**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**Факультет комп'ютерних наук, фізики та математики**

**Кафедра комп'ютерних наук та програмної інженерії**

# РОЗРОБКА ОСВІТНЬОГО ВЕБЗАСТОСУНКУ З ГЕЙМІФІКАЦІЄЮ ДЛЯ ВИВЧЕННЯ АЛГОРИТМІЗАЦІЇ

**Кваліфікаційна робота (проєкт)**

на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу 12-431 групи
Спеціальності: 122 Комп'ютерні науки
Освітньо-професійної програми: «Комп'ютерні науки»
Яковчук Андрій Миколайович

Керівник: кандидатка технічних наук, доцентка кафедри комп'ютерних наук та програмної інженерії
Шишко Людмила Станіславівна

Рецензент: кандидатка педагогічних наук, доцентка кафедри загальнофахової підготовки та морської безпеки Херсонської державної морської академії
Зайцева Т.В.

Івано-Франківськ 2026

# ЗМІСТ

# ВСТУП

## Актуальність теми

Розробка освітнього вебзастосунку з елементами гейміфікації для вивчення алгоритмізації є своєчасною відповіддю на виклики сучасної ІТ-освіти. В умовах стрімкого зростання попиту на кваліфікованих розробників виникає потреба в оптимізації навчального процесу, особливо у сфері фундаментальних дисциплін.

Актуальність проєкту зумовлена наступними факторами:

1. Подолання когнітивних бар'єрів. Вивчення алгоритмів та структур даних традиційно характеризується високим порогом входження через абстрактність матеріалу. Це часто призводить до зниження мотивації студентів та втрати інтересу до спеціальності на ранніх етапах. Впровадження гейміфікації дозволяє трансформувати складну рутину в інтерактивний процес, підвищуючи емоційну залученість користувачів.
2. Зміна освітньої парадигми. Сучасна освіта переходить від пасивного споживання інформації до моделей активного навчання (Active Learning). Вебзастосунок забезпечує миттєвий зворотний зв'язок та візуалізацію процесів виконання коду, що є критично важливим для глибокого розуміння принципів роботи алгоритмів.
3. Підвищення ефективності засвоєння знань. Використання ігрових механік (системи досягнень, рейтинги, рівні складності) у поєднанні з практичними завданнями сприяє кращому утриманню інформації та формуванню стійких професійних навичок.

## Мета і завдання дослідження

**Метою кваліфікаційної роботи** є розробка інтерактивного гейміфікованого веб-застосунку для вивчення алгоритмів та структур даних, який підвищує мотивацію, залученість та ефективність навчання

користувачів завдяки використанню ігрових механік (елементів стратегічних ігор, управління ресурсами та соціальної конкуренції).

Для досягнення поставленої мети було сформульовано та вирішено такі **завдання**:

1. Проаналізувати існуючі освітні платформи для вивчення програмування та дослідити їхні недоліки в контексті утримання уваги студентів.
2. Дослідити теоретичні засади гейміфікації в освіті та обґрунтувати вибір моделі гри-стратегії (збирання ресурсів, побудова бази, мапа місій) для навчального процесу.
3. Спроєктувати архітектуру клієнт-серверного веб-застосунку та розробити оптимальну структуру реляційної бази даних для збереження ігрового прогресу користувачів.
4. Розробити серверну частину (backend) з використанням мови програмування Python та фреймворку Django для реалізації бізнес-логіки (перевірка завдань, нарахування ігрової валюти, система обмежень у часі).
5. Створити адаптивний та інтерактивний користувацький інтерфейс (frontend) із застосуванням сучасних веб-технологій (HTML5, CSS3, Bootstrap), стилізований під ігрове середовище.
6. Провести тестування розробленого програмного продукту та оцінити його працездатність.

**Об’єктом дослідження** є процес дистанційного навчання та закріплення навичок з алгоритмізації та програмування.

**Предметом дослідження** є програмні засоби, веб-технології та методи гейміфікації, що застосовуються для створення освітніх інформаційних систем.

**Методи дослідження**

При виконанні роботи були використані наступні методи:

1. **Аналіз науково-технічної літератури та джерел Інтернет** для дослідження сучасних підходів до вивчення алгоритмізації, визначення проблем у традиційному навчанні та обґрунтування доцільності впровадження гейміфікації.
2. **Порівняльний аналіз** для розгляду існуючих аналогів (освітніх платформ, таких як LeetCode, Codewars, Duolingo), виявлення їхніх переваг та недоліків, що дозволило сформувати унікальні вимоги до розроблюваного застосунку.
3. **Методи системного аналізу** для формалізації предметної області, визначення функціональних вимог до системи та побудови логіки взаємодії користувача з навчальним контентом.
4. **Моделювання (з використанням мови UML)** для проєктування архітектури вебзастосунку.
5. Було побудовано діаграми варіантів використання (Use Case), класів та послідовностей для візуалізації структури системи перед етапом програмування.
6. **Об’єктно-орієнтований підхід** для програмної реалізації модулів системи, зокрема, створення класів для різних типів алгоритмів та ігрових елементів (користувач, досягнення, рівень).
7. **Метод прототипування** для розробки інтерфейсу користувача (UI/UX), що дозволило перевірити зручність візуалізації алгоритмів та ігрових механік до початку написання основного коду.
8. **Експериментальний метод (тестування)** для перевірки коректності роботи реалізованих алгоритмів, тестування навантаження та валідації системи нарахування балів (гейміфікації).

**Основні аспекти практичної цінності роботи:**

1. Програмний засіб навчання. Розроблений вебзастосунок є готовим інструментом для підтримки лабораторних та практичних занять з дисциплін «Алгоритми та структури даних» і «Основи програмування». Він дозволяє студентам візуально спостерігати за роботою алгоритмів, що значно спрощує розуміння складних абстракцій.
2. Автоматизація оцінювання. Система автоматичної перевірки коду звільняє викладача від рутинної роботи з перевірки типових завдань, дозволяючи зосередитися на допомозі студентам зі складними питаннями та архітектурою рішень.
3. Підвищення мотивації. Інтегровані елементи гейміфікації (рейтинги, досягнення) створюють конкурентне середовище, що стимулює студентів виконувати більше практичних завдань та підвищує загальну успішність групи.
4. Універсальність та доступність. Веб-орієнтована архітектура забезпечує кросплатформеність (доступ з будь-якого пристрою без встановлення ПЗ) та масштабованість, що дозволяє легко додавати нові модулі з алгоритмами або мовами програмування.

# РОЗДІЛ 1.
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1. Роль алгоритмізації у підготовці фахівців з комп'ютерних наук

Алгоритмізація та проєктування структур даних є фундаментальною складовою професійної підготовки фахівців у галузі комп'ютерних наук. Це той базис, що відрізняє професійного інженера-програміста (Software Engineer) від спеціаліста, який володіє лише синтаксисом мови програмування (Coder).

Значення глибокого розуміння алгоритмів у формуванні кваліфікованого спеціаліста можна розглядати у трьох ключових аспектах:

1. Формування обчислювального мислення (Computational Thinking). Вивчення алгоритмів розвиває здатність до декомпозиції складних задач, абстрагування, розпізнавання патернів та побудови покрокових рішень. Це дозволяє майбутньому фахівцю не просто використовувати готові бібліотеки, а розуміти принципи їх роботи на низькому рівні та адаптувати їх під конкретні потреби проєкту.

2. Оптимізація та ефективність програмних рішень. У сучасному світі, де обсяги даних зростають експоненціально, критично важливим стає вміння оцінювати складність алгоритмів (аналіз часової та просторової складності). Розуміння того, як код впливає на ресурси системи, дозволяє розробнику створювати масштабовані вебзастосунки, які працюють швидко навіть при високих навантаженнях.

3. Конкурентоспроможність на ринку праці. Більшість провідних технологічних компаній під час технічних співбесід фокусуються саме на перевірці алгоритмічної підготовки кандидата. Здатність швидко вирішувати алгоритмічні задачі є індикатором високого рівня логічного

мислення та технічної ерудиції, що безпосередньо впливає на кар'єрні перспективи випускника.

4. Незважаючи на високу важливість, дисципліна «Алгоритми та структури даних» залишається однією з найскладніших для засвоєння студентами через високий рівень абстракції матеріалу. Саме тому пошук нових методів викладання, таких як використання інтерактивних вебзастосунків, є критично важливим завданням сучасної освіти.

### 1.2. Освітні вебзастосунки як засіб навчання програмування

Стрімкий розвиток інформаційних технологій зумовив трансформацію підходів до навчального процесу, зміщуючи акцент з традиційних лекційних форм на інтерактивні методи навчання. У цьому контексті освітні вебзастосунки (Web-based Educational Applications) стають ключовим інструментом для підготовки майбутніх програмістів. Перевага вебзастосунків над класичними методами навчання (підручниками, статичними презентаціями) полягає у їхній інтерактивності та доступності. Можна виділити наступні ключові характеристики, що роблять їх ефективним засобом навчання програмування:

1. Доступність та кросплатформеність. Вебзастосунки не потребують встановлення спеціалізованого програмного забезпечення (IDE, компіляторів) на локальний комп'ютер користувача. Навчальне середовище доступне через браузер з будь-якого пристрою та операційної системи. Це усуває технічні бар'єри («environment setup overhead»), дозволяючи студенту зосередитися безпосередньо на вивченні алгоритмів, а не на налаштуванні інструментів.

2. Реалізація концепції «Learning by Doing» (Навчання через дію). На відміну від пасивного сприйняття інформації під час читання, вебзастосунки спонукають користувача до активної дії написання коду.

Інтегровані редактори коду та онлайн-компілятори дозволяють виконувати практичні завдання в реальному часі.

3. Миттєвий зворотний зв'язок (Instant Feedback). Однією з найбільших проблем самостійного вивчення програмування є відсутність перевірки. Вебзастосунки автоматизують цей процес: система миттєво аналізує введений код, проводить модульне тестування і повідомляє про помилки або успішне проходження тесту. Це значно прискорює цикл навчання, оскільки студент одразу розуміє свої помилки і може їх виправити.

4. Візуалізація абстракцій. Для вивчення алгоритмів критично важливою є візуалізація. Вебтехнології (HTML5, Canvas, WebGL) дозволяють створювати динамічні моделі, де користувач може покроково спостерігати за виконанням алгоритму (наприклад, переміщення елементів масиву при сортуванні або обхід вузлів графа).


Рис 1.1. Ключові характеристики освітніх вебзастосунків для навчання програмування.

### 1.3. Гейміфікація в освітніх системах: основні підходи

У сучасному науковому дискурсі під гейміфікацією розуміють використання ігрових елементів і методів проєктування ігор у неігровому контексті. В освітній сфері цей підхід набув значного поширення як засіб підвищення ефективності навчального процесу через вплив на мотиваційну сферу студента. Основна ідея гейміфікації полягає не у створенні повноцінної гри, а у інтеграції окремих ігрових механік у традиційний процес навчання для зміни поведінкових патернів користувачів.

Фундаментальним підходом до впровадження гейміфікації є використання моделі MDA (Mechanics, Dynamics, Aesthetics), яка описує взаємозв'язок між компонентами системи та емоційним відгуком користувача. На рівні механіки у навчальних вебзастосунках найчастіше застосовується тріада PBL (Points, Badges, Leaderboards). Бали (Points) слугують індикатором прогресу та забезпечують миттєвий зворотний зв'язок, що є критично важливим при вивченні точних дисциплін. Вони дозволяють студенту миттєво оцінити правильність виконання алгоритмічної задачі, не чекаючи перевірки викладачем.

Система бейджів (Badges) або віртуальних нагород виконує функцію візуалізації досягнень та компетенцій. З психологічної точки зору, отримання нагороди за вирішення складної задачі стимулює викид дофаміну, що закріплює позитивне ставлення до навчання. Лідерборди (Leaderboards) вводять соціальний аспект конкуренції, дозволяючи студентам порівнювати свої результати з іншими учасниками. Такий підхід базується на теорії соціального порівняння, згідно з якою індивіди схильні оцінювати власні успіхи через призму досягнень оточення.

Важливим аспектом гейміфікації є також створення стану «потоку» (Flow) психічного стану, в якому людина повністю занурена в діяльність. Для досягнення цього стану навчальні завдання повинні бути

чітко структуровані за рівнем складності: від елементарних до просунутих. Це дозволяє утримувати студента в коридорі між нудьгою (занадто прості задачі) та тривогою (занадто складні), що забезпечує максимальну залученість у процес вивчення алгоритмів.

Таким чином, гейміфікація виступає не як розважальний додаток, а як потужний дидактичний інструмент, здатний трансформувати рутинний процес написання коду в інтерактивну та вмотивовану діяльність, що сприяє глибшому засвоєнню матеріалу.

Таблиця 1.1.

Порівняльний аналіз розроблюваного застосунку з існуючими аналогами

| **Критерій порівняння** | **Розроблюваний застосунок («Clash of Python»)** | **CodeCombat** | **Codewars** | **LeetCode** |
|---|---|---|---|---|
| **Тип гейміфікації** | Стратегія (розвиток інфраструктури, бази) | RPG (керування персонажем у лабіринті) | Система рангів (бойові мистецтва: Кю / Дан) | Змагальна (рейтинги, турніри, бейджі) |
| **Віртуальна економіка** | **Повноцінна** (Золото, Еліксир, інвентар | Присутня (кристали для купівлі екіпірування | Відсутня (лише бали репутації) | Базова (монети для оформлен |
| **Візуальний інтерфейс** | **Ізометричний 3D-рендер** (Interactive Background Layering) | 2D графіка (вид зверху) | Текстовий (стандартний веб-інтерфейс) | Текстовий (стандартний веб-інтерфейс) |
| **Цільова аудиторія** | Студенти ЗВО, початківці | Школярі молодших та середніх | Досвідчені розробники (Middle/Sen | Спеціалісти, що готуються |

| | | | | |
|---|---|---|---|---|
| | та мідл-розробники | класів | ior) | до технічних співбесід |
| **Фокус навчання** | Алгоритмізація, синтаксис Python, структури даних | Базовий синтаксис, проста логіка | Оптимізація коду, рефакторинг | Складні алгоритми (FAANG рівень) |
| **Архітектура клієнта** | Легка (серверний рендеринг Django + CSS) | Важка (завантаження ігрового рушія в браузері) | Легка (SPA) | Легка (SPA) |

### 1.4. Аналіз існуючих рішень для вивчення алгоритмізації

Ринок програмного забезпечення для вивчення програмування на сьогоднішній день представлений широким спектром платформ, які можна класифікувати за цільовою аудиторією та методологією навчання. Проте детальний аналіз провідних рішень дозволяє виявити суттєві обмеження, що перешкоджають їх ефективному використанню в академічному середовищі для початківців.

Одним із найпопулярніших ресурсів є платформа LeetCode, яка фактично стала галузевим стандартом для підготовки до технічних співбесід.

Її беззаперечною перевагою є величезна база алгоритмічних задач, категоризованих за темами та складністю. Однак, LeetCode орієнтований переважно на досвідчених фахівців або студентів старших курсів, які вже мають фундаментальну теоретичну базу. Для початківців високий поріг входження та відсутність структурованої теоретичної частини створюють надмірне когнітивне навантаження, що часто призводить до демотивації. Крім того, інтерфейс платформи є суто утилітарним і

позбавленим ігрових елементів, що знижує залученість користувачів у довгостроковій перспективі.

Альтернативним підходом характеризується платформа Codewars, яка активно використовує елементи гейміфікації. Система рангів (кю/дан), запозичена зі східних єдиноборств, та змагальний елемент ефективно стимулюють інтерес до навчання.

Проте контент на цій платформі генерується спільнотою (Community-driven), що призводить до нерівномірності складності завдань та відсутності єдиного методичного підходу. Часто "найкращими" рішеннями визнаються не найбільш зрозумілі чи оптимальні з точки зору алгоритмів, а найбільш короткі ("code golf"), що формує у студентів хибне уявлення про якість коду та "best practices" у розробці.

Також варто відзначити платформи на кшталт CodeCombat, які перетворюють процес написання коду на повноцінну рольову гру (RPG). Хоча такий підхід є ідеальним для школярів та дітей молодшого віку, для студентів закладів вищої освіти він є занадто повільним та абстрагованим від реальних задач професійної розробки. Візуальна складова тут часто домінує над змістом, що не дозволяє глибоко зануритися у специфіку складних структур даних.

Таким чином, проведений аналіз свідчить про наявність вільної ніші для розробки спеціалізованого вебзастосунку. Існує об'єктивна потреба у системі, яка б поєднувала академічну ґрунтовність теоретичного матеріалу, властиву університетським курсам, з інтерактивністю та мотиваційними механіками сучасних ігрових платформ, при цьому залишаючись доступною для початківців.

### 1.5. Формування вимог до освітнього вебзастосунку

На основі проведеного аналізу предметної області, існуючих аналогів та особливостей гейміфікації навчального процесу, було сформовано перелік ключових вимог до проєктованої системи. Розроблюваний вебзастосунок має поєднувати в собі функціональність середовища розробки (IDE) та навчальної платформи, забезпечуючи при цьому високий рівень інтерактивності та мотивації користувачів.

Для досягнення поставленої мети система повинна відповідати наступним функціональним вимогам:

1. Підсистема автентифікації та профілів: Система повинна забезпечувати реєстрацію та авторизацію користувачів, збереження їхніх персональних даних, налаштувань та історії проходження курсів.

2. Інтерактивне середовище виконання коду: Вебзастосунок має містити вбудований редактор коду з підсвічуванням синтаксису та можливістю компіляції/інтерпретації програмного коду безпосередньо у браузері (Client-side або Server-side execution).

3. Модуль навчання та перевірки знань: Система повинна надавати теоретичний матеріал у структурованому вигляді та забезпечувати автоматичну перевірку практичних завдань (Unit-тестування рішень користувача) з миттєвим виведенням результатів.

4. Реалізація гейміфікованих механік: Необхідна наявність системи нарахування балів за успішне виконання завдань, візуалізація прогресу (Progress Bar), отримання віртуальних нагород (бейджів) та формування глобального рейтингу користувачів (Leaderboard).

Окрім функціональних аспектів, критично важливими є нефункціональні вимоги, що визначають якість роботи системи:

1. Продуктивність (Performance): Час відгуку системи при компіляції коду не повинен перевищувати прийнятних для користувача

меж (наприклад, 2–3 секунди), навіть при одночасній роботі великої кількості студентів.

2. Безпека (Security): Оскільки система передбачає виконання коду, написаного користувачем, необхідно реалізувати механізми ізоляції (Sandboxing) для запобігання виконанню шкідливих скриптів на сервері. Також має бути забезпечено захист персональних даних та паролів (хешування).

3. Зручність використання (Usability): Інтерфейс користувача повинен бути інтуїтивно зрозумілим, адаптивним (Responsive Design) для коректного відображення на мобільних пристроях та планшетах, а також відповідати сучасним стандартам UX-дизайну.

# РОЗДІЛ 2.
# ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

## 2.1. Загальна архітектура та вибір технологій розробки

При виборі інструментальних засобів для реалізації гейміфікованого вебзастосунку ключовими критеріями були: швидкість розробки прототипу, масштабованість, надійність, безпека користувацьких даних та наявність розвиненої екосистеми бібліотек. На основі глибокого порівняльного аналізу сучасних рішень для веб-розробки було обрано високорівневу мову програмування Python та повнофункціональний вебфреймворк Django.

Архітектура розроблюваної системи базується на класичному для Django патерні **MVT (Model-View-Template)**, який є адаптацією загальноприйнятого MVC (Model-View-Controller). Цей підхід дозволяє чітко розділити проєкт на три незалежні рівні:

1. **Model (Модель):** Відповідає за схему даних та бізнес-логіку взаємодії з базою даних.
2. **View (Відображення):** Виконує роль контролера, приймаючи HTTP-запити від користувача, обробляючи їх відповідно до правил гри та повертаючи результат.
3. **Template (Шаблон):** Відповідає за формування кінцевого HTML-документа, який побачить користувач (інтерфейс гри).

**Детальне обґрунтування вибору технологічного стеку:**

1. **Мова програмування Python:** Python є де-факто світовим стандартом у сфері навчання програмуванню та алгоритмізації завдяки своєму лаконічному, читабельному синтаксису. Використання Python на серверній стороні (Backend) забезпечує високу продуктивність написання коду. Крім того, це відкриває можливості для майбутнього масштабування проєкту: наприклад, легкої інтеграції модулів для автоматичного синтаксичного аналізу

або виконання коду студентів у безпечному ізольованому середовищі (sandbox).

2. **Вебфреймворк Django:** Вибір Django зумовлений його архітектурною філософією «batteries included» (все необхідне в комплекті). На відміну від мікрофреймворків, Django надає готові, перевірені часом механізми для:
3. **Автентифікації та авторизації:** Вбудована система керування сесіями та хешування паролів забезпечує безпечну реєстрацію користувачів та гнучкий розподіл прав доступу (адміністратор/викладач та звичайний гравець).
4. **ORM (Object-Relational Mapping):** Потужний інструмент, що дозволяє працювати з базою даних, оперуючи виключно об'єктами Python. Це не лише значно пришвидшує розробку, уникаючи написання сирих SQL-запитів, але й автоматично екранує всі вхідні дані, забезпечуючи надійний захист від найпоширеніших загроз, таких як SQL-ін'єкції.
5. **Вбудована панель адміністрування:** Автоматично згенерований, зручний веб-інтерфейс дозволяє викладачам легко додавати нові ігрові задачі (мобів), редагувати їх складність та переглядати статистику студентів без необхідності втручання у вихідний код проєкту.
6. **Захист від кібератак:** Фреймворк "з коробки" надає захист від міжсайтової підробки запитів (CSRF) та міжсайтового скриптингу (XSS), що є критично важливим для освітнього середовища.
7. **СУБД SQLite / PostgreSQL:** На етапі проєктування, розробки та локального тестування використовується вбудована реляційна база даних SQLite, яка зберігає всі дані у єдиному файлі та не потребує складного налаштування сервера. Для розгортання в робочому середовищі (production) та обслуговування великої кількості

одночасних підключень студентів, архітектура Django дозволяє змінити лише один файл налаштувань для безшовної міграції на потужну СУБД відкритого коду PostgreSQL.

8. **Клієнтська частина (Frontend):** Для створення інтерактивного та візуально привабливого інтерфейсу користувача використовуються стандарти HTML5 та CSS3. Адаптивність сторінок, що гарантує коректне відображення гри як на ПК, так і на мобільних пристроях, досягається завдяки підключенню фреймворку Bootstrap. Анімації та візуальні ефекти (наприклад, система динамічних хмар та ефекти зіткнення на мапі) реалізовані переважно засобами CSS, що мінімізує навантаження на браузер клієнта.

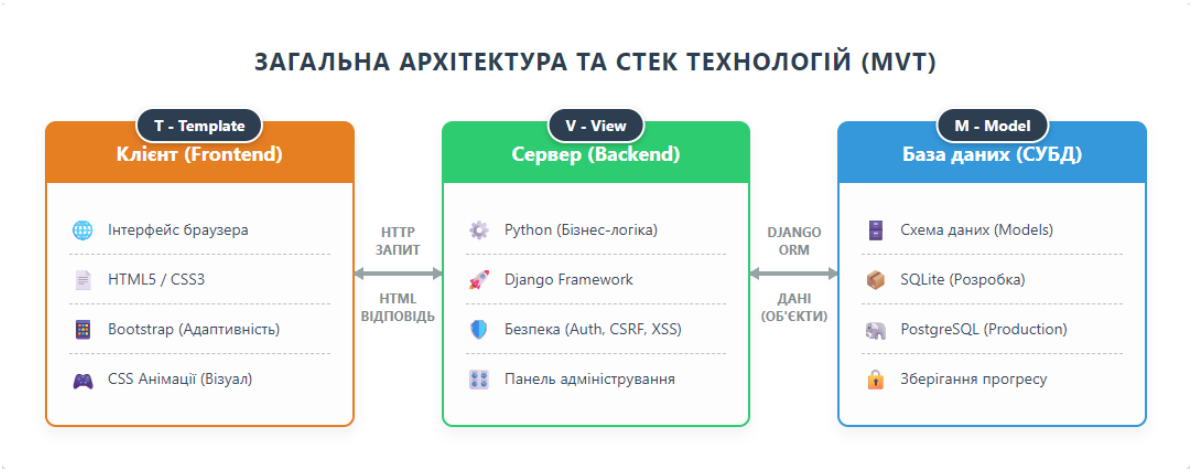


Рис 2.1. Архітектура системи та технологічний стек вебзастосунку.

### 2.2. Проєктування бази даних та моделей предметної області

Для забезпечення стабільної роботи гейміфікованої складової (системи економіки, управління прогресом та збереження історії взаємодії із завданнями) було ретельно спроєктовано реляційну схему бази даних. Завдяки технології Django ORM, структура даних описана у вигляді класів мови Python, що гарантує високий рівень нормалізації даних.

Основу предметної області системи складають дві кастомні сутності: ігровий профіль користувача (Profile) та навчальне завдання (Task).

**Сутність «Профіль гравця» (Profile)**

Для системи авторизації використовується стандартна, безпечна модель User від Django. Щоб розширити її додатковими ігровими атрибутами без втручання в ядро фреймворку, було створено модель Profile, яка пов'язана з моделлю User строгим зв'язком типу «один-до-одного» (models.OneToOneField). При видаленні користувача його профіль автоматично знищується завдяки правилу каскадного видалення (on_delete=models.CASCADE).

Модель Profile інкапсулює ключові атрибути віртуальної економіки:

1. gold (IntegerField) баланс віртуального золота, яке заробляється за правильні відповіді. Використовується як ліквідна валюта у внутрішньоігровому магазині (зокрема, для купівлі підказок).
2. elixir (IntegerField) баланс магічного еліксиру, необхідного для капітальних інвестицій гравця (покращення рівня віртуальної бази).
3. trophies (IntegerField) кількість кубків. Виконує роль рейтингових балів, що визначають позицію студента в загальній таблиці лідерів (Leaderboard), стимулюючи соціальну конкуренцію та прагнення до самовдосконалення.
4. town_hall_level (IntegerField) рівень розвитку головної будівлі («Ратуші»), що візуалізує загальний прогрес проходження курсу.
5. hints_count (IntegerField) інвентар гравця, що відображає кількість доступних магічних підказок.
6. last_attack_time (DateTimeField) системне поле для фіксації точного серверного часу (з урахуванням часового поясу) останньої спроби вирішення завдання. Цей атрибут є фундаментом для реалізації

механіки "кулдауну" (обмеження частоти атак).

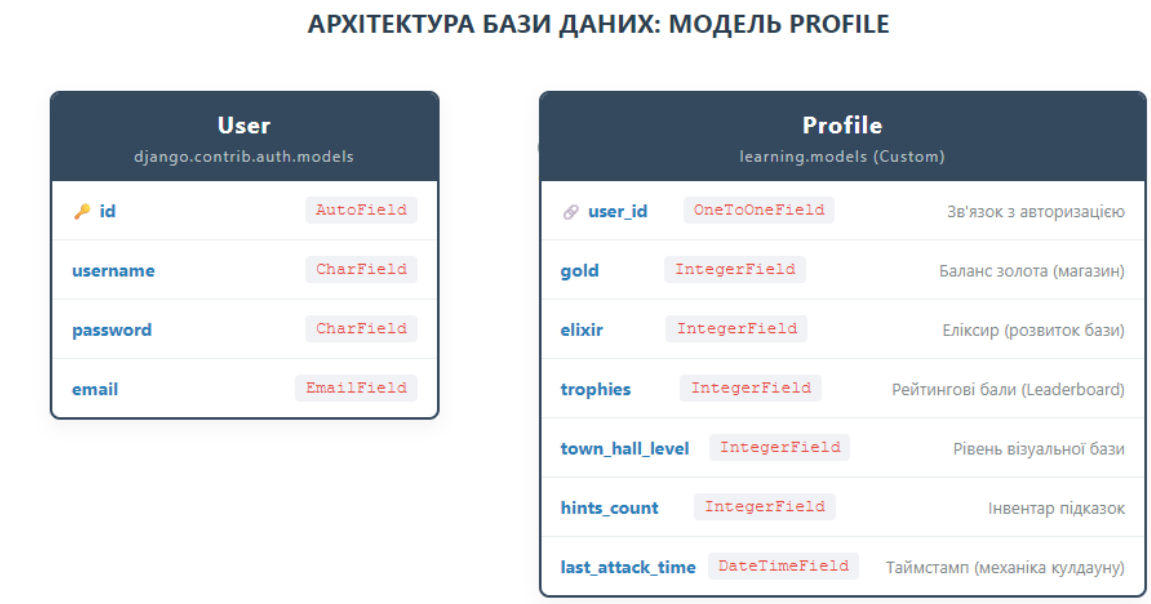


Рис.2.2. Структура моделі даних «Profile» та її зв'язок із системою авторизації Django.

**Сутність «Навчальне завдання» (Task)**

Ця модель репрезентує інтерактивні завдання, які метафорично стилізовані під ігрових супротивників (гоблінів, орків тощо). Модель включає:

1. title (CharField) назва завдання, що відображається в інтерфейсі мапи.
2. description (TextField) детальна алгоритмічна умова задачі, яку повинен проаналізувати студент.
3. correct_answer (CharField) еталонний рядок-відповідь, з яким сервер буде порівнювати введені користувачем дані.
4. Блок економічних нагород: gold_reward, elixir_reward, trophies_reward (IntegerField).

Ці параметри дозволяють викладачу гнучко налаштовувати баланс гри. Складніші завдання приносять експоненційно більший дохід, стимулюючи студентів не зупинятися на легких задачах.

### 2.3. Реалізація бізнес-логіки та ігрових механік

Керування взаємодією між діями користувача, станом бази даних та оновленням інтерфейсу зосереджено в модулі відображень (views.py). Саме тут запрограмовано ключові ігрові цикли застосунку.

**Механіка обробки завдань та система винагород**

Взаємодія студента із завданнями на "Мапі вторгнення" обробляється через безпечні POST-запити. Алгоритм бойової системи працює за наступним сценарієм:

1. Сервер отримує прихований ідентифікатор завдання (task_id) та введену користувачем відповідь (answer).
2. Відбувається процес санітизації даних: введена відповідь очищається від випадкових пробілів по краях (strip()) та переводиться в нижній регістр (lower()). Те ж саме відбувається з еталонною відповіддю з бази даних, що мінімізує ризик хибних спрацьовувань через друкарські помилки (наприклад, введення з великої літери).
3. **Сценарій перемоги:** У разі збігу відповідей ініціюється транзакція. До полів gold, elixir та trophies профілю користувача додаються відповідні винагороди з об'єкта завдання. Метод save() фіксує зміни в БД. Інтерфейс генерує позитивне повідомлення (toast-сповіщення).
4. **Сценарій поразки:** Ресурси не знімаються (щоб не демотивувати гравця на початкових етапах), проте користувач отримує повідомлення про поразку, яке спонукає переглянути підхід до вирішення алгоритму.

**Алгоритм тайм-менеджменту та захисту від підбору (Cooldown)**

Для забезпечення академічної доброчесності та запобігання атакам типу brute-force (сліпе вгадування відповідей за допомогою скриптів або швидкого клікання), розроблено механіку «втоми армії». Перед обробкою будь-якої відповіді система перевіряє значення

last_attack_time. Сервер обчислює різницю у секундах між поточним часом (timezone.now()) та попередньою атакою. Якщо інтервал менший за встановлену константу (наприклад, 30 секунд), виконання логіки переривається. Гравцю повертається інформаційне попередження з точним розрахунком часу, що залишився до відновлення можливості атакувати. Це штучно уповільнює процес і змушує студента інвестувати час в осмислення задачі.

**Внутрішньоігрова економіка:**

Магазин та прогрес Бази Економічний цикл гри замикається можливістю витрачати ресурси, що перетворює застосунок із простого тесту на повноцінну гру:

- **Купівля підказок (Товар за Золото):** Перевіряється умова profile.gold >= cost. Якщо баланс позитивний, золото списується, а лічильник hints_count зростає. Студент може використати підказку під час складного бою.
- **Покращення Ратуші (Інвестиція Еліксиру):** Вартість є динамічною та прогресивною: town_hall_level * 500. Алгоритм перевіряє наявність еліксиру. У разі успіху рівень бази підвищується. Візуалізація зростання рівня на головній сторінці працює як потужний психологічний якір, що утримує інтерес студента на довгостроковій перспективі.

**Соціальна конкуренція (Leaderboard)**

Для впровадження механік соціальної взаємодії та змагальності реалізовано сторінку «Топ Кланів». Формування рейтингу відбувається оптимізованим запитом до БД: Profile.objects.all().order_by('-trophies'). Префікс - забезпечує сортування профілів за спаданням кількості кубків. Дані передаються в шаблон, де динамічно генерується рейтингова таблиця з виділенням трійки лідерів, стимулюючи здорову конкуренцію в навчальній групі.

.

# РОЗДІЛ 3.
# ІНТЕРФЕЙС КОРИСТУВАЧА ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

## 3.1. Проєктування та стилізація інтерфейсу користувача (UI/UX)

Візуальна складова (інтерфейс користувача) відіграє критичну роль у гейміфікованих освітніх системах. Згідно з принципами User-Centered Design (дизайн, орієнтований на користувача), інтерфейс має не лише бути естетично привабливим, але й забезпечувати інтуїтивно зрозумілу навігацію, щоб студент не витрачав когнітивний ресурс на вивчення меню, а фокусувався на вирішенні алгоритмічних задач.

Дизайн вебзастосунку «Clash of Python» було спроєктовано з використанням візуальної мови популярних мобільних стратегій (зокрема жанру Tower Defense та Base Building). Для досягнення максимальної імерсивності (занурення) було застосовано наступні архітектурні UI-рішення:

1. **Типографіка:** Для заголовків, кнопок та ключових ігрових елементів інтегровано кастомний веб-шрифт «Lilita One» (через Google Fonts), який має характерні "мультяшні" та масивні контури, притаманні ігровій індустрії. Для блоків з детальним описом алгоритмічних завдань використано стандартні системні шрифти без зарубок (sans-serif) для забезпечення високої читабельності великих обсягів тексту.
2. **Кольорова психологія:** Палітра розділена на зони. На сторінці «Моя Фортеця» домінують світло-блакитні та зелені відтінки (кольори безпеки та розвитку). Натомість «Мапа вторгнення» виконана у темних, контрастних тонах із використанням текстур каменю та землі, що психологічно налаштовує студента на концентрацію та "бойовий" лад перед вирішенням задачі.

3. **Матеріальний дизайн карток:** Всі інтерактивні елементи (картки будівель, табори гоблінів) стилізовані під фізичні матеріали (камінь, дерево, залізо) за допомогою складних каскадних таблиць стилів. Застосовано множинні тіні box-shadow (внутрішні та зовнішні), CSS-градієнти linear-gradient та радіальні градієнти для створення ілюзії металевих "заклепок" на кутах інтерфейсу.

**Реалізація динамічних візуальних ефектів (CSS3 Animations)**

Щоб уникнути перевантаження клієнтського браузера важкими JavaScript-бібліотеками обробки графіки (наприклад, Three.js чи WebGL), візуальна динаміка реалізована виключно на апаратно-прискореному рівні CSS3 (правила @keyframes).

1. **Ефект багатошарового паралаксу:** На головній сторінці реалізовано систему з п'яти шарів напівпрозорих хмар, згенерованих через псевдоелементи ::before та ::after. Кожен шар має власну швидкість анімації та рівень масштабування (transform: scale), що створює глибокий 3D-ефект рухомого неба.
2. **Бойові ефекти:** На мапі завдань імплементовано анімацію схрещення двох напівпрозорих мечів на задньому плані. Використання властивості transform-origin: bottom center дозволило змістити центр обертання об'єктів на руків'я, імітуючи реалістичну фізику замаху та удару клинками з генерацією ефекту іскри в момент зіткнення.

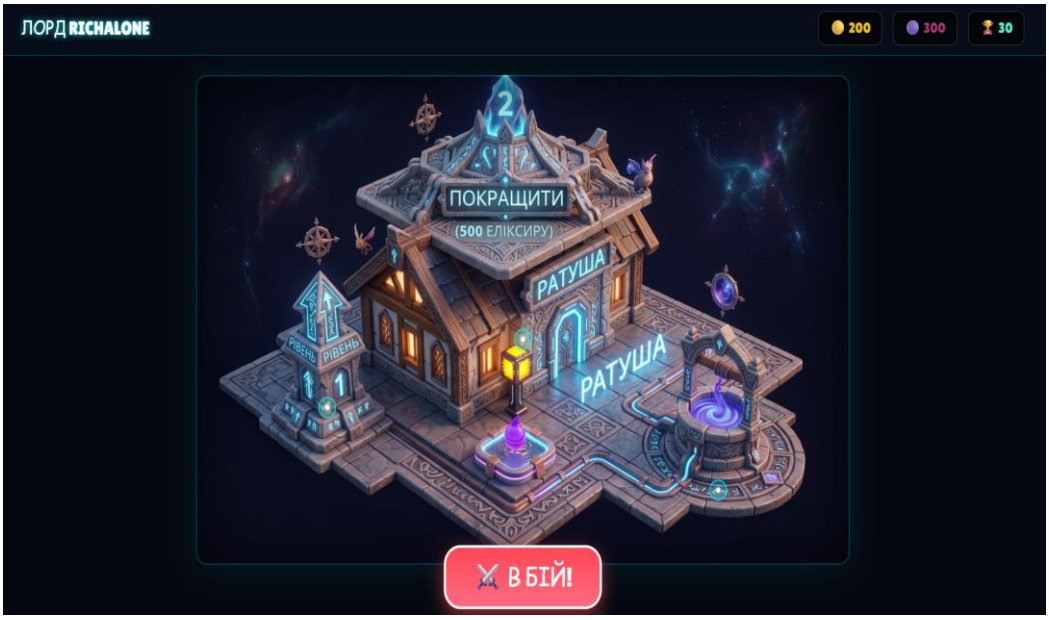


Рис. 3.1. Графічний інтерфейс головного ігрового простору програми веб-сторінка «Головна база»

### 3.2. Адаптивність та кросбраузерність системи (Responsive Design)

Враховуючи сучасні тенденції до мобільного навчання (m-learning) та концепцію "Mobile-First", критичною вимогою до вебзастосунку була його здатність коректно адаптуватися до будь-якої роздільної здатності екрана від широкоформатних моніторів до смартфонів.

Для розв'язання цього завдання було імплементовано популярний CSS-фреймворк Bootstrap 5. Його застосування базується на системі гнучких сіток (12-колонкова Flexbox-сітка).

1. Десктопна версія: На широких екранах (min-width: 992px, клас col-lg) ігрові елементи розташовуються горизонтально в кілька колонок, максимально використовуючи вільний простір.
2. Мобільна версія: При зменшенні ширини екрана (класи col-12, col-md) медіа-запити (Media Queries) автоматично перебудовують макет. Картки завдань та будівель шикуються в один вертикальний стовпець. Крім того, верхня навігаційна панель ресурсів оптимізується: другорядні елементи (наприклад, лічильник кубків)

приховуються класом d-none d-md-flex, залишаючи на екрані смартфона лише найважливіші показники (Золото та Еліксир). Всі кнопки мають збільшену область натискання, що відповідає стандартам touch-friendly інтерфейсів.

### 3.3. Безпека даних та валідація запитів

Оскільки система передбачає збереження персонального прогресу користувачів, архітектура застосунку включає кілька рівнів захисту від поширених веб-вразливостей:

1. Захист від CSRF-атак: Усі форми в грі (покращення бази, відправка відповідей, купівля товарів) захищені вбудованим механізмом Django. Це унеможливлює міжсайтову підробку запитів, гарантуючи, що дія ініційована саме дійсним користувачем, а не стороннім скриптом.
2. Захист від SQL-ін'єкцій: Завдяки використанню ORM-системи, всі дані, введені користувачем у поле відповіді на завдання, автоматично екрануються перед формуванням запиту до бази даних SQLite/PostgreSQL.
3. Санітизація вводу (Input Validation): На рівні бекенду реалізовано алгоритм нормалізації відповідей. Коли студент вводить код або відповідь, сервер застосовує методи очищення рядків (видалення провідних та кінцевих пробілів) та приведення до єдиного регістру. Це не лише підвищує стійкість системи, але й покращує користувацький досвід (UX), оскільки студент не отримує системну помилку через випадково натиснутий пробіл.

### 3.4. Комплексне тестування програмного продукту

1.На завершальному етапі розробки (життєвого циклу ПЗ) було проведено комплексне тестування функціоналу гейміфікованої

платформи для забезпечення якості (QA). Процес складався з перевірки трьох ключових векторів:

1. Функціональне тестування ігрової економіки: Були змодельовані сценарії перевірки транзакцій. Тестування довело, що:
2. При спробі натиснути кнопку "Покращити Ратушу" з балансом еліксиру нижче необхідної вартості, сервер коректно відхиляє транзакцію, рівень не підвищується, а користувач отримує стилізоване Toast-сповіщення про нестачу ресурсів.
3. При успішному вирішенні завдання ресурси (золото, еліксир, кубки) нараховуються суворо відповідно до констант, заданих у моделі Task. Дубляж нарахувань при подвійному кліку заблоковано архітектурою контролера.

2. Тестування бойової системи та тайм-менеджменту: Сценарій тестування захисту від автоматичного підбору відповідей (Brute-force) включав відправку серії швидких POST-запитів до сервера. Результати підтвердили працездатність механіки «Cooldown»: сервер успішно ідентифікує час попередньої атаки через поле last_attack_time і повертає помилку "Вам потрібно відпочити ще X секунд" до моменту повного спливання таймера.

3. Тестування UI та Юзабіліті: Кросбраузерне тестування проводилось у сучасних веб-оглядачах (Google Chrome, Mozilla Firefox, Safari). Анімації, реалізовані через CSS3, працюють плавно без падіння частоти кадрів (FPS). Таблиця лідерів (Leaderboard) коректно генерується на основі поточного стану бази даних, миттєво реагуючи на зміну кількості кубків у профілях користувачів.

Результати всіх етапів тестування підтверджують високу надійність архітектури. Вебзастосунок функціонує стабільно, не містить критичних помилок (багів) і повністю готовий до дослідної експлуатації та впровадження в реальний освітній процес.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно розв’язано актуальну науково-практичну задачу розроблено інтерактивний гейміфікований вебзастосунок для вивчення алгоритмів та мови програмування Python, який поєднує освітній процес із механіками стратегічних відеоігор.

Практична значущість отриманих результатів полягає у готовності розробленої системи до реального використання в навчальному процесі. Вебзастосунок може успішно функціонувати як самостійна база для опанування алгоритмізації, а також слугувати додатковим дидактичним засобом для підвищення залученості студентів на практичних і лабораторних заняттях.

Відповідно до поставленої мети та завдань дослідження отримано наступні результати:

1. Проведено аналіз існуючих підходів до дистанційного навчання та обґрунтовано необхідність впровадження гейміфікації. Доведено, що використання економічної ігрової моделі (збирання ресурсів, покращення віртуальної бази, використання ігрових предметів) ефективніше утримує увагу студентів порівняно з традиційними системами тестування.
2. Спроєктовано клієнт-серверну архітектуру та реляційну базу даних. На базі технології ORM розроблено оптимізовані моделі даних: Profile (для збереження ігрової статистики, балансу ресурсів та контролю часу активності) та Task (для управління параметрами завдань та розмірами винагород).
3. Програмно реалізовано серверну частину (Backend) з використанням мови Python та вебфреймворку Django (архітектура MVT). Успішно імплементовано складну бізнес-логіку: алгоритм

перевірки відповідей з автоматичним нарахуванням віртуальної валюти (золото, еліксир, кубки), механіку внутрішньоігрового магазину та систему тайм-менеджменту (захист від brute-force атак шляхом встановлення обов'язкового інтервалу між спробами).

4. Розроблено адаптивний та інтерактивний інтерфейс користувача (Frontend). Завдяки поєднанню HTML5, CSS3 та фреймворку Bootstrap 5 створено стилізований ігровий простір («Моя Фортеця» та «Мапа Вторгнення»). Без використання ресурсомістких JavaScript-бібліотек реалізовано високопродуктивні CSS-анімації (ефект паралаксу для багатошарових хмар, динамічні бойові ефекти зіткнення мечів), що забезпечило глибоке занурення в ігровий процес (імерсивність).
5. Реалізовано механізми соціальної конкуренції. Розроблено алгоритм формування глобальної таблиці лідерів у режимі реального часу, що ранжує користувачів за кількістю здобутих кубків і стимулює здорову змагальність.
6. Проведено комплексне тестування розробленого програмного продукту. Результати тестування підтвердили стабільність роботи бізнес-логіки, цілісність бази даних при обробці транзакцій та коректність відображення інтерфейсу на різних типах пристроїв.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. **Бхаргава А.** Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих. Київ : Фабула, 2021. 336 с.
2. **Вербах К., Хантер Д.** Втягуй і володарюй. Ігрове мислення на службі бізнесу та освіти. Київ : Наш формат, 2020. 224 с.
3. **Вірт Н.** Алгоритми і структури даних. Київ : Діалектика, 2017. 360 с.
4. **Дакетт Дж.** HTML та CSS. Розробка та дизайн веб-сайтів. Київ : Клуб Сімейного Дозвілля, 2020. 480 с.
5. **Доусон М.** Програмуємо на Python. Київ : О'Райллі, 2021. 416 с.
6. **Застосування гейміфікації в освітньому процесі** : методичний посібник / за ред. О. В. Мельника. Київ : Освіта, 2022. 180 с.
7. **Зікерманн Г., Ліндер Д.** Гейміфікація в бізнесі та освіті: як пробитися крізь шум і заволодіти увагою. Київ : Фабула, 2021. 272 с.
8. **Капп К. М.** Гейміфікація навчання: методи та стратегії інтерактивної освіти. Харків : Ранок, 2023. 310 с.
9. **Кіт Д.** HTML5 для веб-дизайнерів. Київ : Основи, 2019. 144 с.
10. **Кормен Т., Лейзерсон Ч., Рівест Р., Стайн К.** Алгоритми: побудова і аналіз. Київ : К.І.С., 2019. 1328 с.
11. **Лутц М.** Вивчаємо Python. 5-те вид. Харків : Фоліо, 2020. 1150 с.
12. **Макгонігал Д.** Реальність під питанням: Чому ігри роблять нас кращими і як вони можуть змінити світ. Київ : Наш формат, 2022. 384 с.
13. **Макфарланд Д.** CSS3: Вичерпне керівництво. Київ : Діалектика, 2021. 720 с.
14. **Мартін Р.** Чистий код. Створення і рефакторинг за допомогою гнучких методологій. Київ : Фабула, 2019. 464 с.
15. **Матіз Е.** Вивчаємо Python. Програмування ігор, візуалізація даних, веб-додатки. Київ : Наш формат, 2021. 520 с.
16. **Меліо А.** Django 4 на практиці. Створення веб-додатків на Python. Київ : Діалектика, 2023. 510 с.
17. **Ньюмен С.** Створення мікросервісів. Київ : К.І.С., 2021. 304 с.
18. **Персиваль Х.** Розробка через тестування на Python. Київ : Діалектика, 2022. 380 с.
19. **Седжвік Р., Вейн К.** Алгоритми. 4-те вид. Київ : Діалектика, 2020. 1056 с.

20. **Соммервілл І.** Інженерія програмного забезпечення. 10-те вид. Київ : Основи, 2018. 840 с.
21. **Фаулер М.** Архітектура корпоративних програмних додатків. Київ : Діалектика, 2020. 544 с.
22. **Фленеган Д.** JavaScript: Докладний посібник. 7-ме вид. Київ : Діалектика, 2022. 752 с.
23. **Фрімен Е., Робсон Е.** Вивчаємо HTML, XHTML і CSS. Київ : Фабула, 2021. 768 с.
24. **Чоу Ю-К.** Гейміфікація та поведінковий дизайн: фреймворк Octalysis. Київ : Наш формат, 2023. 290 с.
25. **MDN Web Docs: HTML, CSS, JavaScript** [Електронний ресурс]. URL: https://developer.mozilla.org/.
26. **Офіційна документація Django** [Електронний ресурс]. URL: https://docs.djangoproject.com/.
27. **Офіційна документація Docker** [Електронний ресурс]. URL: https://docs.docker.com/.
28. **Офіційна документація Python 3** [Електронний ресурс]. URL: https://docs.python.org/3/.
29. **Специфікація OpenAPI** [Електронний ресурс] // Swagger. URL: https://swagger.io/specification/.

# ДОДАТКИ

## Додаток А

## Оптимізація та реалізація графічного інтерфейсу користувача з елементами гейміфікації

В процесі розробки освітнього вебдодатка з елементами гейміфікації особливу увагу було приділено проєктуванню графічного інтерфейсу користувача (GUI). Одним із ключових завдань стала реалізація інтерактивної головної бази користувача (Ратуші), яка візуально відображає прогрес у вивченні алгоритмізації та розв'язанні практичних завдань на мові програмування Python.

Спочатку розглядався підхід створення повноцінних 3D-моделей виключно засобами CSS3 (CSS 3D Transforms) або з використанням спеціалізованих бібліотек WebGL. Однак для забезпечення максимальної продуктивності браузера на стороні клієнта та досягнення високої візуальної якості рівня AAA-проєктів було обрано стратегію Interactive Background Layering (інтерактивне нашарування фону).

Суть даного архітектурного рішення полягає в наступному:

1. Попередній рендеринг (Pre-rendering): Складні високополігональні ізометричні зображення етапів розвитку бази (від початкової дерев'яної ратуші до високорівневої цитаделі з магічними колодязями еліксиру та обелісками) рендеряться заздалегідь. Це повністю знімає важке обчислювальне навантаження з графічного процесора пристрою користувача.
2. Абсолютне позиціонування інтерфейсу: Поверх статичного зображення, встановленого як background-image у відповідному DOM-контейнері, накладаються невидимі інтерактивні зони

(hotspots) за допомогою точного CSS-позиціонування у відсоткових координатах.

3. Голографічний UI та Glassmorphism: Для взаємодії користувача з об'єктами інфраструктури реалізовано спливаючі інформаційні панелі з ефектом матового скла (backdrop-filter: blur()). Панелі анімовані за допомогою CSS-переходів (transition, cubic-bezier), що створює ефект плавного появи інтерфейсу при наведенні курсора миші. Пульсуючі маркери, реалізовані через директиву , привертають увагу користувача до інтерактивних точок без додаткового перевантаження DOM-дерева скриптами.

**Інтеграція з серверною частиною (Django)**

Візуальна прогресія користувача безпосередньо пов'язана з його навчальною активністю та внутрішнім балансом ігрових ресурсів. У реляційній моделі бази даних (сутність Profile) зберігається поточний рівень ратуші, а також показники прогресу (еліксир, золото, трофеї).

При виконанні користувачем дії «Покращити» (Upgrade) система обробляє HTTP POST-запит, проводить валідацію наявності достатньої кількості ресурсів на рівні бізнес-логіки бекенду, виконує транзакцію списання та інкрементує рівень будівлі в базі даних. На стороні шаблонізатора (Django Templates) реалізовано динамічну зміну фонового зображення залежно від поточної адреси файлу, що зберігається в профілі.

Шлях до актуального графічного ресурсу обчислюється на рівні контролера (View) і передається в контекст рендерингу сторінки. Таким чином, при досягненні певних рівнів (наприклад, з 1 по 10), відбувається моментальна візуальна еволюція об'єкта.

Розроблений підхід дозволив досягти високого ступеня залученості здобувачів освіти завдяки якісній візуалізації (у стилістиці

кіберфентезі з неоновими елементами), зберігши при цьому миттєву швидкість завантаження сторінок та чуйність інтерфейсу, що є критично важливим фактором для сучасних вебдодатків освітнього профілю.


Рис. А.1. Графічні ресурси еволюції ігрових об'єктів вебдодатка.