події, асинхронність.",
        image: "images/egg.png",
        level: "Початківець",
        modules: 10,
        status: "not-started",
        progress: 0
    },
    {
        id: "sql-fundamentals",
        title: "Основи SQL",
        description: "Робота з базами даних. Запити, JOIN, агрегатні функції.",
        image: "images/egg.png",
        level: "Початківець",
        modules: 8,
        status: "not-started",
        progress: 0
    }
];
```

Рисунок 3.9 – Структура даних навчальних курсів

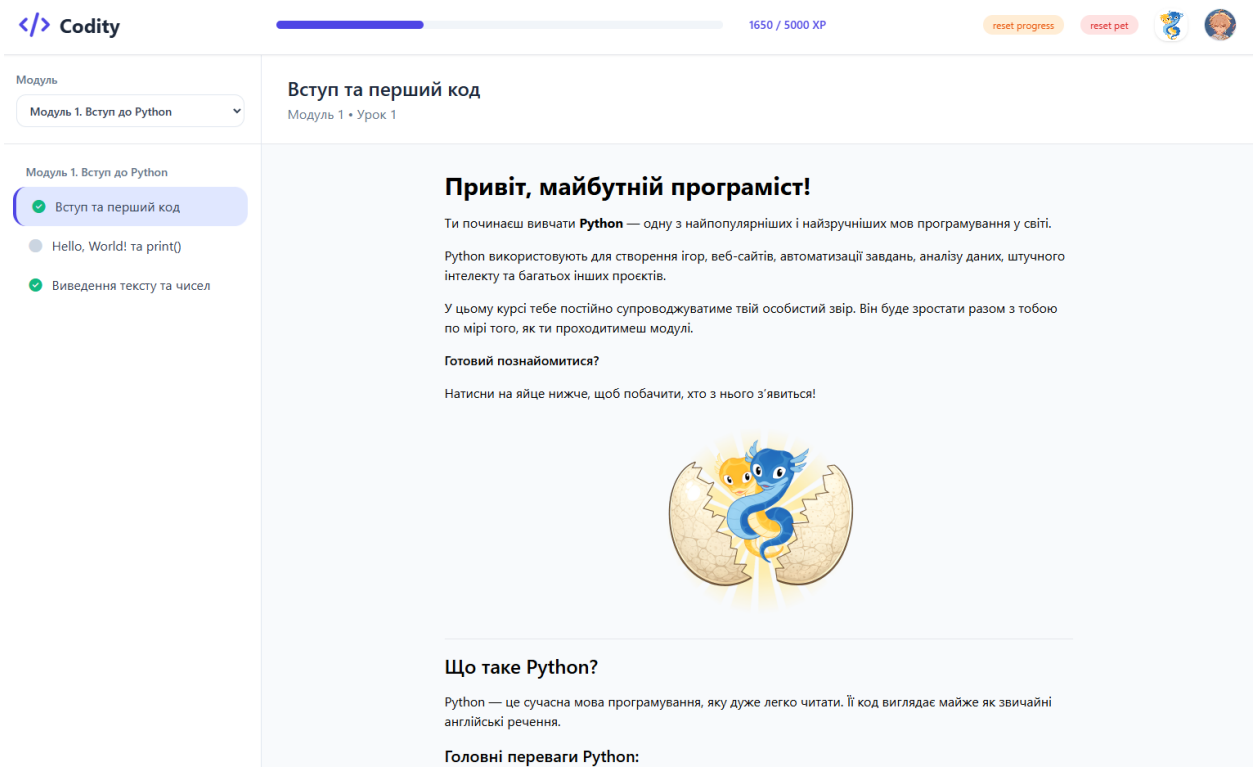

Рисунок 3.10 – Інтерфейс відкритого курсу «Основи Python»

Однією з ключових функціональних можливостей вебзастосунку є реалізація інтерактивних практичних завдань, які дозволяють користувачу безпосередньо виконувати програмний код у середовищі браузера. Для цього використовується вбудований редактор коду, інтегрований із середовищем виконання Python через технологію Pyodide, що забезпечує запуск інтерпретатора Python на стороні клієнта без необхідності серверної обробки.

Редактор коду реалізований як текстове поле з можливістю введення та редагування програмного коду, після чого користувач може запустити його виконання натисканням відповідної кнопки. Виконання коду відбувається шляхом виклику функцій Pyodide, які завантажують інтерпретатор, ініціалізують середовище виконання та передають користувацький код для обробки. Результат виконання відображається у спеціальній області виводу під редактором.

Програмна реалізація ініціалізації Pyodide та запуску коду представлена на рис. 3.11.

```
async function runPythonCode(code) {
    if (!pyodideInstance) {
        const pyodideScript = document.createElement('script');
        pyodideScript.src = "https://cdn.jsdelivr.net/pyodide/v0.26.1/full/pyodide.js";
        document.head.appendChild(pyodideScript);

        await new Promise(resolve => pyodideScript.onload = resolve);
        pyodideInstance = await loadPyodide();
    }

    let output = "";
    pyodideInstance.globals.set("print", (...args) => {
        output += args.map(String).join(" ") + "\n";
    });

    try {
        await pyodideInstance.runPythonAsync(code);
        return output.trim();
    } catch (e) {
        throw new Error(e.message);
    }
}

async function checkCode() {
    const code = document.getElementById('codeEditor').value.trim();
    const outputDiv = document.getElementById('output');
    const feedbackDiv = document.getElementById('feedback');
    const resultPanel = document.getElementById('resultPanel');

    resultPanel.classList.remove('hidden');
    outputDiv.innerHTML = ''; //

    if (!code) {
        feedbackDiv.innerHTML = `<div class="bg-red-100 text-red-700 p-4 rounded-2xl">Напиши хоч якийсь код!</div>`;
        return;
    }

    try {
        // == Pyodide ==
        let output = await runPythonCode(code);

        outputDiv.innerHTML = output.replace(/\n/g, '<br>');

        const lesson = courseData.modules[currentModuleIndex].lessons[currentLessonIndex];
        const isCorrect = checkAnswer(lesson.id, output, code);

        if (isCorrect) {
            feedbackDiv.innerHTML = `
                <div class="bg-emerald-100 text-emerald-700 p-4 rounded-2xl flex items-center gap-3">
                    <i class="fa-solid fa-circle-check text-2xl"></i>
                    <div>
                        <strong>Відмінно!</strong> Код працює правильно.<br>
                        Ти молодець!
                    </div>
```

Рисунок 3.11 – Фрагмент програмної реалізації ініціалізації Pyodide та виконання й перевірки користувацького коду

Реалізація виконання коду на клієнтській стороні зменшує навантаження на сервер та підвищує автономність роботи застосунку.

Після запуску коду система виконує базову перевірку результатів. Вона полягає у порівнянні отриманого виводу з еталонним значенням, яке визначене для конкретного завдання. У разі збігу результатів користувач отримує підтвердження правильного виконання, а також нарахування досвіду (XP).

У вебзастосунку реалізовано механізм збереження прогресу навчання, який базується на використанні локального сховища браузера localStorage. У ньому зберігається інформація про користувача, його досягнення, рівень, накопичений досвід (XP), а також стан проходження курсів, модулів і окремих уроків.

Завершення уроку фіксується шляхом оновлення відповідної структури даних у localStorage. Для кожного користувача зберігається об’єкт courses, який містить інформацію про пройдені модулі, поточний прогрес та список завершених уроків. Це дозволяє відновлювати стан навчання при повторному вході в систему.

Фрагменти структури збереження та завантаження прогресу користувача у localStorage наведено на рис. 3.12 і 3.13.

```
function saveCurrentLesson() {
    const key = `lastLesson_${currentCourse}_${USER_ID}`;
    localStorage.setItem(key, `${currentModuleIndex},${currentLessonIndex}`);
}
```

Рисунок 3.12 – Програмна реалізація збереження останнього відкритого уроку користувача

```
function saveProgress() {
    if (!courseData) return;

    const progressKey = `progress_${currentCourse}_${USER_ID}`;
    localStorage.setItem(progressKey, JSON.stringify(courseData.modules));
}

function loadProgress() {
    if (!courseData || !courseData.modules) return;

    const progressKey = `progress_${currentCourse}_${USER_ID}`;
    const saved = localStorage.getItem(progressKey);

    if (!saved) return;

    try {
        const savedModules = JSON.parse(saved);
```

Рисунок 3.13 – Фрагмент програмної реалізації збереження та відновлення прогресу користувача у localStorage

Додатковим елементом мотиваційної системи є нарахування досвіду (XP) за виконання навчальних завдань. Кожне успішно виконане завдання додає певну кількість XP до загального балансу користувача. При досягненні певного порогу відбувається підвищення рівня персонажа.

Рівень користувача безпосередньо пов’язаний із розвитком віртуального звіра (двох змій Python), який змінює свої характеристики залежно від прогресу навчання та часу. Це створює елемент гейміфікації та підвищує залученість користувача до процесу навчання.

Статистика користувача відображається на сторінці профілю та включає загальний рівень, кількість XP, прогрес проходження курсів, а також інформацію про віртуального персонажа. Дані підтягуються з localStorage та динамічно рендеряться в інтерфейсі користувача.

Для персоналізації роботи із системою на сторінці профілю реалізовано можливість редагування основних даних користувача. Користувач може змінювати відображуване ім’я, яке використовується в інтерфейсі платформи,

а також завантажувати власне зображення профілю. Обраний аватар зберігається у localStorage та автоматично відображається на всіх сторінках вебзастосунку, де присутня інформація про користувача.

Фрагмент коду, який відповідає за формування та відображення статистики користувача через звіра у профілі, наведено на рис. 3.14.

```
const savedPet = JSON.parse(localStorage.getItem(`pet_${USER_ID}`)) || {};
const progressKey = `progress_python-basics_${USER_ID}`;

let progress = 0;

const saved = localStorage.getItem(progressKey);

if (saved) {
    const modules = JSON.parse(saved);
    let total = 0, completed = 0;

    modules.forEach(mod => {
        total += mod.lessons.length;
        mod.lessons.forEach(l => {
            if (l.completed) completed++;
        });
    });

    progress = total ? Math.round((completed / total) * 100) : 0;
}

const level = Math.max(1, Math.floor(progress / 2.5));

container.innerHTML = `
    <div class="bg-white rounded-3xl border p-6">
```

Рисунок 3.14 – Фрагмент програмної реалізації відображення рівня віртуального персонажа на основі прогресу користувача

Найголовнішою особливістю розробленого вебзастосунку є система віртуального компаньйона, яка покликана підвищити мотивацію користувачів до проходження навчальних матеріалів. При старті нового курса користувач отримує інтерактивне яйце, з якого після взаємодії з ним вилуплюється власний віртуальний компаньйон. Такий підхід створює відчуття особистого супутника, що супроводжує користувача протягом усього процесу навчання. Користувач має можливість задати компаньйону власне ім’я, що додатково підсилює персоналізацію взаємодії.

Персонаж реалізований у вигляді двох стилізованих змій, що символізують логотип мови програмування Python. Таке дизайнерське

рішення було обрано для підтримання тематичної цілісності курсу та створення впізнаваного візуального образу системи.

Рисунок 3.15 – Віртуальний компаньйон курсу «Основи Python»

Для оцінювання стану персонажа реалізовано систему характеристик, що включає показники ситості, щастя та рівня. Усі значення зберігаються в об'єкті користувача та оновлюються автоматично під час взаємодії з навчальним контентом, самим компаньйоном або з плином часу.

У звіра є окрема сторінка, де відображається вся інформація про нього. Поруч з ним знаходяться кнопки – «Нагодувати», «Напоїти», «Погладити». Кожна з них по-різному впливає на характеристики звіра (змінює свої значення), але всі дії супроводжуються анімацією сердець. Показники ситості та щастя поступово зменшуються з часом, а сам звір може входити в сплячий стан, з якого його можна вивести просто клікнувши по ньому або скориставшись кнопками взаємодії.

Для створення ефекту «живої» істоти також реалізовано анімацію плавного паріння, завдяки якій персонаж постійно перебуває в русі навіть при своєму статичному зображенні.

Система рівнів персонажа безпосередньо пов’язана з прогресом користувача. Під час проходження навчальних матеріалів та виконання інтерактивних завдань користувач отримує досвід (XP), який одночасно накопичується і для компаньйона. Зі збільшенням кількості досвіду підвищується рівень персонажа, а також поступово змінюється його розмір, що візуально демонструє розвиток протягом навчання.

Програмну реалізацію оновлення характеристик персонажа наведено на рис. 3.16.

```
function checkDecay() {
    const now = Date.now();

    //  HUNGER (-1 кожні 5 хвилин)
    const hungerSeconds = (now - pet.lastHungerCheck) / 1000;
    const hungerLoss = Math.floor(hungerSeconds / (5 * 60));

    if (hungerLoss > 0) {
        pet.hunger = Math.max(0, pet.hunger - hungerLoss);
        pet.lastHungerCheck += hungerLoss * 5 * 60 * 1000;
    }

    //  HAPPINESS (-1 кожні 10 хвилин)
    const happinessSeconds = (now - pet.lastHappinessCheck) / 1000;
    const happinessLoss = Math.floor(happinessSeconds / (10 * 60));

    if (happinessLoss > 0) {
        pet.happiness = Math.max(0, pet.happiness - happinessLoss);
        pet.lastHappinessCheck += happinessLoss * 10 * 60 * 1000;
    }

    if (hungerLoss > 0 || happinessLoss > 0) {
        savePet();
        updateUI();
    }
}
```

Рисунок 3.16 – Програмна реалізація зниження характеристик компаньйона

Важливою складовою системи звіра є також зміна його зовнішнього вигляду. Залежно від значень характеристик ситості, щастя і часу відображаються різні графічні стани компаньйона, а саме: нормальний, сумний, щасливий та сплячий. (рис. 3.1).

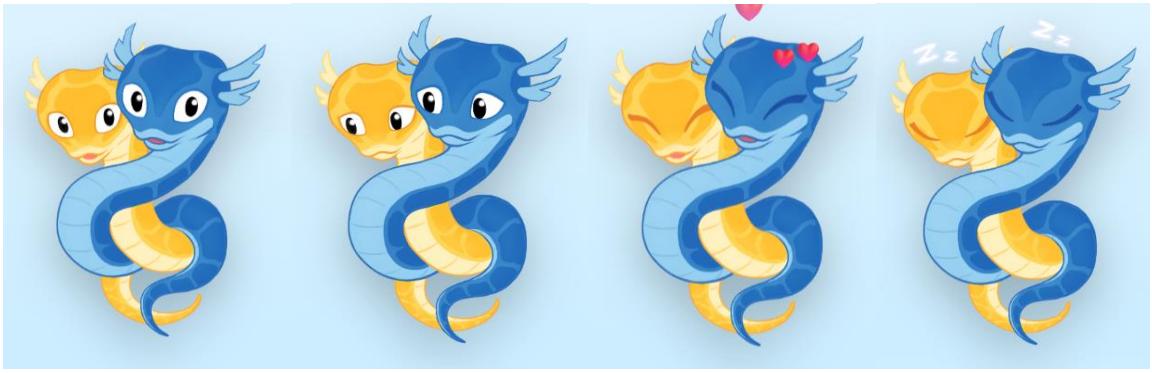

Рисунок 3.17 – Варіанти відображення станів віртуального компаньйона

Таким чином, реалізований механізм віртуального компаньйона поєднує елементи гейміфікації та освітнього процесу. На відміну від звичайних систем відображення прогресу, він формує емоційну прив’язаність користувача до власного персонажа. Результати навчання впливають не лише на числові показники успішності, а й на стан та розвиток компаньйона, що стимулює регулярне проходження курсу та підтримує зацікавленість у навчанні.

3.3.3 Реалізація користувацького інтерфейсу

У процесі розробки вебзастосунку значна увага приділялася створенню зрозумілого та сучасного користувацького інтерфейсу. Основною метою було забезпечення комфортної взаємодії користувача з навчальними матеріалами та елементами гейміфікації незалежно від типу пристрою.

Для побудови інтерфейсу використовувався CSS-фреймворк Tailwind CSS, який дозволяє швидко створювати адаптивні сторінки за допомогою готових класів оформлення. Завдяки використанню цього підходу вдалося забезпечити єдиний стиль усіх сторінок вебзастосунку, включаючи головну сторінку, сторінки курсів, уроків, профілю користувача та віртуального компаньйона.

Також увагу приділено адаптивності інтерфейсу. Розташування елементів автоматично змінюється залежно від розміру екрана, що забезпечує коректне відображення вебзастосунку як на персональних комп'ютерах, так і на мобільних пристроях.

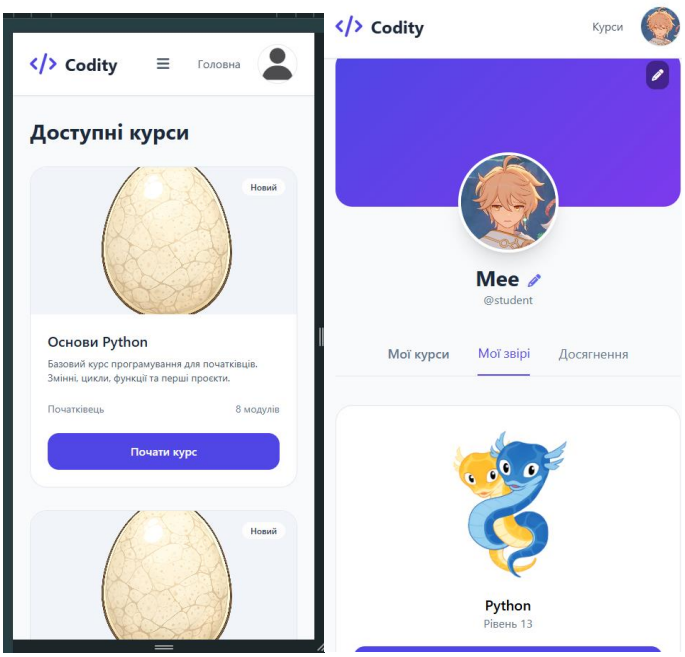

Рисунок 3.18 – Інтерфейс вебзастосунку: мобільна версія та адаптація до зменшеного розміру вікна браузера

Для покращення користувацького досвіду реалізовано анімаційні ефекти, які використовуються під час взаємодії з елементами інтерфейсу. Анімації застосовуються до кнопок, випадаючих меню та віртуального компаньйона, що робить роботу із системою більш динамічною та візуально привабливою.

Навігація між сторінками організована таким чином, щоб користувач міг швидко переходити між курсами, уроками, профілем та сторінкою компаньйона. Для цього використовується єдина структура навігаційних елементів, що забезпечує послідовність взаємодії з усіма складовими вебзастосунку.

3.4 Керівництво користувача

Для початку роботи з вебзастосунком користувачу необхідно відкрити головну сторінку платформи Codity у веббраузері. На головній сторінці

розміщено короткий опис можливостей системи, інформацію про доступні навчальні курси, а також кнопки входу та реєстрації.

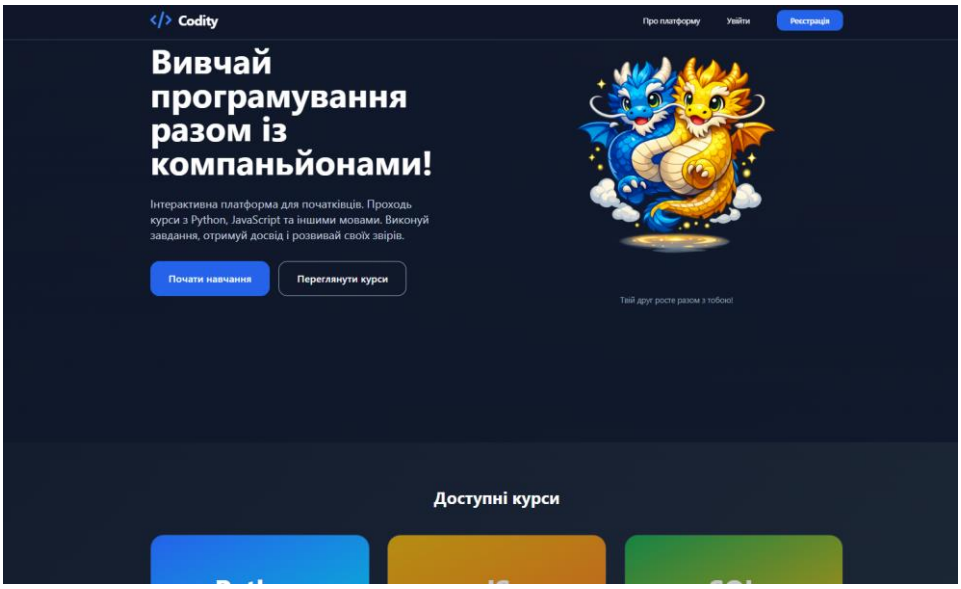

Рисунок 3.19 – Головна сторінка вебзастосунку Cody

Якщо користувач ще не має облікового запису, необхідно натиснути кнопку «Реєстрація» та заповнити форму створення акаунта. Під час реєстрації потрібно вказати ім’я користувача, адресу електронної пошти та пароль. Після успішного створення облікового запису користувач автоматично отримує доступ до функціональних можливостей платформи.

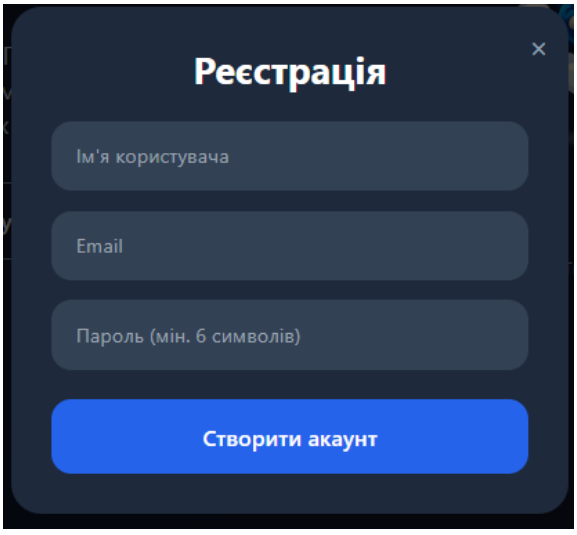

Рисунок 3.20 – Форма реєстрації користувача

Для входу до вже існуючого акаунта необхідно натиснути кнопку «Увійти» та ввести адресу електронної пошти або ім’я користувача разом із паролем. Після успішної авторизації здійснюється перехід до сторінки навчальних курсів.

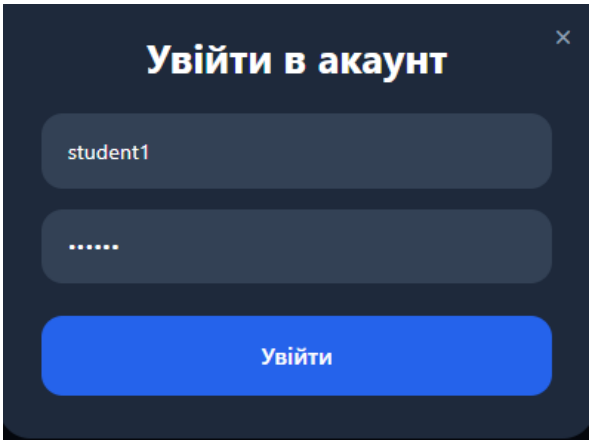

Рисунок 3.21 – Форма авторизації користувача

Після входу користувач потрапляє на сторінку курсів, де відображаються доступні навчальні програми. Для початку навчання необхідно обрати потрібний курс та натиснути на його картку.

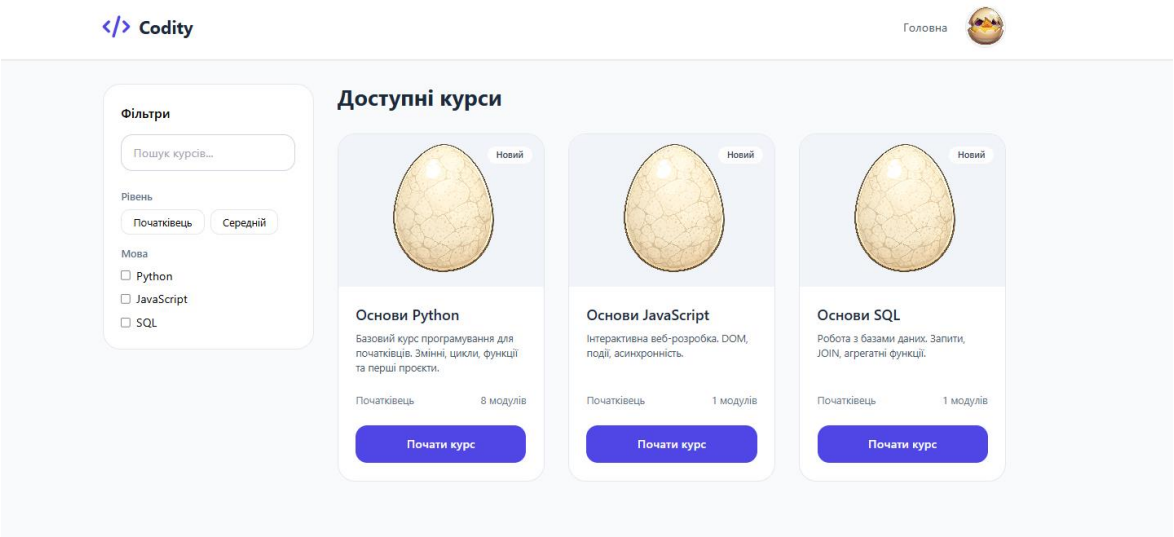

Рисунок 3.22 – Сторінка доступних навчальних курсів

Після відкриття курсу відображається перший модуль (або той, що був останнім при повторному поверненню). Вступна частина пропонує користувачу натиснути на яйце і подивитись, якого звіра вони отримали на цей курс. Користувач може послідовно переходити між модулями відповідно до структури навчальної програми. Кожен модуль містить набір уроків, присвячених відповідним темам.

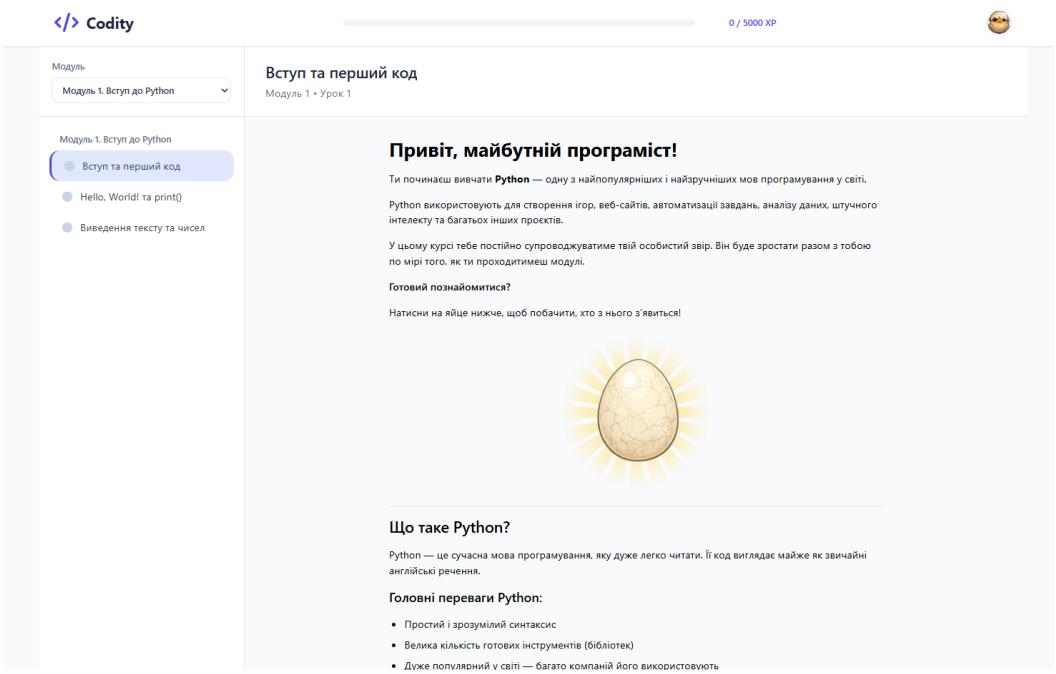

Рисунок 3.23 – Сторінка курсу із відкритим першим модулем

Після того, як користувач натисне на яйце, він побачить свого звіра, а також з’явиться іконка поруч із його аватаркою.

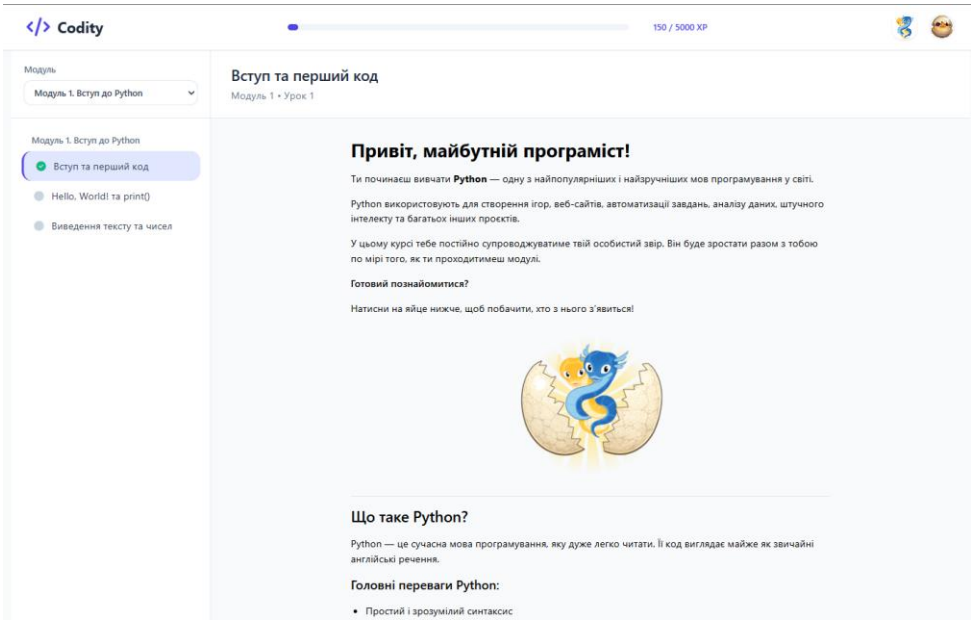

Рисунок 3.24 – Відкритий звір на сторінці курсу

Всі уроки відображаються в лівій частині, де і модулі. Теоретичний урок містить теорію із поясненням теми та прикладами програмного коду. Для навігації між уроками передбачено кнопки переходу до попереднього та наступного матеріалу.

Далі для закріплення отриманих знань після вивчення теорії користувач може виконати практичні завдання. На сторінці інтерактивного уроку розміщено вбудований редактор коду, у якому необхідно написати або доповнити програму відповідно до умови завдання. Після натискання кнопки запуску система виконує код безпосередньо у браузері та відображає результат виконання.

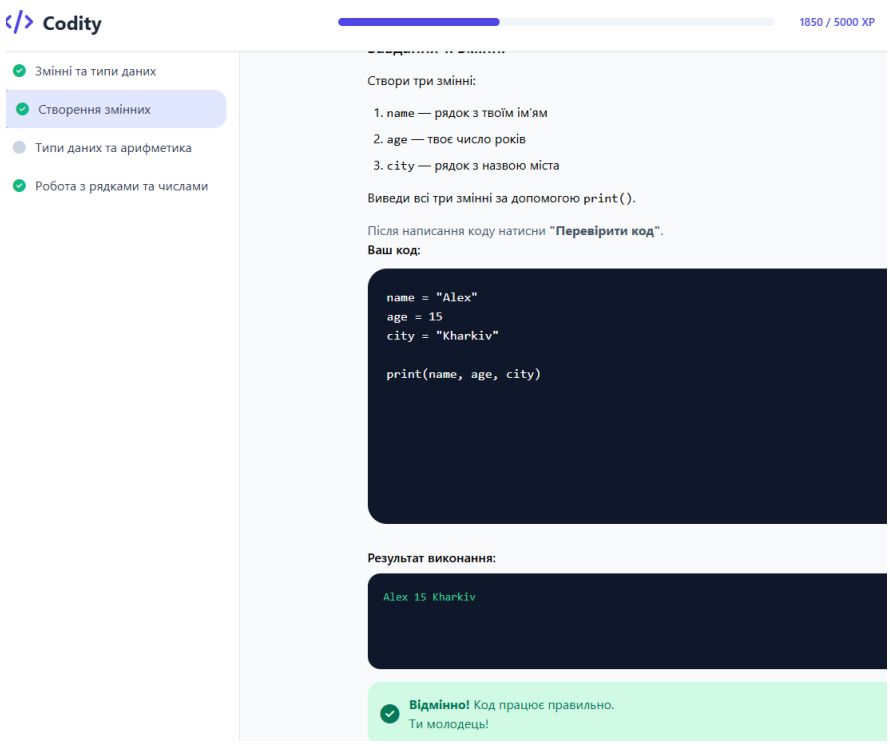

Рисунок 3.25 – Правильно виконане практичне завдання в редакторі коду

У разі правильного виконання завдання користувач отримує досвід (XP), а прогрес проходження уроку автоматично зберігається. Накопичений досвід впливає на рівень віртуального компаньйона та загальний прогрес навчання.

Окрім верхнього прогрес-бара, поточний стан проходження курсу можна переглянути на сторінці профілю користувача. Для переходу до профілю необхідно натиснути на аватар у верхній частині інтерфейсу та обрати відповідний пункт меню.

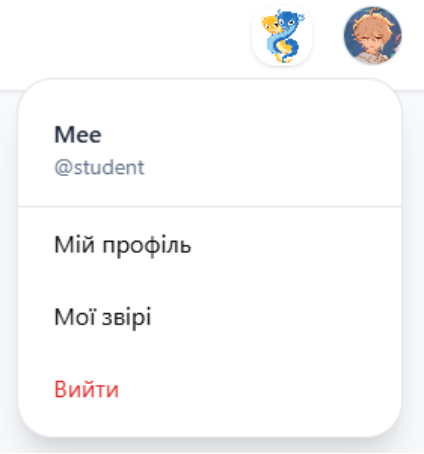

Рисунок 3.26 – Випадаючий список (dropdown) профілю користувача

У профілі відображаються розпочаті та завершені курси, кількість отриманого досвіду, персонаж та його рівень, а також інша статистична інформація. Тут користувач може змінити власне ім’я та встановити нове зображення профілю.

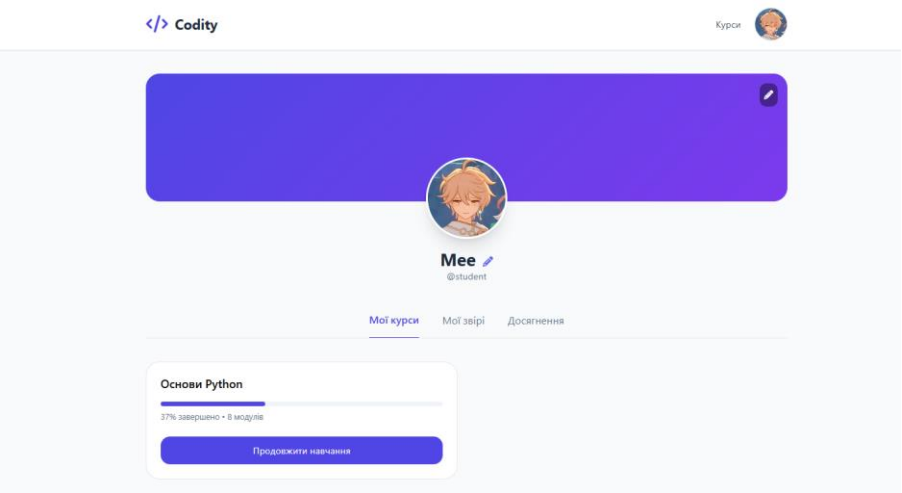

Рисунок 3.27 – Сторінка профілю користувача

Для перегляду інформації про віртуального компаньйона необхідно перейти на його окрему сторінку через навігаційне меню або через вкладку «Мої звірі».

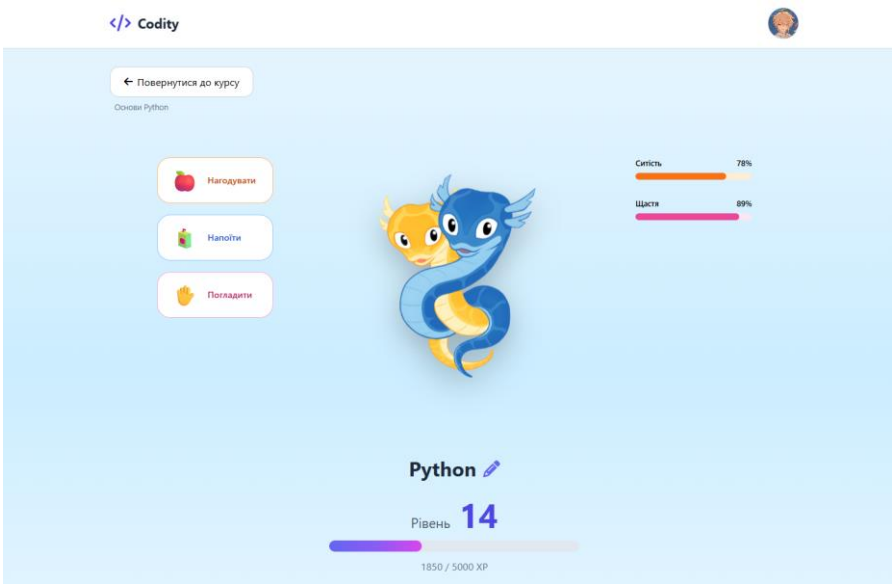

Рисунок 3.28 – Сторінка віртуального компаньйона

На сторінці звіра відображаються його поточний рівень, кількість досвіду, показники ситості та щастя. Користувач може взаємодіяти зі звіром за допомогою кнопок «Нагодувати», «Напоїти» та «Погладити», які впливають на його характеристики та супроводжуються анімаційними ефектами.

Також на цій сторінці доступна можливість змінити ім’я компаньйона. Залежно від його стану можуть відображатися різні емоції та анімації, що дозволяє візуально оцінювати самопочуття персонажа.

Після завершення роботи із системою користувач може вийти зі свого облікового запису за допомогою пункту «Вийти» у випадаючому меню профілю. Після цього активна сесія завершується, а користувач повертається на головну сторінку вебзастосунку.

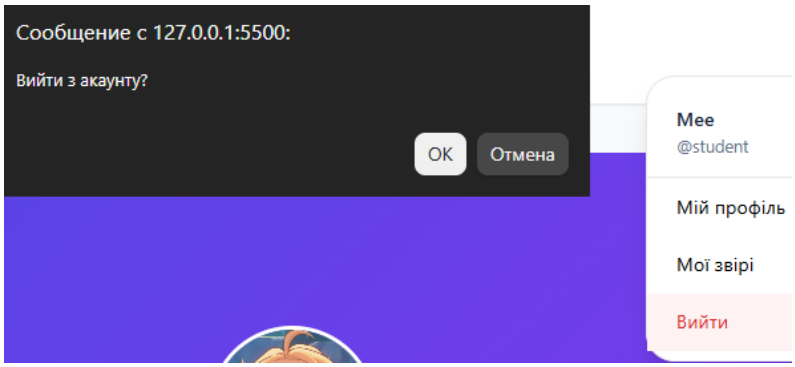

Рисунок 3.29 – Діалогове вікно підтвердження виходу з акаунта

Таким чином, вебзастосунок Codity забезпечує простий та зрозумілий процес навчання, що поєднує проходження теоретичних матеріалів, виконання практичних завдань, відстеження особистого прогресу та взаємодію з віртуальним компаньйоном.

Висновки до розділу

У третьому розділі було розглянуто програмне та технічне забезпечення вебзастосунку Codity, а також особливості його реалізації. Проведено аналіз засобів розробки, обґрунтовано вибір використаних технологій та визначено вимоги до програмного й технічного забезпечення, необхідного для функціонування системи.

У ході розробки реалізовано вебзастосунок для вивчення основ програмування мовою Python, який забезпечує реєстрацію та авторизацію користувачів, проходження навчальних курсів, роботу з інтерактивними уроками та виконання практичних завдань безпосередньо у браузері за допомогою технології Pyodide. Для збереження даних користувачів, навчального прогресу та параметрів системи використано локальне сховище браузера localStorage.

Окрему увагу приділено реалізації механізмів гейміфікації. У вебзастосунку створено систему віртуального компаньйона у вигляді стилізованих змій Python, стан яких залежить від часу та прогресу користувача в навчальних завданнях. Такий підхід дозволяє підвищити зацікавленість у проходженні навчального матеріалу та забезпечує додаткову мотивацію до навчання.

Також було розроблено сучасний користувацький інтерфейс із використанням фреймворку Tailwind CSS, реалізовано адаптивне відображення сторінок для різних типів пристроїв та забезпечено зручну навігацію між основними функціональними модулями системи.

Наведене керівництво користувача демонструє порядок роботи з вебзастосунком та підтверджує доступність основних функцій для кінцевого користувача. Отримані результати свідчать про успішну реалізацію поставлених завдань і готовність вебзастосунку до практичного використання як навчальної платформи для вивчення основ програмування.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

4.1 Організаційно-правові основи забезпечення безпеки праці

Охорона праці розглядається як ключовий елемент організації трудового процесу, спрямований на збереження життя, здоров’я та працездатності працівників під час виконання професійних завдань. Вона охоплює сукупність правових, організаційних, технічних, санітарно-гігієнічних і лікувально-профілактичних заходів, спрямованих на формування безпечних і нешкідливих умов праці [13].

У сучасних умовах значення охорони праці зростає, особливо у сфері інформаційних технологій, де основна діяльність пов’язана з тривалою роботою за комп’ютером. Хоча така діяльність не пов’язана зі значними фізичними навантаженнями, вона супроводжується низкою специфічних ризиків, зокрема перевтомою, напруженням зору, статичними навантаженнями та психоемоційним напруженням. Тому забезпечення безпечних умов праці для користувачів ПК є необхідною умовою збереження їхнього здоров’я та підвищення ефективності роботи [14].

Державна політика України у сфері охорони праці ґрунтується на принципі пріоритетності життя і здоров’я працівників над результатами виробничої діяльності, повної відповідальності роботодавця за створення належних, безпечних і здорових умов праці, а також системного підходу до управління професійними ризиками [13].

Нормативно-правове забезпечення охорони праці в Україні базується на Конституції, законах та підзаконних актах. Основоположним документом у цій сфері є Закон України «Про охорону праці» № 2694-XII, який визначає правові, економічні та організаційні засади забезпечення безпеки працівників, права та обов’язки роботодавців і працівників [13].

Важливим нормативним актом є також Кодекс законів про працю України, який регулює трудові відносини та містить положення щодо гарантій безпечних умов праці [15]. Додатково діють державні санітарні норми та правила, зокрема ті, що регламентують вимоги до робочих місць з екранними пристроями, параметрів мікроклімату, рівнів випромінювання електромагнітних полів та штучного освітлення [16-19].

Для системного управління охороною праці застосовуються міжнародні стандарти, адаптовані в Україні, зокрема ДСТУ ISO 45001:2019 «Системи управління охороною здоров’я та безпекою праці» [20]. Цей стандарт встановлює вимоги до створення ефективної системи управління ризиками на робочому місці.

Таким чином, сформовані організаційно-правові засади охорони праці дозволяють мінімізувати вплив шкідливих виробничих факторів, що виникають під час тривалої роботи за екранними пристроями. Комплексне дотримання цих норм забезпечує належне підґрунтя для безпечного ведення інженерно-програмних робіт, поєднуючи державні стандарти з обов’язками конкретних виконавців.

4.2 Характеристика об’єкта та виявлення потенційних небезпек

Об’єктом проєктування є робоче місце студента-розробника вебзастосунку для вивчення мов програмування з елементами гейміфікації. Розробка здійснюється переважно в умовах домашнього приміщення. Робоче місце являє собою стіл з персональним комп’ютером (системний блок, монітор, клавіатура, миша), офісне крісло, джерела освітлення та електромережу. Характер діяльності – тривала сидяча робота за комп’ютером (програмування, тестування, робота з графікою та документацією), що супроводжується значним розумовим навантаженням і обмеженою руховою активністю.

Незважаючи на відсутність важкого фізичного виробництва, на даному робочому місці присутні різноманітні небезпечні та шкідливі виробничі фактори, що можуть негативно впливати на здоров’я, працездатність і загальний стан розробника. Ідентифікація та аналіз цих факторів здійснювалися відповідно до загальних засад управління безпекою праці та оцінювання професійних ризиків.

Основною метою даного підрозділу є ідентифікація небезпечних та шкідливих факторів, які можуть виникати під час виконання робіт, а також визначення їх можливих наслідків. За природою дії всі потенційні чинники на робочому місці розробника можна поділити на такі основні групи: фізичні, психофізіологічні, хімічні та біологічні.

Фізичні небезпечні фактори пов’язані з умовами навколишнього середовища та технічними засобами, що використовуються під час роботи. До них належать нераціональне або недостатнє освітлення робочої зони [19], підвищений рівень шуму від роботи системи охолодження ПК, невідповідні параметри мікроклімату приміщення [18], а також електромагнітне випромінювання від монітора та супутньої техніки [16, 17]. Окрім цього, критичним ризиком є можливість ураження електричним струмом у разі пошкодження ізоляції, відсутності заземлення або порушення правил експлуатації електроустановок [22].

Психофізіологічні фактори є найбільш вираженими та характерними для ІТ-спеціалістів. Вони зумовлені тривалим перебуванням у статичній позі (сидячи), що створює підвищене навантаження на хребет і м'язи, а також тривалим фокусуванням зору на екрані монітора (ризик розвитку зорового синдрому). Крім того, процес розробки програмного забезпечення супроводжується високим рівнем інтелектуального навантаження, тривалою концентрацією уваги та необхідністю дотримання часових обмежень (дедлайнів), що може викликати стрес, перевтому й емоційне вигорання [14].

Хімічні та біологічні фактори на даному робочому місці мають опосередкований характер і переважно пов’язані з якістю повітряного

середовища в приміщенні. Хімічний вплив може виявлятися через накопичення пилу, виділення з матеріалів оздоблення кімнати або використання побутової хімії під час прибирання. Біологічні чинники включають патогенні мікроорганізми (бактерії, віруси, грибки), які можуть розвиватися при недостатній вентиляції приміщення. Вплив цих факторів може призвести до алергічних реакцій або зниження загального імунітету [18].

Класифікацію основних небезпечних та шкідливих факторів за їх природою наведено в таблиці 4.1.

Таблиця 4.1 – Потенційні небезпеки при розробці вебзастосунку

№	Небезпечний фактор	Джерело виникнення	Можливі наслідки	Категорія фактора
1	Недостатнє або надмірне освітлення	Неправильне розташування освітлювальних приладів	Втома очей, головний біль, зниження продуктивності	Фізичний
2	Підвищений рівень шуму	Робота комп’ютерів, вентиляторів, зовнішні джерела	Зниження концентрації, подразнення	Фізичний
3	Невідповідний мікроклімат	Недостатня вентиляція, опалення або кондиціонування	Дискомфорт, зниження працездатності	Фізичний
4	Електромагнітне випромінювання	Комп’ютери, маршрутизатори	Втома, погіршення самопочуття	Фізичний
5	Електрична небезпека	Несправне обладнання, пошкоджені кабелі	Ураження електричним струмом	Фізичний
6	Забруднення повітря	Пил, хімічні речовини	Алергічні реакції, подразнення	Хімічний
7	Біологічні чинники	Бактерії, віруси, грибки	Захворювання, алергії	Біологічний
8	Статичне навантаження	Тривала робота сидячи	Захворювання опорно-рухового апарату	Психофізіо-логічний
9	Зорове навантаження	Робота за монітором	Погіршення зору, сухість очей	Психофізіо-логічний
10	Психоемоційне перевантаження	Дедлайни, складні завдання	Стрес, вигорання	Психофізіо-логічний

Окрім зазначених факторів, у сучасних умовах слід враховувати додаткові техногенні та специфічні ризики. Зокрема, використання великої кількості електрообладнання створює потенційну пожежну небезпеку на

робочому місці [22]. Також, з урахуванням воєнного стану в Україні, обов'язковому врахуванню підлягають ризики, пов’язані з дією воєнного стану (повітряні тривоги, загроза обстрілів, перебої в роботі критичної інфраструктури), що вимагає чіткого знання правил евакуації та цивільного захисту [21].

Таким чином, навіть у відносно безпечному середовищі розробки програмного забезпечення присутній комплекс небезпечних і шкідливих факторів, які потребують врахування. Їх своєчасне виявлення дозволяє сформувати ефективні заходи щодо зменшення негативного впливу, що буде розглянуто у наступному підрозділі.

4.3 Дослідження ризику реалізації потенційних небезпек на об’єкті проєктування та розробка заходів щодо їх попередження

Оцінка професійних ризиків виступає одним із центральних елементів системи управління охороною праці. Це системний процес, який включає виявлення небезпек, аналіз імовірності їх виникнення та оцінку можливих наслідків для працівника. Основною метою оцінки ризику є попередження нещасних випадків, професійних захворювань та інших негативних наслідків шляхом вживання своєчасних профілактичних заходів.

Процес оцінювання ризиків передбачає такі основні етапи:

- ідентифікацію потенційних небезпек;
- визначення ймовірності їх виникнення;
- оцінку тяжкості можливих наслідків;
- визначення рівня ризику (на основі ймовірності та наслідків);
- класифікацію ризиків за рівнем небезпеки;
- розробку заходів щодо зниження або контролю ризиків.

У даній роботі для оцінки ризиків застосовано два методи: матрицю оцінювання ризиків та аналіз «дерева відмов». Матричний метод дозволяє

поєднати ймовірність виникнення небезпеки з тяжкістю її наслідків і отримати інтегральну оцінку рівня ризику. Метод «дерева відмов» дає змогу простежити причинно-наслідкові зв’язки та виявити основні фактори, що призводять до небажаної події.

Для аналізу були обрані найбільш суттєві небезпеки, виявлені в попередньому підрозділі. Оцінювання ризиків здійснювалося із застосуванням шкали, що передбачає класифікацію наслідків (I – катастрофічні, II – критичні, III – значні, IV – незначні) та ймовірності їх виникнення (A – часта, B – можлива, C – випадкова, D – віддалена, E – неймовірна). Результати оцінки ризиків наведено в таблиці 4.2.

Таблиця 4.2 – Оцінка професійних ризиків

№	Небезпека	Ймовірність (рівень)	Тяжкість наслідків (категорія)	Індекс ризику	Рівень ризику
1	Погіршення зору	Можлива (B)	Критична (II)	2B	Неприпустимий
2	Психоемоційне перенапруження (вигорання)	Часта (A)	Критична (II)	2A	Неприпустимий
3	Порушення опорно-рухового апарату	Можлива (B)	Критична (II)	2B	Неприпустимий
4	Електрична небезпека	Віддалена (D)	Критична (II)	2D	Небажаний
5	Пожежна небезпека	Віддалена (D)	Катастрофічна (I)	1D	Небажаний
6	Надзвичайна ситуація (Воєнна загроза)	Можлива (B)	Катастрофічна (I)	1B	Неприпустимий

Аналіз таблиці 4.2 показує, що найвищий рівень ризику (неприпустимий) мають чотири фактори. Три з них безпосередньо пов’язані з тривалою роботою за комп’ютером: погіршення зору, психоемоційне перенапруження (професійне вигорання) та порушення опорно-рухового апарату. Ці ризики характеризуються високою або частою ймовірністю виникнення та критичним рівнем тяжкості наслідків.

Четвертим фактором з неприпустимим рівнем ризику є воєнна загроза, яка має катастрофічний потенціал наслідків при можливій ймовірності. Електрична та пожежна небезпеки віднесені до небажаного рівня ризику через нижчу (віддалену) ймовірність, хоча їх наслідки також можуть бути критичними або катастрофічними.

Для детального вивчення причинно-наслідкових зв’язків найбільш критичних ризиків було побудовано дерева відмов.

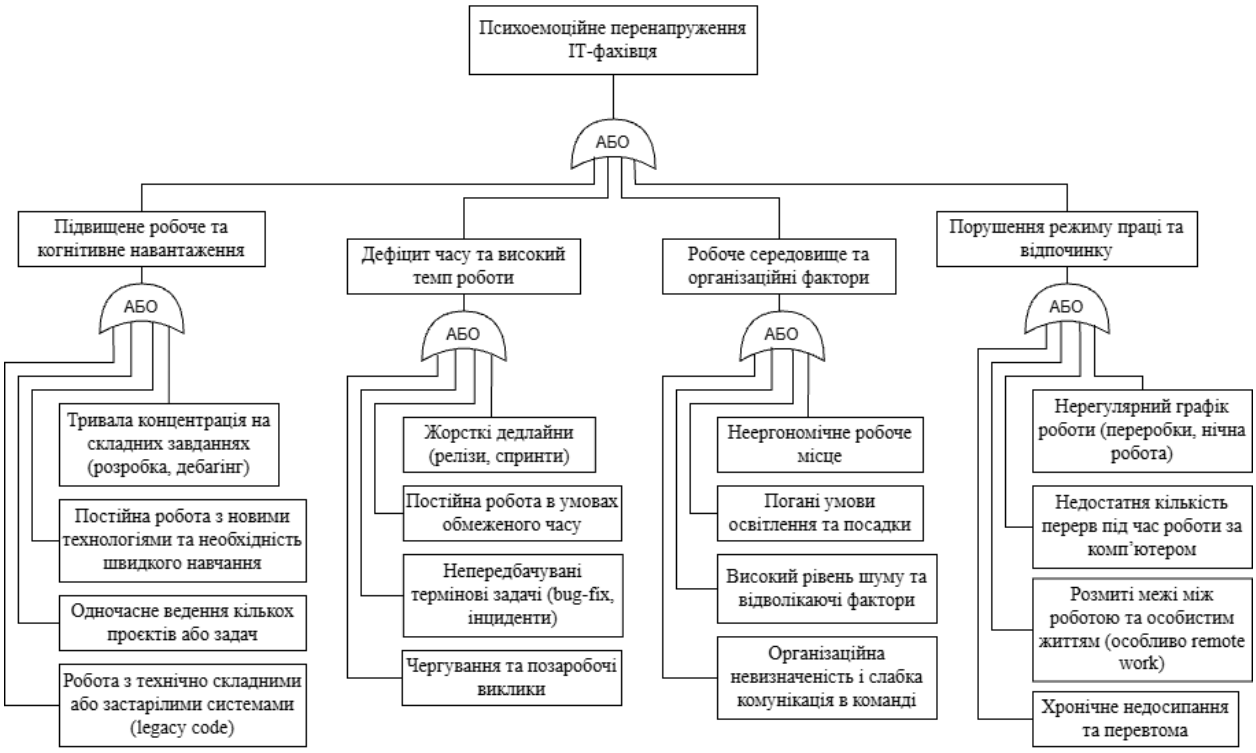

Рисунок 4.1 – Дерево відмов для аналізу психоемоційного перенапруження (професійного вигорання) IT-фахівця

Побудоване дерево відмов унаочнює, що професійне вигорання виникає внаслідок комплексної дії когнітивного навантаження, дефіциту часу, факторів робочого середовища, порушення режиму праці та організаційних факторів. Оскільки більшість подій пов’язані оператором «АБО», для запобігання вигоранню необхідно блокувати базові причини: впроваджувати регламентовані перерви, планувати робочий час та застосовувати техніки управління стресом.

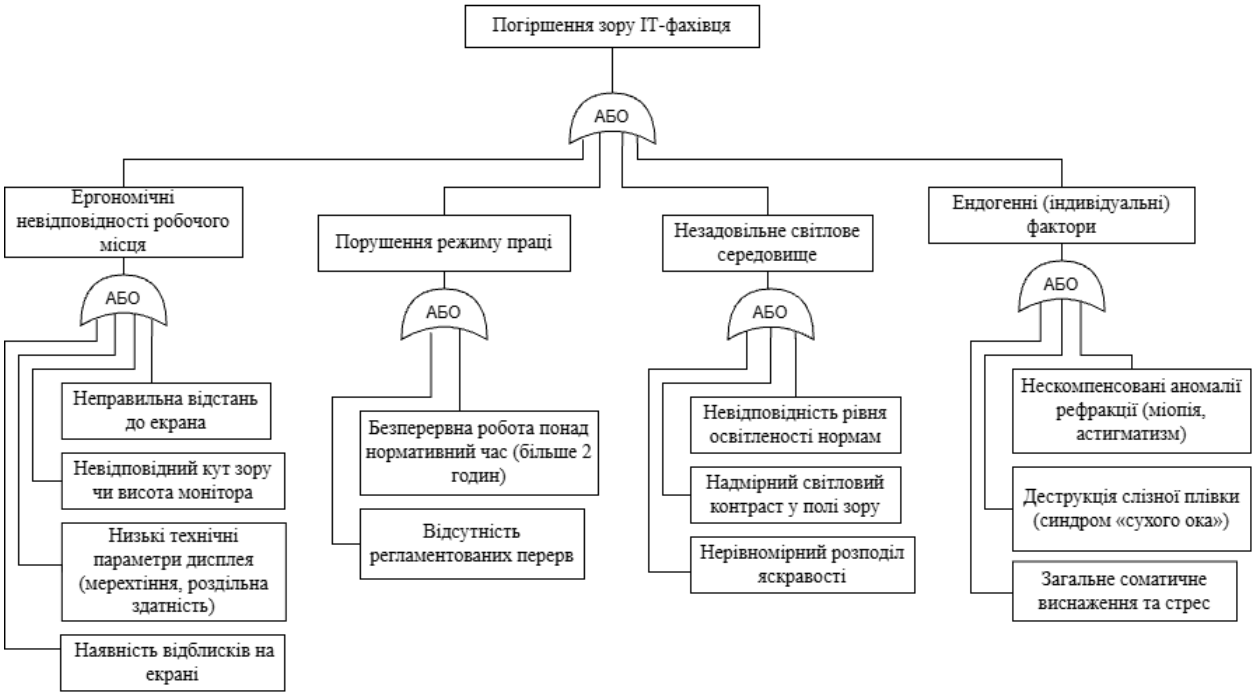

Рисунок 4.2 – Дерево відмов для аналізу ризику погіршення зору (комп’ютерного зорового синдрому) ІТ-фахівця

Як видно з рисунка 4.2, погіршення зору є наслідком тривалого візуального навантаження, неправильної ергономіки робочого місця, незадовільного світлового середовища та індивідуальних (ендогенних) факторів. Основну роль відіграють відсутність перерв, неякісне освітлення та неправильне положення монітора. Запобігти цьому ризику дозволяє дотримання правила 20-20-20, правильне розміщення обладнання та якісне освітлення робочої зони.

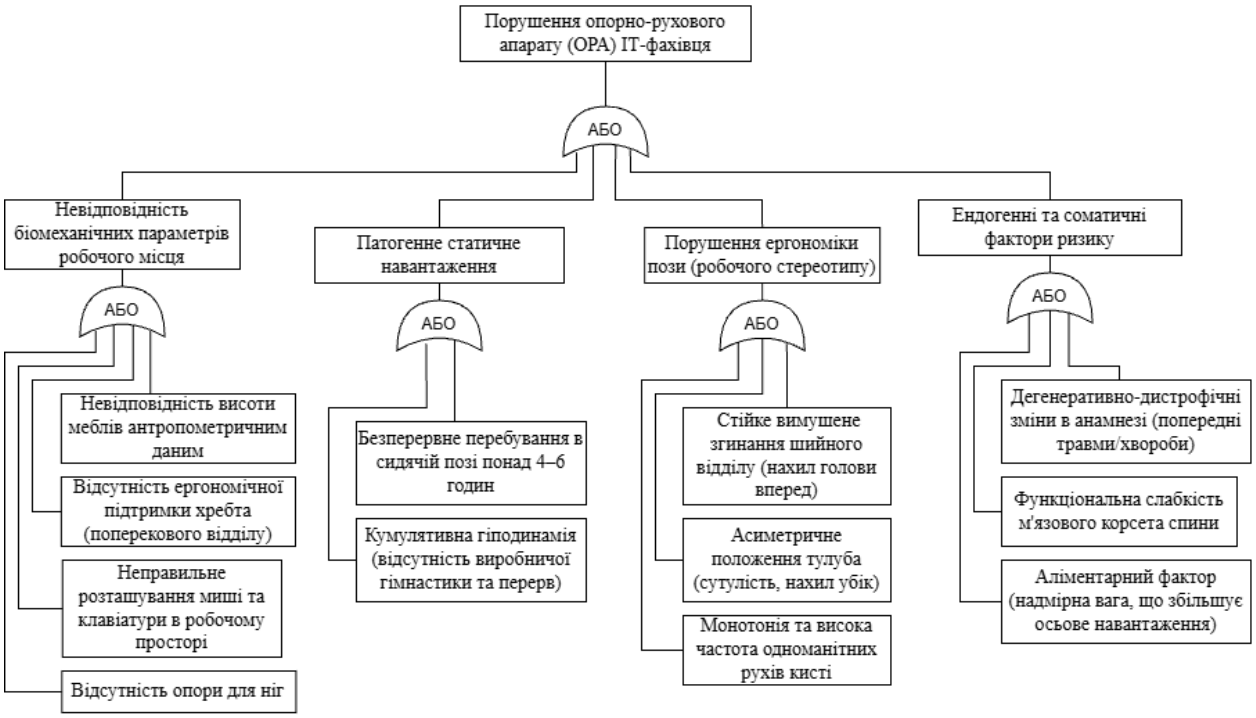

Рисунок 4.3 – Дерево відмов для аналізу ризику порушення опорно-рухового апарату ІТ-фахівця

Дерево відмов на рисунку 4.3 демонструє, що порушення опорно-рухового апарату формується через тривалу статичну роботу в неергономічних умовах та недостатню рухову активність. Ключовими причинами є неправильна посадка, відсутність підтримки спини та відсутність регулярних розминок. Ефективними заходами протидії є ергономічна організація робочого місця та виконання вправ для спини і рук кожні 1–2 години.

На основі проведеного матричного аналізу професійних ризиків (табл. 4.2) та деталізації причин за допомогою дерева відмов, було сформовано комплексні заходи щодо зниження або усунення виявлених небезпек. Перелік цих заходів та очікуваний ефект від їх реалізації наведено в таблиці 4.3.

Таблиця 4.3 – Заходи щодо зниження ризиків

№	Небезпека	Запропонований захід	Очікуваний результат
1	Погіршення зору	Дотримання правила 20-20-20, правильне розміщення монітора, якісне освітлення робочої зони	Зменшення втоми очей, збереження гостроти зору
2	Психоемоційне перенапруження	Регламентовані перерви, планування робочого часу, техніки управління стресом	Зниження рівня стресу, запобігання емоційному вигоранню
3	Порушення опорно-рухового апарату	Ергономічна організація робочого місця, вправи для спини та кистей кожні 1–2 години	Профілактика остеохондрозу та тунельного синдрому
4	Електрична небезпека	Використання справних подовжувачів, перевірка заземлення, пристроїв захисного вимкнення (ПЗВ)	Значне зниження ймовірності ураження струмом
5	Пожежна небезпека	Регулярна перевірка електрообладнання, наявність вогнегасника, знання правил евакуації	Мінімізація ризику пожежі та швидка реакція у разі небезпеки
6	Надзвичайна ситуація (Воєнна загроза)	Підготовка укриття, план дій при сигналі тривоги, резервне джерело живлення	Збереження життя та здоров’я в умовах надзвичайних ситуацій

Запропоновані заходи є комплексними і поєднують технічні, організаційні та гігієнічні рішення. Їх впровадження дозволить суттєво знизити рівень професійних ризиків і створити комфортні та безпечні умови для тривалої роботи за комп’ютером.

Таким чином, проведене дослідження ризиків підтвердило наявність характерних для ІТ-сфери небезпек і дозволило розробити конкретні рекомендації щодо їх попередження. Реалізація цих заходів сприятиме збереженню здоров’я розробника та підвищенню ефективності його праці.

Висновки до розділу

Забезпечення охорони праці є необхідною складовою сучасної професійної діяльності, особливо в сфері інформаційних технологій, де тривала робота за комп’ютером створює специфічні ризики для здоров’я людини. У розділі розглянуто організаційно-правові засади забезпечення

безпеки праці в Україні, проаналізовано умови робочого місця розробника та виявлено основні потенційні небезпеки, характерні для ІТ-сфери.

Аналіз показує, що на робочому місці розробника в умовах домашнього приміщення найбільш значущими є психофізіологічні чинники (зорове та психоемоційне перенапруження, статичне м’язове навантаження), а також фізичні небезпеки (ризик ураження електричним струмом, пожежна небезпека через перевантаження мережі, нераціональне освітлення). У сучасних реаліях критичної актуальності набувають також специфічні ризики воєнного стану.

Проведена оцінка професійних ризиків за допомогою матричного методу дозволила встановити, що найвищий (неприпустимий) рівень небезпеки мають чотири фактори: погіршення зору, професійне вигорання, порушення опорно-рухового апарату та воєнна загроза. Для детального вивчення причинно-наслідкових зв'язків трьох найбільш критичних психофізіологічних ризиків було побудовано та проаналізовано відповідні «дерева відмов», що дозволило виявити їхні базові (ініціюючі) події.

На основі комплексного аналізу побудованих логіко-графічних схем («дерев відмов») та матричної оцінки було розроблено систему заходів безпеки. Вона включає ергономічну організацію робочого місця, впровадження регламентованих перерв за правилом 20-20-20, виконання комплексів фізичних вправ, контроль параметрів мікроклімату й освітлення, а також заходи з електробезпеки та цивільного захисту. Реалізація цих рішень дозволяє знизити професійні ризики до прийнятного рівня.

Таким чином, дотримання вимог охорони праці під час роботи з комп’ютерною технікою є важливим фактором не лише захисту здоров’я фахівців ІТ-сфери, але й забезпечення високої продуктивності та якості їхньої професійної діяльності в цілому.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено вебзастосунок Codity для вивчення основ програмування мовою Python із використанням елементів гейміфікації. Основною метою роботи було створення навчальної платформи, яка поєднує подання теоретичного матеріалу, виконання практичних завдань та систему мотивації користувача.

На початковому етапі було проведено аналіз предметної області та досліджено особливості сучасних освітніх платформ. Розглянуто класичні системи дистанційного навчання та гейміфіковані сервіси, визначено їх переваги й недоліки. Результати аналізу дозволили сформувати вимоги до майбутнього програмного продукту та визначити основні напрями його функціонального розвитку.

У процесі проєктування було визначено вхідні та вихідні дані системи, розроблено функціональну модель, спроєктовано структуру даних користувача та побудовано об’єктно-орієнтовану модель вебзастосунку. Також було розглянуто математичне та алгоритмічне забезпечення, необхідне для реалізації механізмів збереження прогресу, нарахування досвіду, визначення рівня користувача та функціонування системи віртуального компаньйона.

Для реалізації вебзастосунку було використано сучасні вебтехнології HTML, CSS, JavaScript та Tailwind CSS. У роботі обґрунтовано вибір інструментів розробки, визначено вимоги до технічного та програмного забезпечення, а також реалізовано структуру вебзастосунку, що складається з головної сторінки, сторінок курсів, навчальних модулів, уроків, профілю користувача та сторінки віртуального компаньйона.

У межах практичної реалізації створено систему реєстрації та авторизації користувачів, механізм збереження даних у локальному сховищі браузера, систему проходження навчальних курсів і модулів, а також

інтерактивні практичні завдання з можливістю виконання Python-коду безпосередньо у браузері за допомогою технології Pyodide. Також реалізовано механізми збереження навчального прогресу, нарахування досвіду та відображення статистики користувача.

Особливу увагу було приділено створенню системи віртуального компаньйона, яка стала ключовою особливістю розробленого вебзастосунку. Компаньйон реалізований у вигляді стилізованих змій Python та супроводжує користувача протягом усього процесу навчання. Реалізовано систему рівнів, характеристик ситості та щастя, різні стани персонажа, можливість взаємодії з ним, а також залежність його розвитку від активності користувача.

У результаті виконання роботи було створено функціональний вебзастосунок, який забезпечує вивчення основ програмування мовою Python, виконання практичних завдань, відстеження особистого прогресу та взаємодію з віртуальним компаньйоном. Розроблена система демонструє можливість ефективного поєднання освітніх технологій та механізмів гейміфікації для підвищення мотивації користувачів до навчання та може бути використана як основа для подальшого розвитку повноцінної багатокурсової навчальної платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bers M.U., Strawhacker A., Sullivan A. The State of the Field of Computational Thinking in Early Childhood Education. OECD Education Working Papers, No. 274, OECD Publishing, Paris, 2022 [Електронний ресурс] // Режим доступу: OECD Education Working Papers

2. Wooclap. Online learning statistics: Key trends and industry insights, 2026 [Електронний ресурс] // Режим доступу: https://www.wooclap.com/en/blog/online-learning-statistics/

3. Evaluating the Difficulties in Programming Learning: Insights from Polytechnic Students / International Journal of Academic Research in Progressive Education and Development, Vol 14, 2025 [Електронний ресурс] // Режим доступу: Open Access Journal

4. Talken.cloud. Гейміфікація. Використання елементів гри в процесі навчання. [Електронний ресурс] // Режим доступу: https://talken.cloud/blog/gamification-using-game-elements-in-the-learning-process/

5. Головна сторінка освітньої платформи Prometheus [Електронний ресурс] // Режим доступу: https://prometheus.org.ua/

6. Головна сторінка вебдодатку Duolingo [Електронний ресурс] // Режим доступу: https://www.duolingo.com/

7. Головна сторінка вебдодатку Tynker [Електронний ресурс] // Режим доступу: https://www.tynker.com/

8. Nanchu. Tynker [Електронний ресурс] // Режим доступу: https://www.nanchu.me/work/tynker

9. Tailwind CSS [Електронний ресурс] // Режим доступу: https://tailwindcss.com/

10. Pyodide [Електронний ресурс] // Режим доступу: https://pyodide.org/en/stable/

11. Visual Studio Code [Електронний ресурс] // Режим доступу: https://code.visualstudio.com/

12. W3Schools Python Tutorial [Електронний ресурс] // Режим доступу: https://www.w3schools.com/python/default.asp

13. Закон України «Про охорону праці» [Електронний ресурс] / Офіційний сайт Верховної Ради України. // Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12

14. Бедрій Я.І. Основи охорони праці користувачів персональних комп'ютерів / Я.І. Бедрій : навч. посіб. – Тернопіль. : Навчальна книга – Богдан, 2014. – 144 с.

15. Кодекс законів про працю України від 10.12.1971 № 322-VIII [Електронний ресурс] / Офіційний сайт Верховної Ради України. // Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08

16. Державні санітарні норми і правила при роботі з джерелами електромагнітних полів : ДСанПіН 3.3.6.096-2002. – [Чинний від 2002-12-18]. – Київ : МОЗ, 2002. [Електронний ресурс]. / Офіційний сайт Верховної Ради України. // Режим доступу: https://zakon.rada.gov.ua/laws/show/z0203-03

17. Наказ Міністерства соціальної політики України від 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» (НПАОП 0.00-7.15-18) [Електронний ресурс]. / Офіційний сайт Верховної Ради України. // Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18

18. Санітарні норми мікроклімату виробничих приміщень: ДСН 3.3.6.042-99. – [Чинний від 1999-12-01]. – Київ : МОЗ, 1999. [Електронний ресурс]. / Офіційний сайт Верховної Ради України. // Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99

19. Природне і штучне освітлення: ДБН В.2.5-28:2018. – [Чинний від 2019-03-01]. – Київ : ДП «НДІ БК», 2018. // Режим доступу: https://e-construction.gov.ua/files-token/12cd5a14ab489b89ec310d103b987163

20. Системи управління охороною здоров’я та безпекою праці. Вимоги та настанови щодо застосування : ДСТУ ISO 45001:2019. – [Чинний від 2021-01-01]. – Київ : ДП «УкрНДНЦ», 2019. // Режим доступу: https://zakon.isu.net.ua/sites/default/files/normdocs/dstu_iso_45001_2019.pdf

21. Кодекс цивільного захисту України [Електронний ресурс] / Офіційний сайт Верховної Ради України. // Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17

22. Правила улаштування електроустановок (ПУЕ). – Х. : Вид-во «Форт», 2017. // Режим доступу: Правила улаштування електроустановок