МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЩЕРБИНА ДМИТРО СЕРГІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____ О.В. Зелінська
«_____» _____ 2025 р.

НАВЧАЛЬНИЙ ВЕБДОДАТОК ДЛЯ ГРИ В ШАХИ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Потапова Н. А., доцент кафедри
інформаційних технологій,
к. е. н., доцент

_____

Оцінка: _____/_____ / _____
(бали за шкалою ЄКТС / за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2025

АНОТАЦІЯ

Щербина Д. С. Навчальний вебдодаток для гри в шахи. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2025.

У кваліфікаційній (бакалаврській) роботі досліджено та проаналізовано поняття навчального вебдодатку, виділені основні їх принципи та створення. За допомогою таких технологій як: Python, Flask, HTML, CSS, JavaScript, бібліотеки Fetch API був розроблений навчальний вебдодаток, основна мета якого – навчити користувачів грати в шахи.

Ключові слова: навчання, вебдодаток, шахи, гра, Python, JavaScript.

70 с., 32 рис., 42 джерела.

ABSTRACT

Shcherbyna D. Educational Web Application for Playing Chess. Specialty 122 «Computer Science», educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia 2025.

In the qualification (bachelor's) work the concept of educational web application is researched and analyzed, their basic principles and creation are highlighted. With the help of such technologies as Python, Flask, HTML, CSS, JavaScript, library Fetch API developed a educational web application to teach users how to play chess.

Keywords: education, web application, chess, game, Python, JavaScript.

70 p., 32 figures, 42 sources.

ЗМІСТ

ВСТУП

У сучасному світі цифрові технології швидко розвиваються та активно впроваджуються у різні сфери життя, а особливо в освіту. Популярними є інтерактивні вебдодатки, які дозволяють навчатися самостійно, що значно пришвидшує процес навчання та інтерес до нього. Популярність шахів зростає з кожним роком, і це заохочує все більшу кількість людей навчатися грати в них. Професійні шахісти та висококваліфіковані тренери не завжди хочуть витрачати час на навчання новачків, що зменшує шанси останніх на гарний та швидкий ріст. Вирішенням цієї проблеми є створення навчального вебдодтаку, який наддасть можливість для самостійного навчання та освоєння всього необхідного матеріалу. Розробка інтерактивного вебдодатку дозволить зробити процес вивчення гри простішим, наочним і захопливим, особливо для початківців. Це буде сприяти популяризації шахів та впровадження сучасних цифрових рішень у навчальний процес.

Метою бакалаврської роботи є розробка навчального вебдодатку для гри в шахи, яка відтворює ігровий процес в інтерактивному режимі та забезпечує навчання за рахунок проведення уроків різної категорії зі збереженням отриманих навичок.

Об'єктом дослідження є процес розробки навчального додатку для гри в шахи з акцентом на можливості проходження уроків різного рівня з елементами збереження партій.

Предмет дослідження – методи, алгоритмічні підходи та технологій програмування реалізації навчального вебдодатку для гри в шахи.

Для виконання поставленої мети були сформульовані наступні завдання:

– дослідити теоретичні підходи та категорії використання гейміфікації в навчальних процесах із використанням освітніх платформ;

– проаналізувати основні етапи історичного розвитку шахової гри та її освітню цінність;

– провести аналіз технологічних підходів та обґрунтувати вибір інструментарію для розробки вебдодатку для гри в шахи;

– розробити алгоритми основних функцій навчального вебдодатку, зокрема: гри та аналізу партії разом з рушієм, збереження зіграних та проаналізованих партій, проходження уроків з різною тематикою та ін.;

– виконати програмну реалізацію навчального вебдодатку для гри в шахи, включаючи серверну та клієнтську частину;

– провести оцінку функціонування вебдодатку на пробних партіях гри в шахи.

Структура роботи містить вступ, три розділи, висновки, список використаних посилань та додатки. У першому розділі розглядаються теоретичні основи використання елементів гейміфікації в освітніх цілях. Другий розділ містить результати аналізу технологічних підходів щодо розробки навчальних платформ. У третьому розділі наведені результати програмної реалізації вебдодатку для гри в шахи з відповідними функціями.

Практичне значення роботи полягає в тому, що дана розробка може бути використана як один із елементів навчання гри в шахи як в процесі самостійного опанування стратегій гри, так і в закладах освіти з метою залучення до гри в шахи все більшого кола гравців.

Апробація даної роботи була представлена в доповіді «Застосування алгоритму альфа-бета відсікання для оптимізації шахових рішень» на V Всеукраїнській науково-практичній конференції «Комп’ютерні технології обробки даних» (м. Вінниця, 6 грудня 2024 року), VI Всеукраїнській науково-практичній конференції «Прикладні інформаційні технології» (м. Вінниця, 22 травня 2025 року) в доповіді «Гейміфікація навчального процесу у шахах для покращення сприйняття матеріалу».

РОЗДІЛ 1
ТЕОРЕТИЧНІ ПІДХОДИ ГЕЙМІФІКАЦІЇ В РОЗВИТКУ ОСВІТНІХ ПЛАТФОРМ

1.1 Сутність підходів гейміфікації в освітній діяльності

Стрімкий розвиток технологій за останні роки посприяв тому, що вже з малечку діти привчаються користуватися різними новітніми пристроями або гаджетами. Ці умови створюють простір, який зменшує інтерес до методів традиційної освіти. Проте щоб знову наситити дітей бажанням до навчання до нього додають гейміфікацію.

Гейміфікація – це додавання ігрових елементів та механік до якогось неігрового процесу, у даному випадку навчання. Саме це буде стимулювати та надихати учнів та студентів навчатися далі, оскільки ти не просто вивчаєш якийсь матеріал, тобі дають бонуси, винагороди або щось інше. Проте не потрібно перетворювати весь процес на гру, оскільки тоді втрачається сам сенс навчання, бо головною метою стає не навчитися чомусь, а по швидше отримати нагороду, і це не дає сконцентруватися на вивченні потрібного матеріалу [1]. Вдало застосована до навчального процесу гейміфікація має вирішувати наступні проблеми:

1. Кількість винагород – потрібно вдало розрахувати потрібну кількість винагород на відповідному проміжку навчання. Зсув у ту чи іншу сторони може призвести до втрати бажання та мотивації, оскільки дуже велика кількість винагород може перетворити навчання на гру, а їх мала кількість може загасити мотивацію, яка була перед початком.

2. Складність завдань – потрібно як і в будь-якій грі поступово збільшувати складність завдань, які має виконати учень. Це може бути як безпосереднє збільшення складності самих завдань, так і збільшення їх кількості. Паралельно також мають бути покращені винагороди, оскільки вони є взаємозалежними [2].

3. Змагальний тип навчання – ігрова механіка дозволяє окрім винагород також додати рейтинги успішності у вигляді турнірної таблиці. Це дозволить

мотивувати учнів ще більше, оскільки кожен буде намагатися зайняти вершину таблиці, що буде підігрівати жагу до навчання.

4. Особистий план навчання – кожен учень різний, тому потрібно щоб він міг сам налаштувати потрібну складність та кількість завдань, ця функція доступна в багатьох навчальних додатках, таких як Duolingo. Таким чином він зможе самостійно контролювати кількість матеріалу, яку здатний засвоїти у відповідний термін.

Дотримання цих основних вимог гарантує гарне впровадження гейміфікації в навчальний процес. Проте варто пам'ятати, що в першу чергу потрібно забезпечити зручний процес навчання для учнів, тому додавання або зменшення гейміфікованого функціоналу має коригуватися залежно від вікової групи, обставин, галузі та багато іншого.

Саме поняття геймофікація почало свій шлях в 2002 році, коли британський розробник відеоігор Нік Пеллінг описував процес додавання елементів гри до неігрових систем для того, щоб підвищити привабливість та ефективність останніх [2]. Пізніше це поняття набуло більшого розголосу, і його почали розглядати як засіб стимулювання глибокого залучення користувачів через ігровий досвід, що базується на відповідних правилах і цілях та користування їм викликає задоволення (Б. Кірк, Т. Кренк, 2009) [3].

Після 2010 року гейміфікація набула стрімкого розвитку, яка зазнала перетворень від невідомого поняття до міждисциплінарного підходу, який активно використовується в таких сферах як маркетинг, бізнес та освіта. Однією з причин цього явища стало зростання популярності цифрових технологій, що надають широкі можливості для втілення ігрових механік у буденні процеси. У цей період виникло багато компаній та сервісів, які спеціалізуються на гейміфікованому підході. Одним з найвідоміших прикладів є Duolingo, бета-тест якого вийшов наприкінці 2011 року, а сам проект повноцінно стартував в 2012 році. Він запропонував новітній спосіб вивчення різних мов на основі рівнів, балів, щоденних завдань та досягнень. Ще однією відомою платформою є Khan Academy, яка впроваджувала елементи гейміфікації у вигляді значків і балів для

учнів різного віку. Також у цей період розвивалися корпоративні рішення. Наприклад, компанія Bunchball, яка була заснована ще у 2005 році, після 2010 року стала одним із лідерів застосування гейміфікації у сфері бізнесу, вона запропонувала платформу Nitro, яка надавала компаніям інструменти для впровадження ігрових механік у внутрішні процеси, такі як бейджі, бали, індикатори виконання, таблиця лідерів та інше [4]. Це дозволяло трансформувати рутинні робочі завдання в захоплюючу діяльність, яка мотивувала працівників до досягнення цілей, плинність кадрів та покращувала загальну ефективність команди. Ще одним гігантом на цьому ринку стала платформа Badgeville, хоча вона і з'явилася у 2010 році, проте стрімко набрала популярності у 2011-2013 роках. Компанія орієнтувалася на створення гейміфікованих систем, які сприяли підвищенню активності та лояльності користувачів, використовуючи динамічні інструменти мотивації як для співробітників, так і для клієнтів. Система дозволяла відслідковувати дії користувачів у реальному часі, заохочувати їх за виконання головних завдань, а також персоналізувати винагороди. Компанії Bunchball і Badgeville стали джерелами натхнення для наступних поколінь гейміфікованих рішень, демонструючи, що ігрові механіки можуть бути ефективними не лише для розваг або навчання, а й у серйозних бізнес-завданнях.

Значний вклад у популяризацію та теоретичне обґрунтування гейміфікації зробив Ю Кай Чоу, що є відомим фахівцем у сфері поведінкової мотивації та автор відомої мотиваційної моделі Octalysis. Створена ним у 2014 році, модель стала однією із найвідоміших і найбільш використовуваних структур у практиці гейміфікації. Вона базується на восьми основних мотиваційних рушіях, які, як вважає автор, визначають поведінку користувачів у системах з ігровими елементами. Ці рушії включають такі фактори як відчуття мети, розвиток майстерності, унікальний досвід, соціальний вплив, уникнення витрат тощо. Головною особливістю Octalysis є орієнтація на лише на зовнішні стимули, такі як нагороди, бали чи рейтинги, але й на внутрішню мотивацію, а саме: задоволення, самовираження та відчуття прогресу. Модель отримала гарну підтримку завдяки своїй гнучкості, оскільки її можна використовувати у різних

сферах, таких як бізнес, дизайні продуктів, освіті чи соціальних проектах. Octalysis допомогла багатьом фахівцям краще зрозуміти, як саме створювати ефективні гейміфіковані рішення, які не просто заохочують до взаємодії, а формують постійне залучення та позитивний досвід користувача [5].

Таким чином, розвиток гейміфікації пройшов свій шлях від окремих спроб використання ігрових механік у 2000-х роках до цілих концепцій після 2010-х років. Враховуючи стрімкий темп поширення гейміфікації, варто оцінити її основні переваги та можливі недоліки для того, щоб ефективно використовувати дану методику у навчальному процесі. Тому щоб підсумувати всю цінність застосування геймофікації, варто розглянути, які переваги вона має:

1. Легше візуалізувати та наголосити на тому матеріалі, який важко сприймається з допомогою традиційних методів навчання.
2. Краще стимулює інтерес до нового матеріалу, оскільки пізнавати нові знання, які відрізняються від буденних своїм креативних підходом, набагато краще.
3. При використанні елементів гри легше направити учня на певний шлях, оскільки виникає відчуття власного вибору, що не спричиняє додаткового примусу та тиску від викладача, тим самим покращує сприйняття матеріалу.
4. Нестандартний підхід до навчання дозволяє розвивати критичне та творче мислення, оскільки виконання таких завдань змушує вийти за звичні межі і шукати нові шляхи вирішення задач.
5. Додавання різних обмежень як загальний час виконання, кількість «життів» тощо, дозволить одночасно навчити важливості вибору та інших важливих навичок, а також дасть можливість помилятися, що дуже важливо для навчального процесу.
6. Можна відстежувати діяльність кожного учня, що дозволяє регулювати його план навчання відповідно до його темпу проходження завдань.
7. Встановлення чітких правил виконання та оцінювання дозволяє справедливо оцінити кожного учасника навчального процесу.

8. Залучає всіх учнів, що дає змогу покращити їх комунікацію та командну роботу, а також створює кращу атмосферу для проведення занять [6].

Ці переваги виглядають дуже привабливо, крім того, можливо, можна знайти й інші переваги або аспекти в яких гейміфікація буде проявляти себе в повну міру. Проте будь-який метод має свої переваги та недоліки, це не оминуло і гейміфікацію. Тому варто розглянути можливі недоліки, які можуть виникнути під час її застосування, щоб забезпечити її надійність. До таких недоліків належать наступні елементи:

1. Оскільки використання ігрових елементів передбачає наповнення в більшій мірі основним матеріалом, то іноді можна не повністю передати ту чи іншу тему, що може вплинути на розуміння усього матеріалу.

2. Надмірна кількість ігрових елементів може призвести до того, що втратиться межа між навчанням та грою, і все перейде в бік останньої, що перетворить навчальний процес на просту гру.

3. При використанні рейтингової системи не варто надавати великі бонуси для лідерів, оскільки це може викликати негативне ставлення від іншої частини учасників. Натомість варто або надавати бонуси схожі на звичайні, які можна отримати за інші завдання, або взагалі не видавати нагороди за рейтингові місця, пояснюючи це тим, що у кожного різний темп навчання.

4. Одноманітність завдань може знизити інтерес до процесу навчання, а як наслідок почнуть забуватися знання, оскільки не буде отримуватися задоволення від виконання наступних завдань.

5. Уподобання користувачів можуть різнитися, хтось може не сприймати велику кількість ігрових елементів, а хтось, взагалі, негативно ставитися до самої ідеї гейміфікації, тому це може відштовхнути їх.

6. Потрібно мати великий запас ресурсів як фінансових, так і часових, оскільки створення ігрових механік для різних груп користувачів є дуже затратним процесом [6].

Тому перед створенням гейміфікованої системи варто проаналізувати аудиторію, її вподобання, темп та терміни розвитку, а також інші важливі аспекти,

які впливають на формування вимог.

Класифікувати гейміфікацію можна за різними критеріями, які будуть залежати від підходу до використання ігрових механік. Одним із найпоширеніших поділів є розбиття на дві великі групи, а саме структурну та контентну гейміфікацію. Структурна класифікація передбачає додавання ігрових механік до навчального процесу без зміни його основної сутності. Наприклад, можуть з'явитися бейджі, бали або рейтингові таблиці, але самі навчальні матеріали та методи залишаються традиційними. Даний підхід спрямований на зростання залученості користувачів через створення додаткової мотивації при не втручанні в основний процес. А ось контентна гейміфікація безпосередньо змінює саму методику подачі матеріалу, роблячи його більш ігровим та інтерактивним. У даному випадку навчальні завдання перетворюються на квести, місії та інтерактивні задачі, які потребують від користувача активної участі, критичного та творчого мислення [7]. Отже, правильний вибір типу гейміфікації є важливим для ефективнішого планування і впровадження ігрових механік відповідно до поставлених цілей і специфіки аудиторії.

Для побудови ефективної гейміфікаційної системи потрібно правильно обрати та інтегрувати основні елементи гри. Серед ключових механік переважно виділяють:

– Бали – є одним з найпростіших способів відстеження прогресу користувача. Їх можна отримати за визначену в правилах активність, наприклад виконання завдань, час проведений за навчанням тощо.

– Бейджі – нагороди, які надаються за певні досягнення, демонструють ваш прогрес перед іншими користувачами.

– Рівні – відображають етап на якому знаходиться користувач. Підвищення рівня супроводжується отриманням нових нагород, бонусів, доступу до додаткового контенту та інше.

– Завдання або місії – допомагають структурувати діяльність користувача, надаючи зрозумілі та чіткі цілі та мотивацію для їх виконання або досягнення.

– Винагороди та бонуси – додаткові стимули, які підсилюють мотивацію для виконання завдань [8-9].

Ці елементи мають бути ретельно збалансовані та орієнтованими на потреби аудиторії. Непотрібне нагромадження ігрових механік може призвести до втрати інтересу користувача щодо всього процесу. Тому при проектуванні системи гейміфікації потрібно врахувати не лише, які типи елементів задіяти, а й їх послідовність та взаємодія між собою.

Станом на сьогодні гейміфікація продовжує активно розвиватися, підлаштовуючись під нові технологічні та соціальні потреби. Одним із ключових напрямків є інтеграція штучного інтелекту у гейміфікаційні платформи. Завдяки штучному інтелекту гейміфікація стане більш персоналізованою, оскільки система буде аналізувати поведінку користувача і підлаштовувати завдання, нагороди або рівень складності під індивідуальні можливості кожного. Іншим важливим напрямком є використання доповненої та віртуальної реальностей. Ігрові елементи стають частиною реального світу через мобільні пристрої та окуляри віртуальної реальності. Це дозволить створювати ще більш захопливі та інтерактивні навчальні середовища в будь-якій сфері застосування гейміфікації. Також зростає популярність гейміфікація корпоративного навчання. Багато компаній почали використовувати гейміфікаційні підходи для підвищення мотивації співробітників до розвитку навичок, проходження тренінгів та участі у внутрішніх проектах. Отже, сучасні технології значно покращили можливості застосування гейміфікації, і її подальший розвиток буде тісно пов’язаний із персоналізацією досвіду та впровадженням нових цифрових рішень.

Підсумовуючи, можна сказати, що гейміфікація пройшла тривалий шлях розвитку від перших концептуальних ідей до створення повноцінних моделей. Завдаяки різним підходам та інноваційним технологіям гейміфікація сьогодні активно використовується в освітньому процесі, бізнесі й інших сферах, демонструючи значний потенціал для розширення свого впливу. Водночас її впровадження вимагає ретельного врахування як переваг, так і можливих

недоліків, що дозволить забезпечити доцільність і результативність застосування гейміфікованих підходів у конкретних умовах.

1.2 Історія шахової гри та її освітня цінність

Шахи – стратегічна гра на спеціальній дошці, яка має назву шахівниця і має по 32 білі та чорні клітинки. Гра на шахівниці ведеться між двома гравцями, кожен з яких перед початком партії має по 16 білих та чорних фігур. Комплект фігур складається з короля та ферзя (королеви), двох тур, коней та слонів (офіцерів), а також восьми пішаків (рис. 1.1).

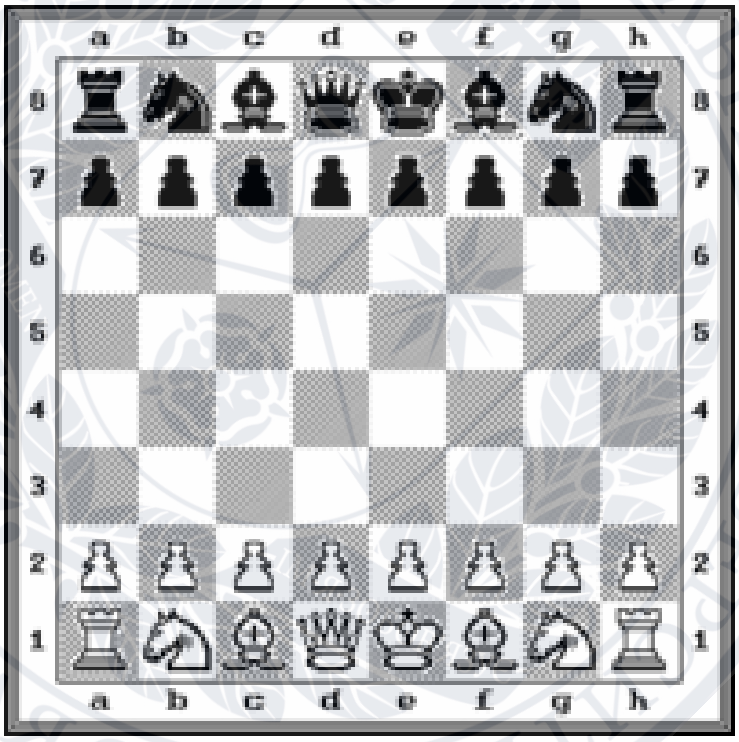

Рисунок 1.1 – Шахівниця та фігури

Кожна фігура має різні правила пересування, силу та вартість. Для легшого підрахунку вартість кожної фігури оцінюється в пішаках, оскільки вони є найслабшими фігурами. Отже, маємо наступний розподіл вартості кожної фігури:

- Кінь – 3 пішака;
- Слон – 3 пішака;

– Тура – 5 пішаків;

– Ферзь – 9 пішаків.

У цьому списку немає лише короля, і на це є дві причини. По-перше, за правилами шахів король єдина фігура, яка завжди присутня на шахівниці, і може бути лише в одному екземплярі. По-друге, він є найважливішою фігурою, від якої залежить доля партії. Тому можна сказати, що король є безцінною фігурою.

Кожна шахова партія поділяється на три етапи гри. Дебют (початок партії) є першим етапом, переважно він триває від 10 до 15 ходів. Зазвичай дебют складається з теоретичних ходів, які вже давно проаналізовані й людина обирає той початок, який їй більше подобається, або підходить проти того чи іншого супротивника. Середня стадія гри – мітельшпіль, він є найменш дослідженим етапом гри як теоретично, так і практично. Саме протягом цього етапу можна побачити найбільший рівень гри, яку може продемонструвати гравець. Тривалість мітельшпілю може сягати від декількох до кількох десятків ходів. Основною причиною цьому є кількість фігур, чим менше фігур, проте це не стосується пішаків, залишається на дошці, тим ближче третій етап гри. Останнім етапом шахової гри є ендшпіль, він потребує найбільшої майстерності як в теоретичних знаннях, так і у вашій можливості прорахувати варіанти наперед. Даний етап є найбільш математичним у шахах, оскільки існують такі позиції, де серед багатьох варіантів є лише єдиний, який приведе вас до бажаного результату.

Історія шахів починається з 5-6 століття нашої ери, тоді ця гра на території Індії називалася чатуранга. Вона мала зовсім інші правила гри, які зараз будуть виглядати дивакуватими, але компоненти гри були такі самі. Чатуранга мала схожість з сучасними шахами через те, що фігури на дошці мали різну силу та кінець гри залежав від однієї фігури, нині цю роль відіграє король [10].

Трохи пізніше чатуранга еволюціонувала та перетворилася на шатрандж, як саме це сталося наразі дізнатися неможливо. Нова версія гри відрізнялася від свого попередника правилами самої гри, все інше залишилося без змін. Саме шатрандж вийшов за межі Індії і почав розповсюджуватися Азією. А вже ближче

до кінця першого тисячоліття форми чатуранги та шатранджу потрапили до таких країн як Персія, Візантійська та Аравійська імперії, а також Європа. Саме після цього пішов процес формування нових шахів, які поступово дійдуть до відомої всім форми.

Приблизно в 16 столітті формування правил гри в шахи нарешті було завершено і починаючи з того часу можна побачити стандартизовані правила гри, які збереглися по сьогоднішній день. Першою людиною, яка зробила вагомий внесок у формування тогочасних шахів, став іспанський священик Руй Лоцес. Він першим проаналізував початкову стадію гри в шахи – дебют, у своїй книзі, яку опублікував у 1561 році. Хоча теорія шахів тоді була дуже примітивною, це стало вагомим вкладенням у розвиток тогочасних шахів [11].

Починаючи з чатуранги та шатранджу були застосовані різні декоративні форми фігур, але єдиного вигляду вони не мали. Так як шахи того часу були грою для знаті, то нерідко у дизайн фігур додавалися різні коштовності та дорогоцінні метали. Це відволікало гравців, і вони приділяли більшість своєї уваги не самій грі, а перегляду різних дизайнів фігур. Також через різну ідеологію та назви фігур, багато країн мали не тільки різні дизайни фігур, але і різні розміри. Але приблизно в 1835 році англієць Натаніель Кук виготовив стандарт для сучасних наборів з простим дизайном. Після того як цей дизайн був запатентований у 1849 році, його схвалив найсильніший гравець того часу Говард Стаунтон, саме тому цей дизайн шахів почав носити його фамілію, а не фамілію автора, і називався він шахи Стаунтон. Після цього цей новий стиль фігур став популярним і почав використовуватися на турнірах і в клубах по всьому світу. Фігури Стаунтон та їх розмірні варіації до сих пір вважаються стандартом для турнірних шахів по всьому світу. Найбільшими виробниками таких фігур є Польща та Індія, а загальноприйнятими для турнірів вважаються шахи Стаунтон №6 (висота короля 98 мм), іноді допускаються їх менша версія – Стаунтон №5 (висота короля 90 мм) (рис. 1.2). Саме ці два вили наборів шахів мають найбільш вдалі пропорції для гри на змаганнях, інші будуть або занадто великі (№7 та №8), або занадто малі

(№3 та №4), проте для тренувань можна використовувати будь-який набір, оскільки там припускаються неуважність та помилки [12].

Пізніше у 19 столітті з’явилися шахові годинники. Їхня поява була необхідна оскільки партії без обмеження у часі могли тривати дуже довго. Спочатку це були просто механічні годинники з прапорцем, коли прапорець падав, то це означало програш за часом. Пізніше, наприкінці 20 століття, одинадцятий чемпіон світу з шахів Роберт Джеймс Фішер у 1988 році запатентував електронний шаховий годинник (рис 1.2). Сутність цього годинника, крім великої кількості вбудованих часових режимів, полягала в тому щоб надати гравцям можливість довести партію до логічного кінця. У шахах є таке поняття як цейтнот – стадія гри, коли в одного чи двох гравців залишається мало часу, саме через це Фішер запропонував зробити додавання часу за кожен зроблений хід, оскільки через додавання часу можна буде вдало завершити партію. Перший електронний годинник було представлено на матчі Фішер – Спаський в 1992 році, хоча це був і не досконалий годинник, тоді це був прототип годинника Fisher, він вдало себе зарекомендував. Проте все не зупинилося на гарному старті і розвиток даного напрямку продовжився, а саме зараз нідерландська компанія DGT виробляє найкращі електронні годинники, якими користується весь шаховий, і не тільки, світ [12].

Рисунок 1.2 – Шахи Стаунтон №6 та різновид електронного годинника

Коли шахи ще не користувалися такою популярністю як зараз, вони привертали увагу, у більшості випадків, матчем між двома людьми за звання чемпіона світу. До 1886 року це звання було неофіційним. Проводилися багато різних матчів у великих містах як Лондон та Нью-Йорк. Через невеликі знання в теорії більшість людей віддавали перевагу спостерігати за стрімким атакуючим стилем, тому найпопулярнішим та найсильнішим гравцем на той час вважався Пол Морфі. Він був еталоном атакуючої гри й переміг багатьох сильних гравців того часу. Він надихнув своєю грою багатьох людей, такиї як Адольф Андерсен, Луїс Паульсен, Даніель Харвін та багато інших талановитих шахістів, які потім стали сильними майстрами тієї епохи [11].

Першим офіційним чемпіоном світу з шахів став Вільгельм Стейніц у 1886 році. Незадовго до того як стати офіційним чемпіоном, він вже мав статус найсильнішого шахіста. Оскільки він був повною протилежністю Морфі і не визнавав атакуючий стиль шахів, тому в якості першого сильного гравця у позиційному стилі, він легко перемагав ще не звиклих до такої стратегії гравців. Саме притримуючись такого стилю гри, він переміг у матчі проти Йоганна Цукертopта і став першим офіційним чемпіоном світу. Стейніц зробив великий внесок у розвиток шахів, оскільки він почав вимагати, щоб матчі за титул чемпіона світу проводилися за певних правил та фінансових умов, що змусило людей ставитися до шахів як до серйозного спорту. [11]

Після Стейніца і до сьогоднішнього дня було 18 чемпіонів світу. Про кожного чемпіона можна розповісти багато цікавих фактів, а багато з них робили рекорди, які назавжди запам’ятає світ та шахова спільнота. Наприклад, Емануель Ласкер, другий чемпіон світу з шахів, досі утримує рекорд найдовшого володіння шаховою короною, а саме 27 років, звичайно цей рекорд не зовсім можна вважати справедливим, оскільки у ті часи теорія була не дуже розвинена, тому тяжко було підготуватися до суперника, але навіть без цього кількість років є вагомою. Наступний шаховий рекорд є найвагомішим зі всіх, що можна уявити в шахах, а саме цей рекорд належить 16 чемпіону світу Магнусу Карлсену (рис. 1.3), який досяг відмітки рейтингу у 2882 одиниці з класичних шахів в 2014 році. Цей

рекорд можна назвати абсолютним і навряд чи хтось зможе його побити, оскільки починаючи з липня 2011 року не було жодного місяця, коли Карлсен поступився першим місцем в рейтингу класичних шахів, крім того позначка в 2882 одиниці покорилася йому ще одного разу в серпні 2019 року, що свідчить про вплив віку на силу шахіста, оскільки на побиття власного рекорду не вистачило сил. Саме за такий рівень гри Магнуса Карлсена вважають найсильнішим шахістом за всю історію шахів [11].

Рисунок 1.3 – Магнус Карлсен – 16 чемпіон світу з нахів

Велику роль у розвитку шахів велику роль зіграли комп’ютери. Щоб обрати правильний хід людина прораховує варіанти ходів і не тільки своїх, а і суперника, для правильності розрахунку потрібно обраховувати не один варіант, іноді ці числа можуть бути двозначними. Проблема полягає в тому, що потрібно вираховувати ходи білих та чорних не тільки на один хід, а на декілька ходів вперед, а це є дуже важким процесом, тому на допомогу прийшли комп’ютери. Спочатку комп’ютери не вміли прораховувати велику кількість комбінацій, оскільки чим більше потрібно продумати ходів наперед, тим більше потрібно зробити обрахунків, наприклад для 20 перших ходів білих та чорних комп’ютеру

потрібно порахувати 400 позицій, а зі збільшенням ходів збільшується і кількість позицій для обрахунку. Саме через це комп'ютер 1960-х років міг бачити наперед лише на два ходи [13]. Головна відмінна риса людини від комп'ютера полягає в тому, що їй не потрібно переглядати усі можливі варіанти ходів, вона може оцінити позицію й обрати найкращі продовження з можливих. Саме такий спосіб започаткував Клод Шеннон. Він припустив, що можна оцінити позицію кожної фігури, її можливі ходи та активність. Ця теорія стала відправною точкою для покращення роботи оцінки комп'ютера, яку потім програмісти з усього світу доповнювали та вдосконалювали [14]. Результатом цього стало, те що вже у 1988 році комп'ютер HiTech, який був розроблений в університеті Карнегі-Меллона (Пенсильванія, США), переміг гросмейстера Арнольда Денкера, і пізніше в тому ж році інша програму цього ж університету перемогла сильного гросмейстера Бента Ларсена [15].

Розвиток комп'ютерів не зупинився і вже у 1997 році в Нью-Йорку відбувся другий матч між 13 чемпіоном світу Гаррі Каспаровим, та другою версією комп'ютера Deep Blue. Сам матч закінчився у нічию: по 1 виграшу та 3 нічиї, а Deep Blue вже міг обраховувати в середньому 200 мільйонів позицій в секунду. Такий розвиток комп'ютерів допоміг не тільки у аналізі нових партій, а і у покращенні старих аналізів та теорій. Так починаючи з 1986 року Кеннет Томпсон за допомогою комп'ютера зробив низку досліджень в останній стадії гри – ендшпілі, а саме низка позицій з кількістю не більше 5 фігур в книжках вважалися нічийними, але попри це комп'ютер показував неймовірні показники перемоги для сторони з перевагою у матеріалі. Такий розвиток комп'ютерів дозволяє робити величезні досягнення, наприклад існують таблиці Налімова, які здатні обрахувати будь-яку 8 фігурну позицію до кінцевого результату, найдовший обрахована позиція має близько 400 ходів і закінчується перемогою однієї зі сторін [16-17].

Шахи завжди були доволі популярними, але більшість зацікавлених були тими людьми, які хоч якось мали справу з ними. У жовтні 2020 року виходить серіал «Ферзевий гамбіт» в якому жінка долає шлях до становлення сильною

шахісткою, саме після цього серіалу сталася перша хвиля різкого росту популярності шахів (рис. 1.4). Однак після початку коронавірусної пандемії, коли дуже багато людей перейшли в офлайн режим, вони почали звертати увагу на різні речі, якими раніше не цікавилися. Саме велика кількість вільного часу стала причиною зацікавленості шахами. Це стало другою хвилею приросту нових людей в сферу шахів. За даними сайту Chess.com, кількість зареєстрованих аккаунтів з жовтня 2020 року по квітень 2022 зросла вдвічі. Також ще одним прикладом, який доводить різку зацікавленість шахами є турнір PogChamps серед стрімерів Twitch. До вищезазначених подій, даний турнір мав рекордний онлайн в 165 000 глядачів у травні 2020 року, але вже в лютому 2021 року ця позначка сягала 375 000 глядачів, що свідчить про дуже стрімкий приріст глядацької аудиторії. Але до цих подій багато людей, які ніколи не мали справу з шахами, не могли зрозуміти їхнього спектру застосування, оскільки шахи не обмежені лише шахівницею, а їхні можливості дуже великі й широко використовуються в різних сферах [18].

Рисунок 1.4 – Плакат серіалу «Ферзевий гамбіт»

Шахи суттєве допомогли розвинутися штучному інтелекту. У нашому світі все взаємопов’язано, тому як було сказано вище комп’ютери допомогли розвинутися шахам в аналізі та теорії, так і відповідно цьому шахи допомогли розвинутися комп’ютерам у своїх розрахункових можливостях. Клод Шеннон не

створював свої теорії тільки для розвитку шахів, у першу чергу він намагався створити комп'ютер, який міг мислити і приймати складніші рішення. Рано чи пізно людина створить комп'ютер, який буде розумнішим за неї, оскільки він мав ряд переваг як відсутність втоми та нудьги, що впливали на концентрацію, так вважав Шеннон. Його підхід до розробки таокго комп'ютера був наступним: він намагався створити його схожого на людину, і використовував усі слабкі та сильні сторони. Головна проблема комп'ютера була в тому, що він не міг навчатися у процесі, тобто у нього були відсутні аналітичні здібності. Тому Шеннон використав сильні сторони комп'ютера – швидкість і точність, він створював різні стратегії для оцінки позиції на дошці та вибору найкращого ходу. Саме його підхід дозволив еволюціонувати комп'ютерам і зараз можна спостерігати за успіхом розвитку усього штучного інтелекту [15].

Штучний інтелект, який використовувався для аналізу шахових позицій продовжував невпинно розвиватись. Так починаючи з 2008 року, почала виходити серія потужних шахових механізмів StockFish, і з того часу вона вважалася незмінним фаворитом в аналізі шахів. Її механізм, як і у попередників, базувався на пошуку найкращого ходу в дереві гри. Але революційною стало винайдення нового механізму, який перший було започатковано в штучному інтелекті AlphaZero. Цей новий підхід називається навчання з підкріпленням, суть даного підходу була в тому, що штучний інтелект грає сам проти себе та нагороджує себе за виграш та нічию. Саме таким методом AlphaZero перевершив традиційні шахові двигуни, і як доказом переміг StockFish в матчі зі 100 партій з рахунком 28 перемоги та 72 нічиї. Порівнюючи зі StockFish, який славився сильною позиційною оцінкою та глибиною пошуку в дереві, новий AlphaZero мав здатність робити креативні та несподівані кроки [16]. Поява такого шахового двигуна як AlphaZero стала революційною у світі програмування, оскільки це показало, що нетрадиційний підхід до вирішення стандартних задач може стати ключем до вирішення більш складних завдань. Зараз є багато нових різновидів штучного інтелекту, які виконують різні функції. Наприклад цікавим вирішенням став «аналог» шахів – розробка програмного забезпечення. Ці два напрямки є

схожими за рядом показників: вимагають креативності, стратегічного мислення, адаптивності та розпізнавання шаблонів, оскільки без цих факторів ви не отримаєте перемоги або не зможете створити нові програми. Саме тому такий підхід у використанні можливостей штучного інтелекту принесе ряд нових рішень, які покращать як швидкість написання коду, так і його якість.

Також важливим напрямком використання шахів є навчання, це в першу чергу не пов'язується з досягненням певного рівня гри, а в концепції загального розвитку інших навичок учня. Завдяки своїм стратегічним напрямком шахи можуть бути корисними для розрахунку ризиків. Оскільки в шахах є таке поняття як жертва – віддача матеріалу заради майбутньої вигоди, то розрахунок ризиків є головним способом для здійснення вдалої жертви. Якщо поміркувати, то прийняття будь-якого рішення це є розрахунок ризиків, оскільки кожне наше рішення може викликати ряд інших подій. Саме тому вплив шахів у навчанні дозволить учням навчитися прораховувати ризики в кожному своєму рішенні. Також шахи виконують не тільки функцію розвитку мислення, але є дисциплінарну. Для вдалого проведення партії вам завжди потрібно бути зосередженим та контролювати свої емоції, оскільки зайва проява радісних або сумних емоцій можуть суттєво вплинути на результат партії. Тому ці дві якості важливі при грі в шахи, а оскільки з часом їх все важче набути, то оптимальний час для їхнього опанування є шкільний вік. Крім того не потрібно забувати, що шахи гарно поліпшують пам'ять та розвивають аналітичне мислення. Для перемоги в шахах потрібно знати багато комбінацій, і коли ці комбінації відшліфуються у вашій пам'яті, за діло береться розвинуте аналітичне мислення. Оскільки шахові комбінації - це не секретні матеріали, то більшість шахістів знають однотипні комбінації і все залежить від особистих навичок, а саме наскільки стало гнучким ваше мислення, щоб обрати варіант, який приведе вас до перемоги. Тому заради цих нагород людям, які вивчають точні науки, варто інколи грати в шахи, щоб тренувати свою пам'ять та розвивати мислення [19].

У XXI столітті шахи дедалі частіше інтегруються в освітній процес як ефективний інструмент розвитку інтелектуальних і соціальних навичок учня. У

багатьох країнах шахи стали частиною шкільної програми або факультативною дисципліною, спочатку це були країни такі, як США, Китай, Іспанія та інші, а вже поступово дійшло до країн Африки: Гаяна, Намібія та Танзанія тощо. Освітні програми, що базуються на шаховій грі, демонструють позитивний вплив на успішність учнів з математики, мов та інших предметів. Це все є частиною найбільш відомої ініціативи «Шахи в освіті», яку активно підтримує міжнародна шахова федерація (ФІДЕ). ЇЇ головна мета забезпечити доступ до шахової освіти для дітей у всьому світі та використати їх як один із засобів підвищення загальної якості освіти [20]. Також сучасні цифрові платформи, такі як Lichess та Chess.com, активно використовуються в освітньому середовищі. Вони дозволяють проводити онлайн-заняття, аналізувати партії, грати в турнірах, а також контролювати індивідуальний прогрес. Окрім академічних результатів, викладання шахів сприяє формуванню соціально значущих якостей, а саме: відповідальності, самодисципліни, чесності та вміння гідно приймати поразку. Завдяки своїй універсальності, шахи стали не лише інтелектуальною грою, а й потужним освітнім інструментом із доведеною ефективністю.

Отже, шахи є не лише давньою стратегічною грою, а й цінним освітнім інструментом, що сприяє розвитку мислення, пам’яті, уваги та інших корисних якостей. Історія розвитку шахів свідчить про їхню універсальність і здатність адаптуватися до соціальних змін і технологічного прогресу. Сьогодні шахи інтегруються в освітні програми по всьому світу, демонструючи високий потенціал у розвитку когнітивних здібностей учнів або студентів, а також у формуванні навичок необхідних для сучасного суспільства. Тому шахи мають значний потенціал для того щоб стати невід’ємною частиною освітньої системи.

1.3 Розвиток цифрових платформ для навчання та гри в шахи

Поява цифрових технологій відкрила нові горизонти для вивчення шахів. Першими такими платформами можна вважати шахові програми 1980-1990-х років Chessmaster і Fritz. Вони дозволяли грати проти комп’ютера на різних

рівнях складності, а також аналізувати зіграні партії. Ці програми не можна назвати навчальними, проте деякі елементи самоосвіти є. Пізніше з'явилися перші онлайн-сервери, які дозволяли людям грати проти один одного через інтернет. Найвідомішими були Internet Chess Club та Free Internet Chess Server, вони також не мали повноцінного навчального функціоналу, проте знову додалися деякі елементи, які корисні при навчанні шахам, а саме перегляд партій гросмейстерів у реальному часі, бази даних та чат. Прогрес розвитку не зупинявся й поступово дійшов до сучасних платформ, якими користуються багато шахістів по всьому світу [21].

Однією з найвідоміших платформ є Chess.com, яка поєднує функції онлайн-гри, освітнього ресурсу та спільноти. Вона містить тисячі навчальних відео, вправ, тематичних тренувань, ігор проти комп'ютера з різною силою гри, а також функції для тренерів. Також користувачі можуть отримати миттєвий аналіз кожної зіграної партії, вивчати різні дебюти та виконувати інші цікаві вправи [22]. Аналогічною платформою є Lichess, проте вона має відкритий код, а також надає безкоштовний доступ до всіх функцій, на відміну від попередньої. Дана платформа має інтуїтивно зрозумілий інтерфейс, інструменти для створення навчальних класів, завдань або тематичних тестів. Викладачі можуть проводити заняття в реальному часі, слідкувати за прогресом учнів та створювати домашні завдання (рис. 1.5) [23].

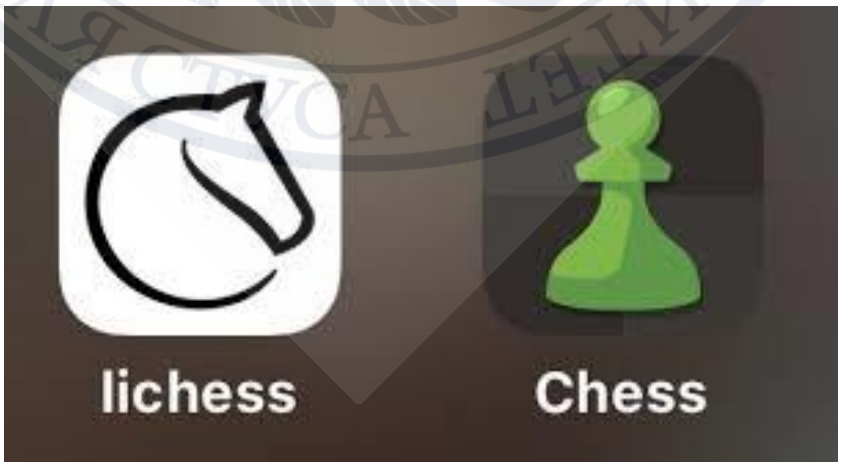

Рисунок 1.5 – Логотипи двох найвідоміших платформ для навчання шахам

Окрім цих двох найпопулярніших гігантів у шаховій сфері, існує ще багато менших платформ як за їх популярністю, так і ресурсами для навчання, які вони можуть надати потенційним учням. Розглядати їх окремо немає сенсу, оскільки вони мають схожі недоліки як вузька спеціалізація навчального контенту, фокусування більше на теоретичній частині або повністю платне навчання, яке не відповідає своїй вартості. Ці недоліки є основними для маленьких платформ, які поступово виправляються, але цього все рівно недостатньо, якщо порівнювати з Chess.com та Lichess. Проте не варто повністю забувати такі платформи, оскільки вони можуть і не надавати великий об'єм знань з шахів, але деякі теми або корисні поради можна знайти і на них, адже при досконалому розгляданні конкретної теми можна повністю визначити її особливості.

Цифрові шахові платформи мають багато переваг для учасників навчального процесу. Перш за все, це легка доступність, оскільки учні зі всього світу можуть отримати доступ до контенту без обмежень у часі або місці. По0друге, персоналізація навчання дозволяє адаптувати темп і складність матеріалів до рівня підготовки кожного учня. Більшість платформ пропонують індивідуальні рекомендації, що ґрунтуються на попередніх іграх, вирішених задачах, пройдених уроках та зроблених помилках. Також велике значення має інтерактивність, оскільки завдяки аналізу ходів в реальному часі, підказам зі стрілками та відеоурокам навчання стає більш ефективним і захопливим. Ще однією перевагою є миттєвий зворотній зв'язок. Система одразу повідомляє про неправильно зроблений хід, показує альтернативні варіанти й пояснює різні тактичні та стратегічні ідеї. Це все сприяє глибшому розумінню партій та усвідомленому аналізу позицій. Цифрові платформи також є гарним ресурсом для самоосвіти. Оскільки учні можуть самостійно обирати теми, тренувальні модулі, змагатися онлайн та відслідковувати власний прогрес.

Попри численні переваги, використання цифрових шахових платформ може супроводжуватися певними труднощами. Напевно, найголовнішою проблемою може стати те, що учень може просто не мати нормально доступу до інтернет з'єднання, або навіть не мати пристрою на якому можна скористатися

відповідними платформами. Ще однією важливою проблемою є підготовка викладачів, оскільки потрібно володіти одразу двома навичками, а саме: гарними шаховими знаннями, достатніми для того щоб донести матеріал до учня, а також навичками користування сучасними технологіями, оскільки це потрібно для роботи на цифрових платформах. Цікавою проблемою є те, що через прогрес шахових рушіїв, при аналізі партій на таких цифрових платформах учень може перестати думати, оскільки буде автоматично намальовані підказки і показані правильні ходи. У такій ситуації варто спочатку самостійно проаналізувати партію або позицію, а вже потім звертатися по допомогу до рушія.

Розвиток цифрових платформ для навчання гри в шахи суттєво трансформував підхід до шахової освіти, зробивши її більш доступною, персоналізованою та інтерактивною. Від перших комп'ютерних програм до сучасних цифрових платформ, таких як Chess.com та Lichess, вони перетворили шахи на захопливий та гнучкий навчальний процес, що враховує індивідуальні потреби кожного учня. Також вони стали потужним інструментом для викладання як у формальній, так і позашкільній освіті, сприяючи розвитку логічного мислення, аналітичних навичок і самостійності. Водночас ефективне впровадження таких рішень, потребує належної цифрової інфраструктури, кваліфікованих вчителів та збалансованого поєднання технологій із традиційними методами навчання.

РОЗДІЛ 2
ТЕХНОЛОГІЧНІ АСПЕКТИ РОЗРОБКИ НАВЧАЛЬНОГО ВЕБДОДАТКУ

2.1 Аналіз технологічних підходів та інструментарію для розробки вебдодатків

Розробка сучасного вебдодатка, зокрема для навчання гри в шахи, потребує ретельного вибору технологій, архітектурних рішень і інструментів, які зможуть забезпечити гарну функціональність, зручність використання та ефективність. Вебдодатком називається інтерактивна програма, яка працює у браузері та дозволяє користувачу виконувати дуже різноманітні дії, які можуть починатися простим переглядом інформації до складної взаємодії з різними сервісами та базами даних. Завдяки своїй універсальності та доступності з будь-якого пристрою, який має підключення до Інтернету, вебдодатки стали основною формою реалізації більшості сучасних цифрових сервісів, таких як освітні платформи, інтернет-магазини, соціальні мережі тощо.

Переважно існує два варіанти з яких частин може складатися вебдодаток, проте в обох варіантах є дві частини, які є обов’язковими, це клієнтська та серверні частини, також додатково до них можуть додавати підключення до бази даних. Створення вебдодатку без баз даних можливе в тому випадку, коли не потрібно зберігати суттєву кількість інформації, проте це відбувається дуже рідко, тому можна сказати, що наявність баз даних, майже, обов’язкова. Якщо розглядати, які ролі виконує кожна частина, то можна зробити це наступним чином. Клієнтська частина відповідає за інтерфейс користувача, відображення інформації та обробку взаємодії користувача з браузером. Серверна частина виконує всю бізнес-логіку вебдодатка, обробляє запити, забезпечує безпеку та передає дані як з бази даних при виконанні дій, так і назад до неї, коли потрібно щось зберегти. Сама база даних зберігає всю необхідну інформацію, а саме: облікові записи користувачів, основний контент вебдодатку, дані про дії користувача та інше [24].

Архітектура вебдодатків існує двох видів, які залежать від кількості HTML-сторінок. Одним з видів є Single Page Application (SPA), він характерний тим, що завантажується одна HTML-сторінка на якій зміна контенту відбувається без додаткового перезавантаження сторінки, тому такий метод може забезпечити плавну взаємодію з користувачем. Іншим видом є Multi Page Application (MPA), що навпаки складається з багатьох HTML-сторінок, які створюються окремо для кожного виду контенту, і завантажуються при переході на нову HTML-сторінку. Звичайно, ці два види не різняться тільки кількістю HTML-сторінок, у кожного з них є свої переваги та недоліки. Серед переваг SPA є висока швидкість роботи, інтерактивність і невелике навантаження на сервер. Проте додатки з такою архітектурою складніше оптимізувати для пошукових систем і вони можуть мати складнішу реалізацію безпеки. З іншого боку MPA має кращу підтримку пошукових систем, є простішим для реалізації контролю доступу до даних та зберігає стабільну поведінку при складній навігації, але він також має повільну взаємодію та більші витрати ресурсів при кожному переході на нову сторінку. Таким чином, можна дійти висновку, що переваги і недоліки одного виду архітектури, є протилежними якостями іншої. Тому вибір архітектури залежить від типу вебдодатку, складності функціоналу та вимог цільової аудиторії [25].

Для створення клієнтської частини вебдодатку використовуються такі основні технології:

- HTML – мова гіпертекстової розмітки, яка використовується для створення та структурування вмісту на вебсторінках. Вона дозволяє описувати структуру інформації використовуючи теги для визначення різних елементів, таких як заголовки, параграфи, зображення, різні посилання та інше. HTML постійно розвивається, і остання версія HTML5 включає нові функції, такі як робота з відео та аудіо, графічне малювання та інші семантичні елементи, які допомагають краще описувати вміст вебсторінок. Також остання версія HTML5 покращує інтерактивність та сумісність з мобільними пристроями, що дозволяє робити адаптивні дизайни [26].

- CSS – каскадні таблиці стилів, що відповідають за візуальне оформлення HTML-елементів. Вони дозволяють створювати більш привабливі дизайни, керуючи такими елементами, як колір тексту, шрифти, відступи та багато іншого. Остання версія CSS3 підтримує анімації, переходи, адаптивний дизайн через медіа-запити, змінні стилів і модульну структуру [27].
- JavaScript – динамічна мова програмування, яка виконується на боці клієнта в браузері та дозволяє реалізувати різноманітну інтерактивну поведінку як реагування на події, валідацію форм, маніпулювання DOM-структурою, асинхронну взаємодію з сервером та інше [28].

Рисунок 2.1 – Логотипи HTML5, CSS3 та JavaScript

Також допоміжними засобами, які набагато пришвидшать створення вебсторінок, є фреймворки, що застосовуються для CSS та JavaScript. Для CSS найпопулярнішим фреймворком є Bootstrap, який має дуже велику кількість різних функцій та компонентів. Його плюсами є швидка розробка з великою кількістю шаблонів, забезпечує стабільний дизайн незалежно від браузера, а також має багато ресурсів для вивчення та обговорення з іншими користувачами. Проте його головними мінусами є те, що багато дизайнів є однотипними, а також у його шаблонах є багато зайвих компонентів, які надмірно засмічують загальний код. Тому при використанні Bootstrap варто більш ретельно підходити до верстки вебсторінок, щоб був баланс між гарним виглядом та використанням ресурсів.

Для JavaScript також існує багато фреймворків, а найпопулярнішими є React, Vue та Angular. React є бібліотекою, яка була розроблена компанією Facebook, основна її мета є створення динамічних користувацьких інтерфейсів. Він забезпечує високу швидкодію та має, приблизно, середній поріг входження, проте використання деяких концепцій вимагає більш глибоко розуміння та досвіду користування даним фреймворком. Натомість Vue має трохи схожі характеристики з React як гарна продуктивність та зручну інтеграцію з іншими бібліотеками, проте Vue має набагато менший поріг входження та гарно підходить новачкам. Останнім з найпопулярніших фреймворків є Angular, він має більш розширений функціонал, ніж React та Vue, це може бути вбудовані рішення для маршрутизації, роботи з HTTP-запитами та ін'єкції залежностей. Проте через складну структуру та використання TypeScript поріг входження є набагато вищий, ніж в інших фреймворках. Тому для самостійного використання для створення невеликих проектів або додатків ватро використовувати React або Vue, а вже Angular краще підходить для роботи в команді над великими корпоративними проектами [29].

Таким чином щоб створити клієнтську частину вебдодатку варто використовувати HTML, CSS та JavaScript у поєднанні з відповідними фреймворками для пришвидшення створення вебсторінок, їх дизайну та логіки, які будуть гарно демонструвати контент користувачу.

Серверна частина відграє ключову роль у функціонуванні вебдодатка, оскільки вона відповідає за обробку запитів, збереження та обробку даних, автентифікацію користувача, логіку взаємодії з базою даних, а також іншу специфічну бізнес-логіку, яку потрібно реалізувати у вебдодатку. Вибір мови програмування та фреймворка для backend-розробки залежить від таких факторів як складність проекту, продуктивність та швидкість розробки. Оскільки фреймворки написані під певні мови програмування, тому їх потрібно обирати разом, і враховувати всі переваги та недоліки одного чи іншого компоненту.

Одним з таких варіантів є використання мови програмування JavaScript як для backend, так і для frontend частини. Для цього потрібно скористатися Node.js,

що є середовищем виконання JavaScript на стороні серверу і значно спростить розробку, а також завдяки асинхронній природі забезпечить доволі високу продуктивність під час обробки великої кількості одночасних запитів. Фреймворком можна обрати Express.js, що є мінімалістичним і гнучким при створенні вебсерверів та REST API, також він дозволить швидко реалізувати авторизацію, обробку маршрутів та інтеграцію з базами даних, наприклад SQLite чи MongoDB.

Іншою мовою програмування, яка може бути використана для серверної частини є Python. Вона є одною з найпоширеніших мов програмування, оскільки є доволі простою в розумінні та читанні коду, а також має дуже велику кількість бібліотек. Python використовується в багатьох галузях, починаючи від аналізу даних та машинного навчання, а закінчуючи автоматизацією та веброзробкою. Оскільки дана мово програмування має велику кількість різноманітних бібліотек та вагому кількість фреймворків, варто обрати те, що гарно підходить для певних потреб [30]. Для веброзробки на Python найчастіше використовують два найпопулярніших фреймворка – Flask та Django. Flask є простим мікрофреймворком, що надає лише базовий набір можливостей, таких як маршрутизацію, роботу з HTTP-запитами, рендеринг шаблонів тощо. Він не потребує конкретної структури, дозволяючи розробникам самостійно обирати інструменти для роботи з базами даних, формами, авторизацією та іншим функціоналом, тому він часто використовується в проектах, які потребують гнучкість та контроль над архітектурою. Це є плюсом для навчальних додатків, оскільки це дозволяє більше зосередитися на логіці, а не на складній конфігурації. Також Flask популярний для побудови RESTful API, який потрібен для вебдодатків створених на JavaScript [31].

Проте Django навпаки є повноцінним фреймворком, який реалізує дуже велику кількість компонентів, а саме: вбудовану систему автентифікації, систему шаблонів, генерацію форм, має потужну ORM систему для взаємодії з базами даних, адміністративну панель для керування контентом та багато інших компонентів. Даний фреймворк є дуже гарним вибором для проектів, які

прагнуть отримати стабільну та масштабовану архітектуру з компонентами, які будуть готові до використання, а також де важливі безпека даних та контроль доступу до інформації. Підсумовуючи, можна сказати, що вибір між Flask та Django залежить від того, що потрібно вашому проекту, якщо потрібно вільний простір для рішень та легкість, то тоді варто обирати Flask, а якщо чіткі логіка та функціональність, тоді варто зупинитися на Django [31].

Третьою важливою частиною вебдодатка є бази даних, тому що збереження інформації про користувачів, контент та іншу дані потребує надійної системи керування базами даних (СКБД). Вибір конкретної СКБД залежить від складності структури даних, очікуваного навантаження на сервер, простоти розгортання та майбутнього масштабування. Бази даних поділяють на реляційні та нереляційні. Реляційні бази даних організовані у вигляді таблиць, які мають чітку схему, що складається з таких компонентів як назви стовпців, типи даних, первинні та зовнішні ключі, тому така схема дозволяє створювати складні зв'язки між об'єктами бази даних. Найпоширенішими реляційними СКБД є:

- PostgreSQL – потужна СКБД, яка підтримує складні запити, транзакції, розширення, збережені процедури, індекси та інше. Вона є відкритою та кросплатформною, а також активно використовується в масштабованих проектах. Тому її варто використовувати, коли завчасно відомо про великі обсяги даних [32].
- MySQl є одною з найпопулярніших і, водночас, найстаріших СКБД у світі. Хоча вона трохи поступається PostgreSQL за гнучкістю та можливостями, проте її досить легко налаштувати, і вона має гарну підтримку с багатьох хостингових середовищах. Також MySQl має форк MariaDB, що розробили після того як MySQl була придбана компанією Oracle у 2009 році. Оскільки над MariaDB працювали ті самі спеціалісти, що і над MySQl, то можна сказати, що вона є її більш новішою версією, оскільки підтримує сумісність і розширює функціонал проектів, де використовувалася MySQl.
- SQLite є вбудованою та легкою базою даних, яка зберігається у вигляді

одного файлу. Вона не потребує окремого сервера, тому є ідеальним варіантом для невеликих проектів, локальної розробки або вебдодатків із обмеженим навантаженням. Завдяки простоті розгортання SQLite часто використовується у навчальних і демонстраційних проектах [32].

Рисунок 2.2 – Логотипи найпопулярніших реляційних баз даних

Відмінною нереляційних баз даних, від реляційних, є те, що вони не вимагають фіксованої схеми. Тому нереляційні бази даних краще підходять для зберігання неструктурованих або напівструктурованих даних, або коли структура інформації часто зазнає змін. Вони забезпечують високу швидкодію при масштабуванні, проте можуть значно поступатися у складності запитів або транзакцій. Найпопулярнішими нереляційними базами даних є:

- MongoDB є базою даних документного типу, яка зберігає дані у форматі BSON, що є розширеним форматом JSON. Вона дозволяє зберігати вкладені структури, що є зручним для об’єктно-орієнтованого програмування. Також MongoDB легко інтегрується з JavaScript стеком, наприклад Node.js, тому вона є популярною у проектах, де важлива гнучкість структури даних.
- Firebase Realtime Database є хмарною базою даних від Google, що підтримує обробку даних в режимі реального часу, автоматичну синхронізацію

та масштабування. Вона чудово підходить для мобільних застосунків або односторінкових сайтів, але через те, що це є більш хмарним сервісом, ніж повноцінною базою даних, вона має деякі обмеження, коли це стосується запитів із складною логікою.

Підсумовуючи, можна дійти висновку, що вибір бази даних для проекту залежить не тільки від його нинішнього стану розробки, але і від майбутніх планів на нього. Тому для створення локального проекту варто обрати SQLite, а далі при можливому масштабуванні перейти на PostgreSQL.

Останнім кроком для початку створення вебдодатку є вибір додаткових інструментів, які допоможуть покращити та пришвидшити процес розробки. До таких інструментів належать система контролю версій та середовище розробки. В обох варіантах є безперечні лідери, а саме Git та Visual Studio Code. Git є найпопулярнішою розподіленою системою контролю версій, яка використовується в більшості проектах. Вона дозволяє створювати різні гілки в проекті, відновлювати старі версії, слідкувати за історією змін в проекті, а також контролювати доступ до усіх етапів розробки. А Visual Studio Code є найпопулярнішим середовищем розробки, оскільки це кросплатформний редактор від Microsoft, який має велику кількість розширень для різних мов програмування. Також він підтримує термінал, інтеграцію з Git, дозволяє працювати з різними базами даних, а ще має велику кількість тем оформлень та конфігурацій, що не тільки пришвидшить розробку, але і зробить її приємною для перегляду.

2.2 Функціоналі вимоги до навчального вебдодатка гри в шахи

Під час розгляду необхідних функціональних вимог начального вебдодатку гри в шахи варто поділити їх на дві групи. Перша буде стосуватися користувацького функціоналу, а друга безпосередньо самої шахової гри. Оскільки головна мета в першу чергу навчитися гри в шахи, то більше уваги варто приділити саме другій групі, адже вона є центром даного вебдодатку. До першої групи мають входити наступні елементи та функції як створення свого

аккаунта, перегляд власного профілю та статистики, а також можливість завантажити зіграну партію. До другої групи будуть належати всі правила гри в шахи та способи їх реалізації, а також можливість зіграти партію проти шахового рушія або провести аналіз партії.

Якщо розглядати першу групу більш детально, то вона має задовольняти дві основні характеристики, а саме: мати приємний вигляд та легкість у використанні. Стосовно реєстрації власного аккаунту, то можна обмежитися простим логіном і паролем, оскільки для навчальної програми не потрібно дізнаватися іншу інформацію про користувача, а також це зменшить навантаження на сервер, оскільки не потрібно буде обробляти зайву інформацію.

Профіль та власна статистика мають буди доступні кожному користувачу для самоконтролю навчального процесу. Основною інформацією, яка має бути представлена в профілі є те, які уроки проходив користувач, та скільки помилок він зробив під час проходження певного уроку. Це наддасть йому можливість побачити свої слабкі сторони та сформувати подальшу траєкторію навчання. Схожа ситуація зі зіграними партіями, які можна зіграти проти шахового рушія, а після завантаження партії можна буде пізніше проаналізувати її і звернути увагу на свої недоліки як в теоретичній частині, так і в практичній.

Починаючи розглядати другу групу функціональних вимог варто з шахівниці на якій будуть проходити всі основні події. Вона має квадратний розмір 8 на 8 клітинок чорного та білого кольору, які чергуються у відповідному порядку. Також на шахівниці мають бути розміщені фігури двох кольорів, і вони мають бути однакового стилю та масштабу, щоб не відволікати під час гри або проходження уроку.

Оскільки в даному вебдодатку можна не тільки навчатися, а й грати проти шахового рушія, то важливо зберегти всі правила гри, які впливають на неї. Першим і основним правилом є те, що має дотримуватися чергування ходів, тобто один хід роблять білі, а інший чорні, і так далі по черзі. Другим важливим правилом є те, що будь-який гравець не може здійснити хід на клітинку зайняту своєю фігурою. Він може здійснити таку дію тільки, якщо це фігура суперника, і

цей хід дозволений правилами за якими роблять ходи шахові фігури. Саме на ці правила потрібно зробити акцент, оскільки вони дуже впливають на попередні правила.

При визначенні правил для кожної фігури не враховується колір фігур, вони одні як і для білих, так і для чорних. Всього існує шість видів фігур: пішак, кінь, слон, тура, ферзь та король, і кожна з них має свої особливості. Пішакам властивий рух та взяття тільки вперед, і вони є єдиними, хто не може повернутися назад. Перший хід пішака може бути здійснений або на одну, або на дві клітинки вперед, але не зважаючи на обраний варіант першого ходу, другий і подальші ходи робляться на одну клітинку вперед. Взяття пішаком відбуваються по діагоналі на одну клітинку вперед і вправо, і вліво, якщо це передбачено рамками шахівниці. Дійшовши до останньої лінії пішак може перетворитися на будь-яку фігуру з 4 варіантів: ферзь, кінь, слон, тура. До цього списку не входять власне сам пішак, а також король, оскільки він має бути тільки один на дошці.

Слони можуть пересуватися по діагоналям на будь-яку кількість дозволених клітинок. На дошці є слони двох кольорів – чорний та білий, проте це не відноситься до їх характеристик як фігур, це визначає по яким клітинкам вони ходять, а саме по чорним чи по білим. На які саме клітинки має право ходити слон можна дізнатися подивившись на його стартову клітинку. Дуже схожими на слонів є тури. Вони не мають обмеження на колір, проте вони можуть ходити лише по горизонталям та вертикалям, на всі можливі клітинки. Якщо вести мову про правила ходу для ферзя, то можна сказати, що його правила найлегші зі сторони програмування, оскільки вони є поєднанням правил слонів та тур. Ферзь може ходити як по горизонталі і вертикалі, так і по діагоналям [33].

Проте найцікавішими правилами володіє кінь, він є єдиною фігурою, яка може перестрибнути через іншу. Це все можливе, оскільки він пересувається «літерою Г», тому в такий спосіб він може ігнорувати позицію фігур, які розташовані поперед нього. Ось чому в початковій позиції існує двадцять можливих ходів, а саме шістнадцять пішаками та чотири конями. Але цікавим фактом є те, що найважливіша шахова фігура має найпримітивніші правила. Це

все через те, що король може ходити тільки на одну клітинку в усі сторони від себе. Можливо, його важливість показується в даних правилах, оскільки щоб не наражати його на небезпеку, він не може далеко відходити від своїх фігур. Саме через такі правила шахові пішакові ендшпілі стають переважно математичними задачками, адже там в прямому сенсі потрібно лише порахувати шлях до перемоги, оскільки і пішаки, і королі мають примітивні та короткі можливості ходів [33].

Розглядання базових правил ходів закінчено, проте варто приділити увагу ряду особливих ходів, які вимагають більш детального розгляду. Першим з них є рокірування – це хід, коли король на тура змінюються місцями, а саме король переміщується ближче до краю дошки, а тура навпаки – центр. Рокірування бувають двох видів: королівське та ферзеве. Їхню різницю можна зрозуміти з назви, а саме шахова дошка має вісім вертикалей, і по центральним стоять король та ферзь, звідси і назва двох флангів, там де король буде королівський фланг, а там де ферзь – ферзевий. Звідси і назва типів рокірувань, які виконуються в сторону відповідного флангу. Коли відбувається королівське рокірування для білих фігур, то король переміщується на дві клітинки праворуч, а тура на дві клітинки ліворуч, при ферзевому рокіруванні король йде на дві клітинки ліворуч, а тура на три клітинки праворуч. Для чорних фігур все відбувається за такими є правилами, проте все має бути дзеркально, тобто якщо потрібно піти вправо, то вони мають піти вліво, і навпаки. Проте є ряд умов при виконанні яких рокірування може бути здійснене. Першою умовою є те, що між королем та турою не має бути жодної фігури як своєї, так і іншого гравця. Другою умовою є те, що ні король, ні поле на яке він переміститься, ні поле через яке він пройде не мають бути атаковані фігури іншого гравця. Третьою умовою є те, що до моменту рокірування ні тури, ні король не мають робити будь-які інші ходи, проте є одна маленька деталь, що якщо походить тура, то рокірування буде не можливе тільки в сторону цієї тури, але якщо походить король, то рокірування неможливо буде здійснити в обидві сторони [33].

Іншим особливим ходом є взяття на проході. Його сутність полягає в тому, що коли пішак робить свій перший хід на дві клітинки вперед, то пішак суперника може здійснити взяття пішака на тій клітинці, яку перший пішак пропустив. Цей хід можна зробити тільки один раз тоді, коли суперник походив пішаком на дві клітинки вперед, якщо ви відразу не заберете або просто не можна буде зробити даний хід, то іншим ходом у вас вже не буде можливості здійснити взяття на проході [33].

Далі згідно правил існує ряд особливих позицій. Першою такою позицією є шах. Під час шаху король знаходиться під атакою фігури суперника. Коли виникає така позиція, то є три шляхи вирішення. По-перше, ми можемо просто забрати атакуючу фігуру. По-друге, можна поставити власну фігуру між королем та атакуючою фігурою, але дана дія можлива лише за умови, якщо атакуючою фігурою є не кінь. По-третє, залишається відійти королем з під шаху на будь-яку можливу клітинку. Другою особливою позицією є зв'язка – це позиція, яка створюється як наслідок після другої дії після шаху, коли між атакуючою фігурою та королем знаходиться захищаюча фігура. Саме тоді захищаюча фігура не може зробити хід, оскільки король буде знаходитися під шахом. Третя та четверта особливі позиції виникають під час завершення гри, це мат та пат. Мат виникає тоді, коли король знаходиться під шахом і не можна задіяти ні один з трьох шляхів для вирішення проблеми, саме тоді сторона, яка поставила мат стає переможцем. Пат є протилежністю мату, коли він настає, то сторона, якій поставили пат не може здійснити будь-який можливий хід при тому, що її король не знаходиться під шахом. Після того як виникає пат, партія завершується в нічию [33].

Окрім фіксації пату є ще декілька варіантів, коли партія може закінчитися у нічию без згоди обох сторін. Першим варіантом такої нічиї є трьохразове повторення позиції. Воно виникає тоді, коли позиції повторилася три рази. Другим варіантом є не виконання ходів пішаком або будь-якого взяття на протязі п'ятдесяти ходів. Це правило дуже схоже по своєму змісту на попереднє, хоча тут позиція може не повторюватися, проте є одна маленька деталь. Хід пішака та будь-яке взяття безповоротно змінює позицію, тобто якщо ці дві умови не

виконувати, то сама структура позиції не змінюється, що і призводить до нічиї. У реальних шахах для того щоб скористатися цими правилами потрібно покликати суддю, а реалізувати це у вебдодатку можна автоматично.

Тому дотримання даних вимог дозволить забезпечити всі основні потреби для функціонування додатку та вирішення основної мети, а саме навчанню гри в шахи.

2.3. Алгоритмічні підходи до реалізації шахового рушія та інтерактивного навчання

Шаховий рушій є програмою, яка здатна аналізувати шахову позицію, прораховувати можливі ходи, перевіряти їх на правила, а також приймати рішення в грі проти користувача. Щоб створити такий рушій для навчального додатку не обов'язково створювати складну аналітичну систему, яка буде схожа на професійний шаховий рушій, який використовують професіонали для аналізу. Проте базовий аналіз ходів та відповідь від рушія мають бути переконливими та логічно обґрунтованими. На базовому рівні шаховий рушій реалізується як набір функцій, які аналізують стан дошки та генерують допустимі ходи згідно правилам гри. Структура шахової дошки має бути представлена у вигляді двомірного масиву або відповідного запису, який базується на форматі Forsyth-Edwards Notation (FEN). FEN дозволяє зберігати шахову позицію в компактному вигляді рядка, в якому закодовано розташування фігур, черга ходу, можливість проведення рокіровки, взяття на проході та інші елементи [34].

Основним алгоритмом, який використовується для прийняття рішень шаховим рушієм у спрощених системах, є мінімакс, який прораховує можливі ходи обох гравців по черзі. Алгоритм працює за принципом максимізації свого виграшу та мінімізації виграшу суперника при цьому розглядаючи всі можливі ходи. Проте даний алгоритм не є ефективним, оскільки він переглядає всі можливі ходи, а це потребує дуже великої кількості ресурсів. Щоб не було повного перебору ходів, застосовують покращену версію мінімаксу, а саме алгоритм альфа-бета відсікання (рис. 2.3). Він дозволяє відсікати гілки дерева

можливих ходів, які вже не можуть покращити відомий результат, тому прибирання завідомо поганих варіантів значно пришвидшує розрахунки [35-36].

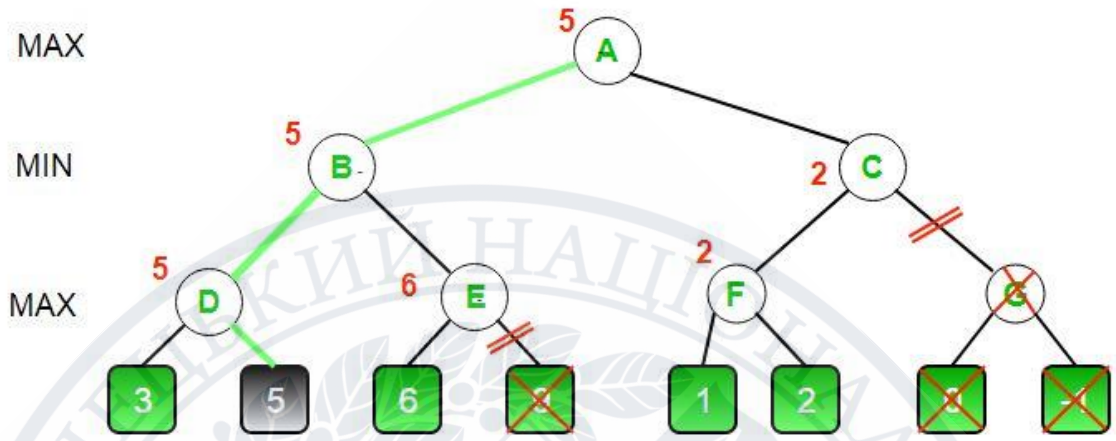

Рисунок 2.3 – Алгоритм альфа-бета відсікання

У простому навчальному додатку вистачить реалізувати глибину аналізу на один чи два ходи вперед, що дозволить створити відчуття інтелектуальної гри, а також не перевантажувати сервер. Для оцінки позиції всі фігури отримують власну вагу, яка буде еквівалентна кількості пішаків при оцінці людьми, а саме: пішак буде мати вагу 1, кінь та слон – 3, тура – 5, ферзь – 9. Проте також можна додати до методів оцінки такі критерії як контроль центру, мобільність фігур та безпека короля.

Якщо говорити про реалізацію навчання, то варто приділити увагу підтримці формату Portable Game Notation (PGN). У PGN можна записати хронологію ходів у текстовому вигляді разом із метаданими: гравці, результат партії, дата та можливі коментарі. Навчальні уроки можуть бути побудовані на PGN-фрагментах, де учень має виконати дію, яка є правильною за тематикою уроку. Тому для цього має бути виконаний правильний механізм перевірки правильності ходу, оскільки десь може бути один правильний хід, а в інших завданнях їх може бути декілька. Через це варто передбачити різні моменти, а при неправильному ході давати підказки, які допоможуть учневі знайти правильне рішення [37].

Інтерактивність навчального процесу досягається завдяки сценаріям уроків, які задаються як послідовність позицій і ходів. Такий підхід дозволяє реалізувати як пасивне навчання, коли відбувається перегляд теоретичних позицій, так і активне навчання, коли учень виконує завдання, а система перевіряє його правильність. Для цього використовується frontend-компонента, який візуалізує позицію, і backend, який зберігає логіку уроку, перевіряє правильність відповіді, підраховує прогрес та веде статистику користувача. Це дозволить слідкувати за навчальним планом користувача, а саме: коригувати його складність, повторно проходити уроки, які були з помилками, рекомендувати наступні уроки, спираючись на вже пройдених.

Підсумовуючи, можна сказати, що навчальний вебдодаток для гри в шахи має забезпечити ефективне поєднання гейміфікації, інтерактивності та поступового ускладнення. Технологічна реалізація цього підходу тісно пов’язана з алгоритмами контролю прогресу, генерації адаптивних сценаріїв, реакції на дії користувача та забезпечення візуальної привабливості інтерфейсу. Таким чином, алгоритмічні підходи до реалізації шахового рушія та інтерактивного навчання відіграють основну роль при створенні ефективного та мотивуючого середовища для користувача.

РОЗДІЛ 3
РОЗРОБКА НАВЧАЛЬНОГО ВЕБДОДАТКА ДЛЯ ГРИ В ШАХИ

3.1 Програмна реалізація серверної та клієнтської частини вебдодатка

Вебдодаток побудований за принципом трьохрівневої архітектури, де є клієнтська частина, серверна частина та база даних. Клієнтська частина працює в браузері користувача та відповідає за інтерфейс та зручність використання додатком. Серверна частина обробляє запити, відповідає за логіку гри та авторизацію, а також працює з базою даних. База даних виконує свою основну роль – це зберігання інформації, а також демонстрація її на вимогу серверу.

Для реалізації серверної частини вебдодатку основною мовою програмування було обрано Python. Це рішення має такі переваги, як гарне поєднання backend частини з frontend частиною, а також Python широко використовується для штучного навчання, що гарно підходить для створення простого шахового рушія. Оскільки основною мовою програмування було обрано Python, то Flask є гарним фреймворком для створення REST API [38]. Стосовно клієнтської частини було обрано стандартний стек HTML, CSS та JavaScript, що дозволить зробити гарний інтерфейс та інтерактивність. Взаємодія між серверною та клієнтською частинами була створена за допомогою вбудованої функції JavaScript, а саме Fetch API, що дозволить обмінюватись даними між обома частинами без перезавантаження сторінки [39]. Серед баз даних була обрана SQLite, оскільки вона гарно підходить для локального зберігання даних при реалізації невеликих проектів.

Серверна частина має виконувати дуже багато функцій, а саме: уся логіка шахової гри, така як правила ходів фігурами, черга ходу, стан гри та багато іншого; реєстрація та авторизація користувачів, збереження та отримання даних про шахові партії, уроки та їх прогрес; взаємодія з користувачем, а також серверна частина має взаємодіяти з базою даних, щоб правильно користуватися всіма необхідними даними, які зберігаються в ній.

Щоб розділити функції за метою їх використання, потрібно створити ряд файлів, які допоможуть легше орієнтуватися під час розробки (рис. 3.1).

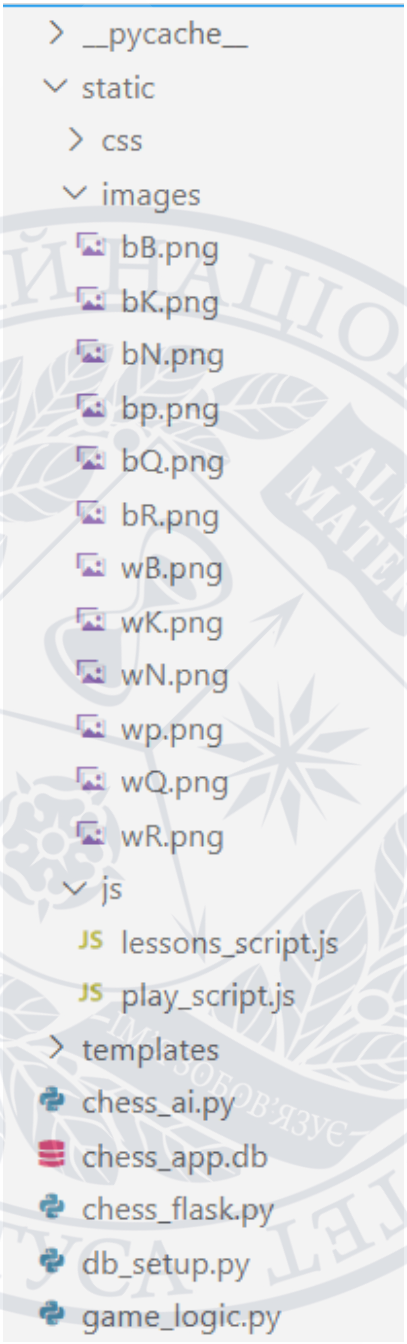

Рисунок 3.1 – Перелік файлів, що відповідають за логіку вебдодатка

Файл game_logic.py відповідає за всю логіку гри в шахи, а саме:

- шахівницю та фігури, які розташовані на ній;
- функція для ходу будь-якою фігурою;

- функція для рокірування;
- функція для перевірки всіх ходів (рис. 3.2);
- функція для перевірки шаху (рис. 3.2);
- окремі функції для ходів кожної фігури з відповідними правилами (рис. 3.3);
- функція для отримання шахової нотації.

```
# перевірка чи не знаходиться король під шахом
def inCheck(self):
    if self.whiteToMove:
        return self.squareUnderAttack(self.whiteKingLocation[0], self.whiteKingLocation[1])
    else:
        return self.squareUnderAttack(self.blackKingLocation[0], self.blackKingLocation[1])

# перевірка всіх ходів
def getAllPossibleMoves(self):
    moves = []
    for r in range(len(self.board)):
        for c in range(len(self.board[r])):
            turn = self.board[r][c][0]
            # перевіряємо обрану фігуру
            if (turn == "w" and self.whiteToMove) or (turn == "b" and not self.whiteToMove):
                piece = self.board[r][c][1]
                self.moveFunctions[piece](r, c, moves)
    return moves
```

Рисунок 3.2 – Всі можливі ходи та перевірка на шах

```
# метод для слона
def getBishopMoves(self, r, c, moves):
    directions = ((-1, -1), (-1, 1), (1, -1), (1, 1))
    enemyColor = "b" if self.whiteToMove else "w"
    for d in directions:
        for i in range(1, 8):
            endRow = r + d[0] * i
            endCol = c + d[1] * i
            if 0 <= endRow < 8 and 0 <= endCol < 8:
                endPiece = self.board[endRow][endCol]
                if endPiece == "--":
                    moves.append(Move((r, c), (endRow, endCol), self.board))
                elif endPiece[0] == enemyColor:
                    moves.append(Move((r, c), (endRow, endCol), self.board))
                    break
                else:
                    break
            else:
                break
```

Рисунок 3.3 – Функція для ходів слоном (офіцером)

Файл chess_ai.py відповідає за простий шаховий рушій, який створений за допомогою алгоритму альфа-бета відсікань з глибиною в чотири ходи. Тобто прораховується дві пари ходів від обох сторін, де останнім ходом є сторона, яка ходила другою. Такий простий шаховий рушій гарно підходить для новачків, оскільки вони ще не можуть прораховувати позицію на багато ходів наперед.

Файл db_setup.py запускається один раз для того щоб створити базу даних, яка має наступну схему (рис. 3.4):

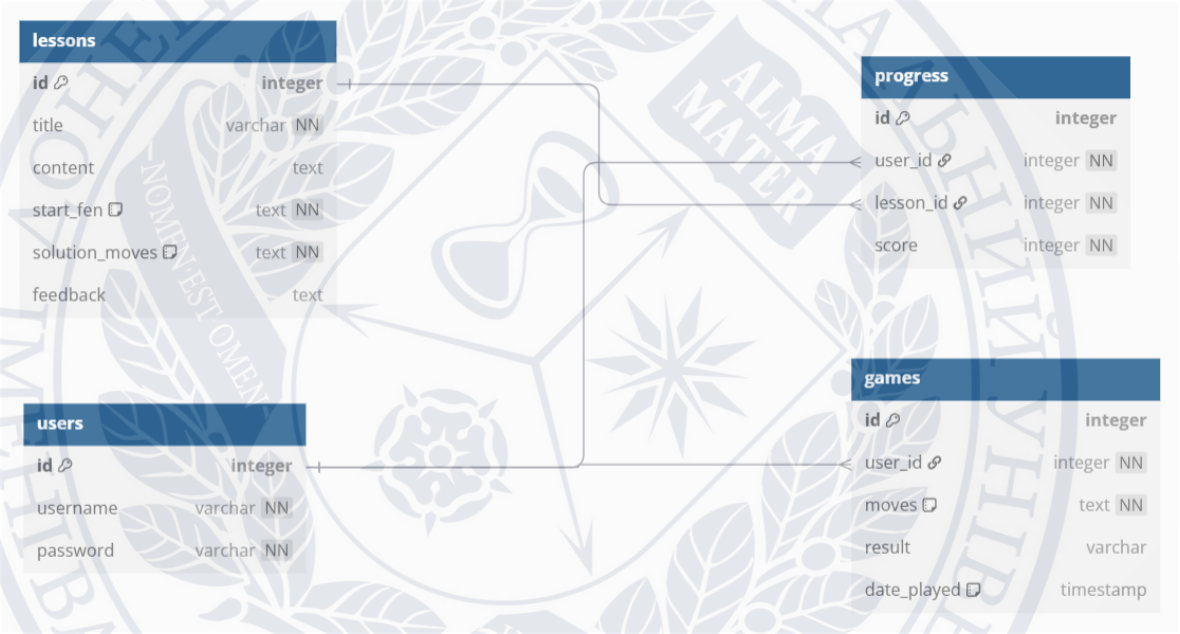

Рисунок 3.4 – Схема бази даних

Саме файл chess_app.db є цією базою даних, яка складається з чотирьох стовпців: користувачі, уроки, прогрес та зіграні партії. З цих стовпців три, а саме користувачі, прогрес та зіграні партії, автоматично заповнюються після відповідних дій користувача. А щоб додати новий урок до таблиці, треба виконати відповідний SQL запит на сервері.

Файл chess_flask.py відповідає за всю серверну частину вебдодатка. Саме він забезпечує зв’язок між клієнтом та сервером, який здійснюється за

допомогою HTTP-запитів, що є основним механізмом обміну даними у вебдодатках [40]. Існує декілька типів запитів, які мають різні значення:

- GET використовується для отримання даних з сервера, наприклад, при вході в обліковий запис.
- POST використовується для надсилання нових даних на сервер, як відповідь користувача або результат проходження уроку.
- PUT або PATCH використовуються для оновлення існуючих даних, наприклад, при зміні прогресу уроку.
- DELETE використовується для видалення будь-яких даних, наприклад, при видаленні облікового запису.

Дані, що викликаються запитами, передаються у форматі JSON, який є легким та зручним для обробки та зчитування програмою. Ці дані передаються по REST API-маршрутам, кожен з яких є ізольованим, що спрощує їх використання [41]. Ось деякі REST API-маршрути, що стосуються шахів, які будуть реалізовані у вебдодатку:

- /play_chess – відповідає за сторінку на якій буде відбуватися процес гри в шахи проти комп’ютера, або аналіз гри з його допомогою.
- /board – відповідає за все, що відбувається на шаховій дошці, наприклад, черга ходу, мат або пат (рис. 3.5).
- /move – відповідає за хід користувача (рис. 3.6).
- /engine-move – відповідає за хід комп’ютера.

```
@app.route('/board')
def get_board():
    return jsonify({
        "board": gs.board,
        "turn": "white" if gs.whiteToMove else "black",
        "checkmate": gs.checkmate,
        "stalemate": gs.stalemate
    })
```

Рисунок 3.5 – REST API-маршрут для шахової дошки

```python
@app.route('/move', methods=['POST'])
def make_move():
    data = request.json
    try:
        sr = Move.ranksToRows[data['from'][1]]
        sc = Move.filesToCols[data['from'][0]]
        er = Move.ranksToRows[data['to'][1]]
        ec = Move.filesToCols[data['to'][0]]

        new_move = Move((sr, sc), (er, ec), gs.board)
        valid_moves = gs.getValidMoves()

        if new_move in valid_moves:
            gs.makeMove(new_move)
            return jsonify({"message": "Хід виконаний"})
        else:
            return jsonify({"error": "Неможливий хід"}), 400
    except Exception as e:
        return jsonify({"error": str(e)}), 400
```

Рисунок 3.6 – REST API-маршрут для ходу користувача

Усі зображення шахових фігур розташовані в папці images, і мають назву в одному форматі – wK.png, де перша літера показує колір фігури (білий або чорний), друга її назву (король, ферзь, тура, слон, кінь та пішак), і одне із стандартних форматів для фото – png. Вони завантажується сервером на клієнтську сторону, коли це потрібно для гри або проходження уроків.

За логіку гри в шахи на клієнтській стороні відповідає файл play_script.js. Він виконує наступні функції:

- завантажує і відображає шахову дошку у браузері користувача;
- завантажує та відображає шахові фігури;
- реагує на дії користувача, а саме переміщення фігур мишею;
- оновлює стан дошки з сервера;
- виконує хід шахового рушія, який надіслано з серверу;
- зберігає гру для подальшого перегляду та аналізу;

- показує виконані ходи в окремому вікні;
- скидає гру після закінчення або за іншої потреби (рис. 3.7).

```
function resetGame() {
    fetch("/reset").then(() => {
        updateBoard();
        moveHistory = [];
        isWhiteTurn = true;
        clearMoveLog();
    });
}
```

Рисунок 3.7 – Функція скидання гри

Логіку виконання уроків на клієнтській стороні забезпечує файл lessons_script.js. Він виконує наступні функції:

- завантажує весь список уроків з серверу з допомогою відповідного маршруту;
- зчитує шахову позицію з FEN-рядка, який зберігається у базі даних, та відображає її на дошці, реалізація якого наведена в додатку А;
- показує інструкцію до виконання уроку;
- реагує на дії користувача, а саме перевіряє правильність ходу відповідно до умов уроку;
- надає зворотній зв’язок після виконання завдання;
- оновлює прогрес користувача після завершення уроку.

Також важливим елементом якому варто приділити особливу увагу є безпечне зберігання паролів. Можливо, для невеликого навчального додатку це не є якоюсь великою проблемою, але безпечне зберігання паролів важливе для повноцінних проектів. Тому варто почати використовувати заходи безпеки ще на стадії, коли проект є або локальним, або невеликим. Щоб забезпечити зберігання паролів можна використати модуль werkzeug.security, який є вбудований у Flask. У цьому модулі є функції, які дозволять хешувати паролі, а не зберігати їх у

відкритому вигляді [42]. Хешування працює таким чином, що пароль який вводить користувач під час реєстрації, перетворюється за допомогою спеціальних алгоритмів на набір символів, і зберігається в базі даних, а коли користувач хоче увійти у свій обліковий запис, введений ним пароль так само хешується, і вже порівнюється з тим, який був введений раніше. Перевагою хешування паролів є те, що ця дія є односторонньою, і відновити пароль лише за хешом неможливо, тому навіть у разі доступу третьої особи до бази даних, паролі користувачів залишаться у безпеці.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        password_hash = werkzeug.security.generate_password_hash(password)

        try:
            conn = sqlite3.connect(DB_PATH)
            cursor = conn.cursor()
            cursor.execute(
                "INSERT INTO users (username, password) VALUES (?, ?)",
                (username, password_hash)
            )
            conn.commit()
            conn.close()
            return redirect('/login')
        except sqlite3.IntegrityError:
            flash("Користувач з таким ім'ям вже існує")
            return redirect('/register')

    return render_template('register.html')
```

Рисунок 3.8 – Хешування пароля при реєстрації

Таке розподілення на серверну та клієнтську частину, а також базу даних, дозволить створити простий вебдодаток, який гарно підходить для початківців, що хочуть навчитися гри в шахи.

3.2 Програмна реалізація користувацького інтерфейсу та інтерактивних навчальних модулів

При розробці вебдодатка, який буде орієнтуватися на навчання початківців гри в шахи, якими в більшості випадків є діти, варто приділити увагу інтерфейсу та інтерактивності, що дозволить привернути увагу та зацікавити користувачів. Також щоб трохи гейміфікувати дизайн сторінок, варто додати різні іконки та позначки, що привернуть увагу користувача. Щоб реалізувати гарний користувацький інтерфейс та інтерактивність уроків було створено ряд файлів (рис. 3.9):

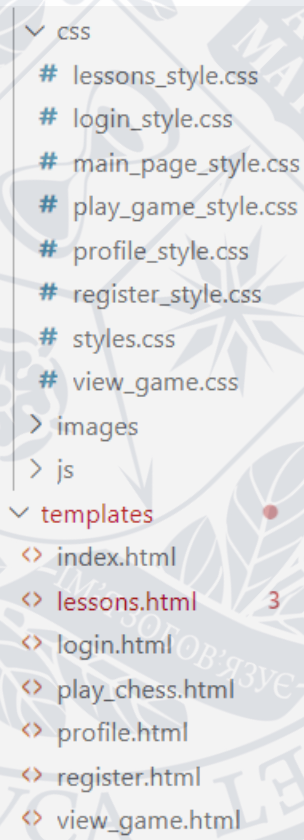

Рисунок 3.9 – Перелік файлів, які відповідають за оформлення вебдодатка

Було створено сім HTML-сторінок, а саме: головна, гра в шахи, уроки, вхід та реєстрація користувача, профіль та перегляд вже зіграних партій. Для них відповідно зі схожими назвами було створено сім CSS файлів, які відповідають за стилі кожної сторінки. Також є окремий файл styles.css, де знаходяться

загальний шаблон для всіх сторінок, який вже потім змінюється та підлаштовується під потреби кожної сторінки.

При вході у вебдодаток користувач потрапляє до головної сторінки – index.html. На ній він може побачити символічну назву вебдодатка, а також здійснити такі дії, як перегляж профілю, увійти або зареєструватися, почати грати в шахи, або пройти уроки (рис. 3.10):

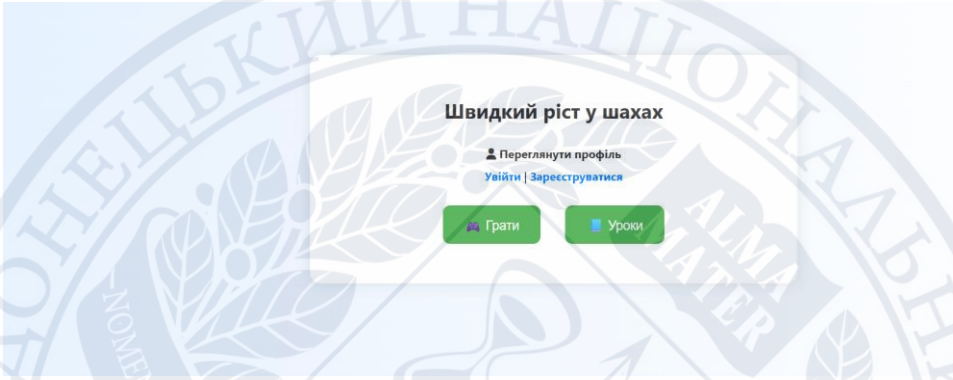

Рисунок 3.10 – Головна сторінка

Якщо користувач не авторизований будь-яка дія перенаправить його за маршрутом /login на сторінку входу, де він побачить відповідні поля вводу даних для здійснення входу у вебдодаток (рис. 3.11):

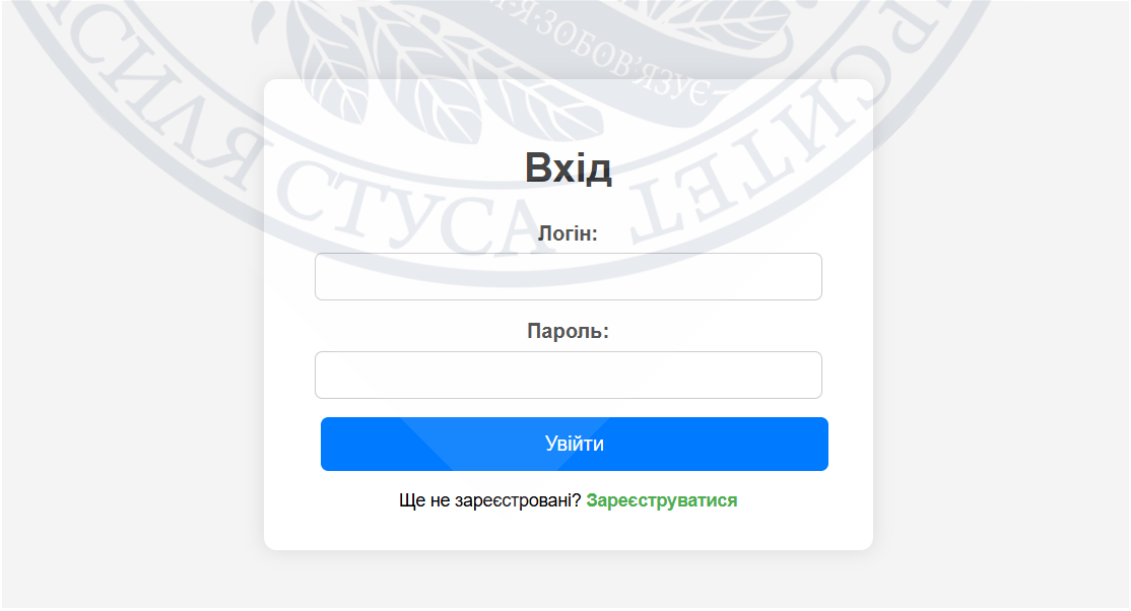

Рисунок 3.11 – Сторінка входу

На сторінці входу користувач має ввести свій логін та пароль, щоб продовжити користуватися вебдодатком, проте якщо він не зареєстрований, то йому потрібно пройти даний процес. Щоб здійснити реєстрацію користувачу потрібно перейти за відповідним посилання на сторінку реєстрації, після натиснення на неї сервер за допомогою маршруту /register перенаправить користувача на відповідну сторінку (рис. 3.12):

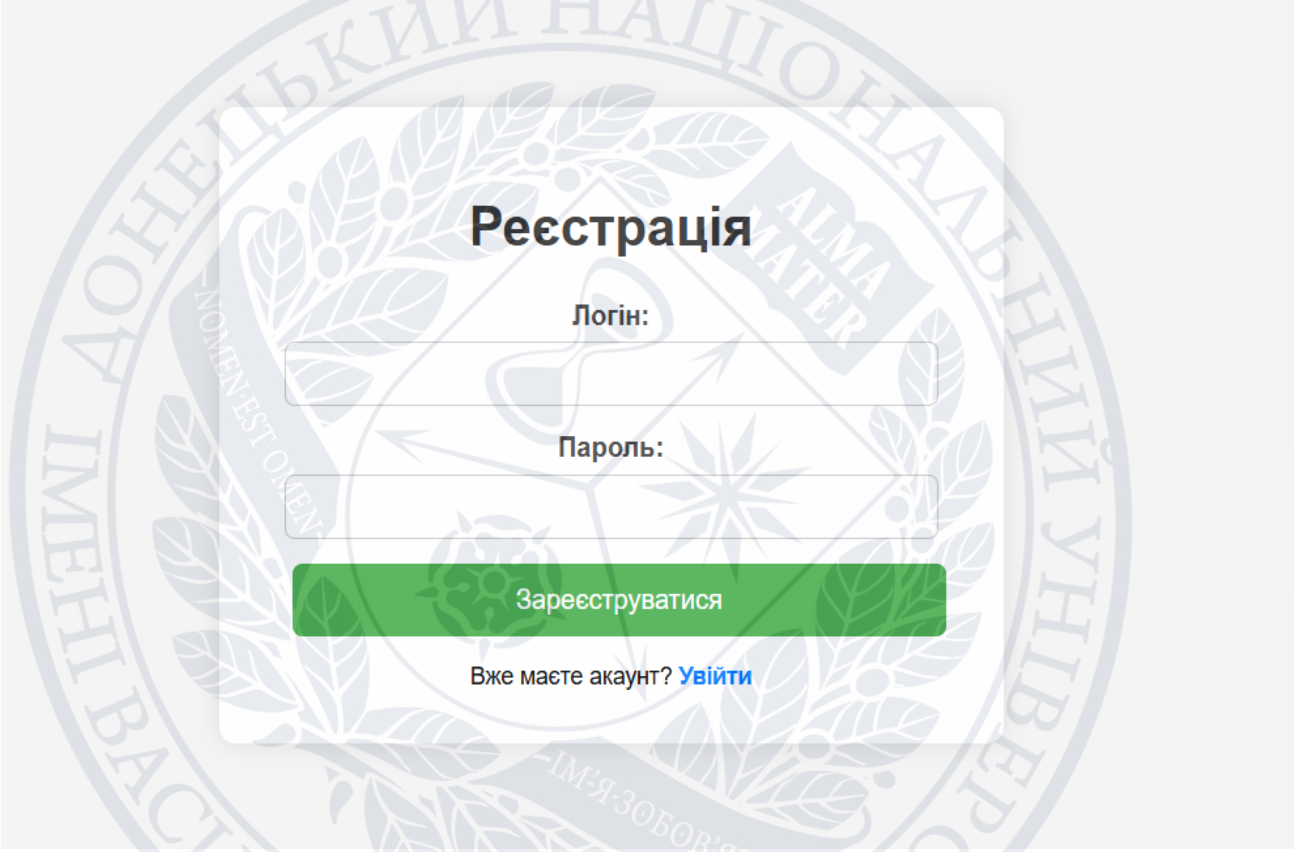

Рисунок 3.12 – Сторінка реєстрації

На сторінці користувач може побачити поля для вводу логіну та паролю. Після їх введення та натиснення кнопки «Зареєструватися», користувача одразу сервер перемістить на сторінку входу за відомим вже маршрутом /login. На даній сторінці, як зазначалося вище, при заповненні всіх відповідних полів та натиснення кнопки «Увійти», користувача буде перенесено на головну сторінку, де він вже зможе скористатися усіма функціями вебдодатку.

Щоб почати грати в шахи, або провести аналіз користувач повинен натиснути на відповідну кнопку «Грати», яка для кращого розуміння представлена разом із значком джойстика, який асоціюється з іграми (рис. 3.13):

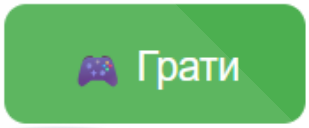

Рисунок 3.13 – Кнопка для переходу на сторінку з грою в шахи

Після натиснення на кнопку сервер перенесе користувача за маршрутом /play_chess на відповідну сторінку, де він зможе побачити шахову дошку з усіма фігурами, які малюються за допомогою відповідних функцій у файлі play_script.js. Також справа від дошки користувач зможе побачити поле в якому будуть відображені усі ходи. Під дошкою користувач побачить чергу ходу (хід білих або чорних) або стан гри (мат або пат). А під ним будуть три кнопки: нова гра, хід комп’ютера та зберегти гру (рис. 3.14):

Рисунок 3.14 – Сторінка для гри та аналізу

Щоб виконати хід на дошці, користувач повинен натиснути на обрану фігуру та здійснити хід, у разі виконання неможливого ходу, статус гри покаже підказку, що був здійснений неможливий хід, і користувачу потрібно зробити інший хід. Ходи користувача сервер приймає через маршрут /move, через аналогічні маршрути працюють і кнопки, які знаходяться під дошкою. Щоб комп'ютер виконав свій хід, користувач повинен натиснути на кнопку «Хід комп'ютера», і тоді по маршруту /engine-move буде виконаний даний хід на сервері, який наддасть відповідь у форматі JSON до браузера користувача, а відповідна функція з play_script.js відтворить даний хід на дошці. Аналогічно цьому відбуваються процеси при натисненні кнопок «Скинути гру» та «Зберегти гру» по маршрутам /reset та /save відповідно.

Якщо користувач забажає пройти уроки, то йому потрібно на головній сторінці натиснути кнопку «Уроки», яка за допомогою маршруту /lessons перемістить його на відповідну сторінку (рис. 3.15-3.16):

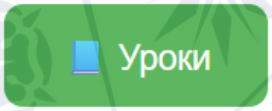

Рисунок 3.15 – Кнопка переходу до уроків

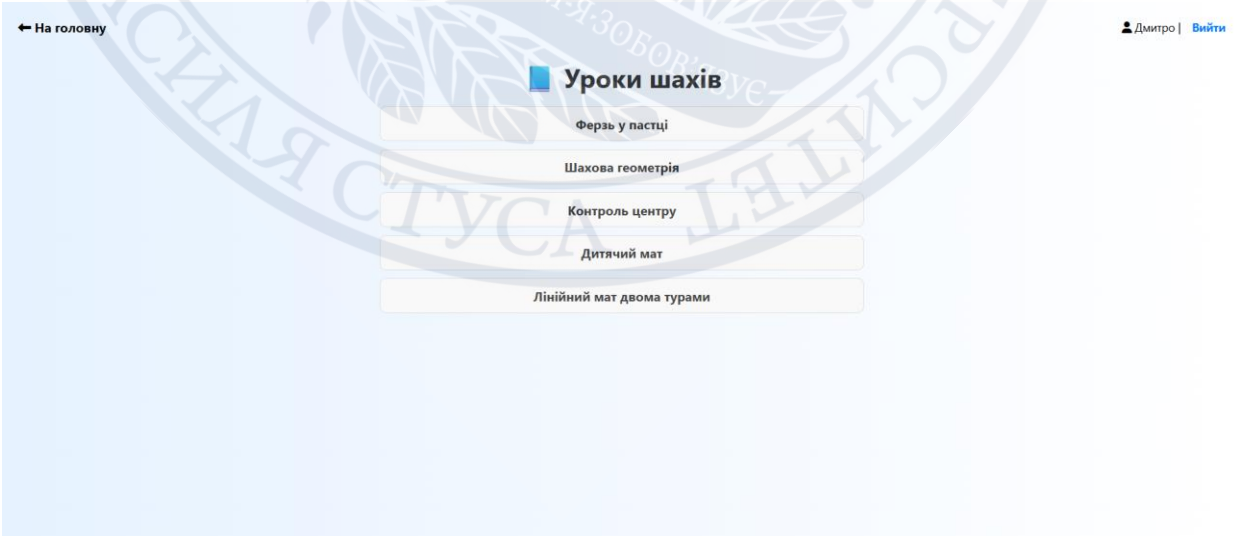

Рисунок 3.16 – Сторінка зі списком доступних уроків

Після того як користувач переміститься на сторінку з уроками, перед ним відкриється повний список уроків, який доступний йому для проходження. Список уроків зі всім їх вмістом дістається сервером з бази даних з допомогою маршруту /lessons/<int:id> та відправить потрібну відповідь до браузера користувача. Щоб переглянути урок та мати можливість його пройти, користувача має натиснути на назву уроку, і після цього перед ним відкриється дошка з позицією та завданням для уроку (рис. 3.17):

Рисунок 3.17 – Перегляд уроку

Для проходження уроку користувач має перетягнути обрану фігуру на потрібну клітинку, якщо хід буде не вірний, то спробу потрібно буде повторити. Спосіб перетягування фігури при проходженні уроків обраний навмисно, оскільки на відміну від звичайної гри в шахи, де користувач перед здійсненням ходу, його попередньою обміркував, і знайшов причину для обрання саме цього ходу, при проходженні уроків він цю дію робить уперше. Тому для кращої запам’ятовуваності краще буде зробити таку ситуацію, коли користувач змушений повністю відтворити хід, а не просто натиснути дві клітинки навмання.

Після проходження уроків та можливих збережених партіях, користувач може переглянути свою успішність у профілі. Щоб це зробити він має на головній сторінці натиснути на перегляд профілю, і тоді до сервер за допомогою маршруту /profile перенаправить користувача на сторінку з власним профілем (рис. 3.18):

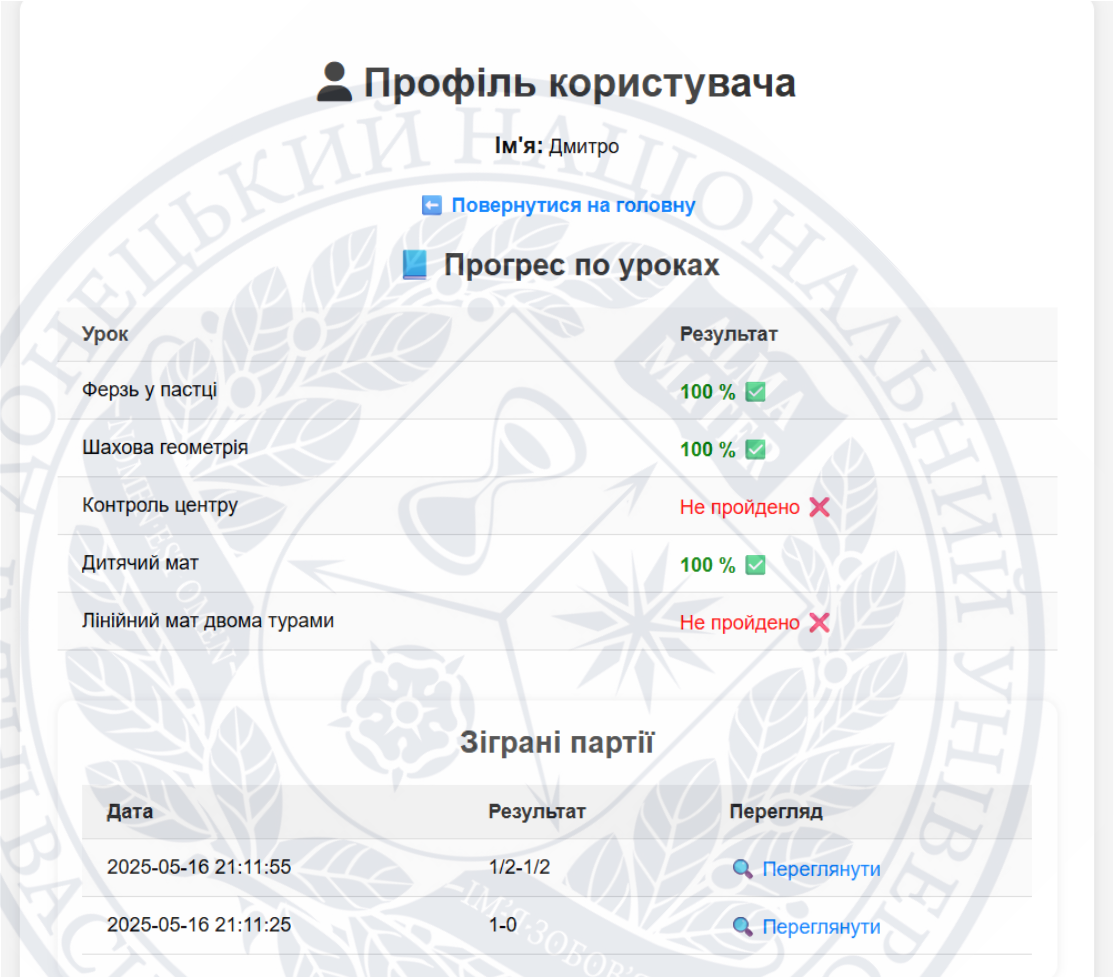

Рисунок 3.18 – Сторінка з профілем користувача

Під час перегляду свого профілю користувач може побачити кількість та прогрес пройдених уроків, що гарно підкреслено відповідною кольоровою гаммою та позначками. Окрім перегляду прогресу користувача також може побачити інформацію про свої зіграні та збережені партії, а саме дату, коли вона була зіграна, та результат з яким вона закінчилася. Можливість зберігати партії дасть змогу створити свою шахову мінібазу, яка може зберігати не тільки власні партії, а і партії інших гравців, які були проаналізовані як навчальний приклад.

Також дані партії можна не тільки зберігати, а і переглядати, для цього потрібно натиснути на відповідну кнопку, яка з допомогою маршруту /game/<int:id> перенаправить на відповідну сторінку (рис. 3.19):

Рисунок 3.19 – Перегляд збереженої партії

Перегляд партії здійснюється з допомогою відповідних кнопок: наступний та попередній ходи. Таким чином це дозволить повторювати деякі моменти з партії по декілька разів, що значно прискорить вивчення та засвоєння матеріалу.

Саме створення всіх потрібних сторінок з мінімалістичним дизайном, який вдало доповнений різними елементами, дозволить одночасно зосередити користувачів на вивченні шахів та зробить цей процес більш захопливим.

3.3 Тестування та оцінювання функціонування навчального вебдодатку гри в шахи

Під час тестування навчального вебдодатку гри в шахи варто перевірити чи дотримуються всі базові правила гри в шахи, чи правильно працює перевірка при

проходженні уроків, наявність підказок при неправильних діях в різних місцях вебдодатку, а також чи вірно та коректно працює передача даних від клієнта до сервера та навпаки.

Першим етапом тестування йде авторизація, а саме як відображаються помилки при неправильних діях користувача (рис. 3.20-3.21):

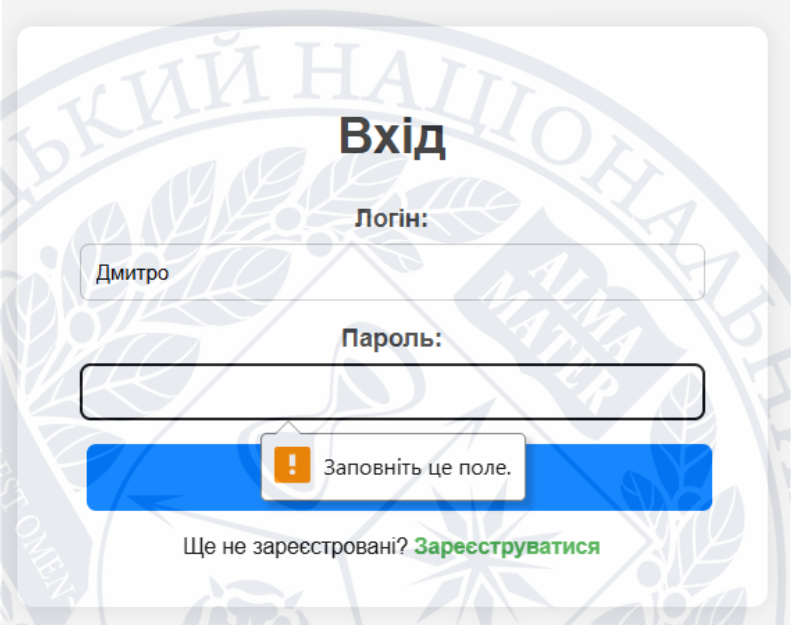

Рисунок 3.20 – Підказка при невведені пароля чи логіна

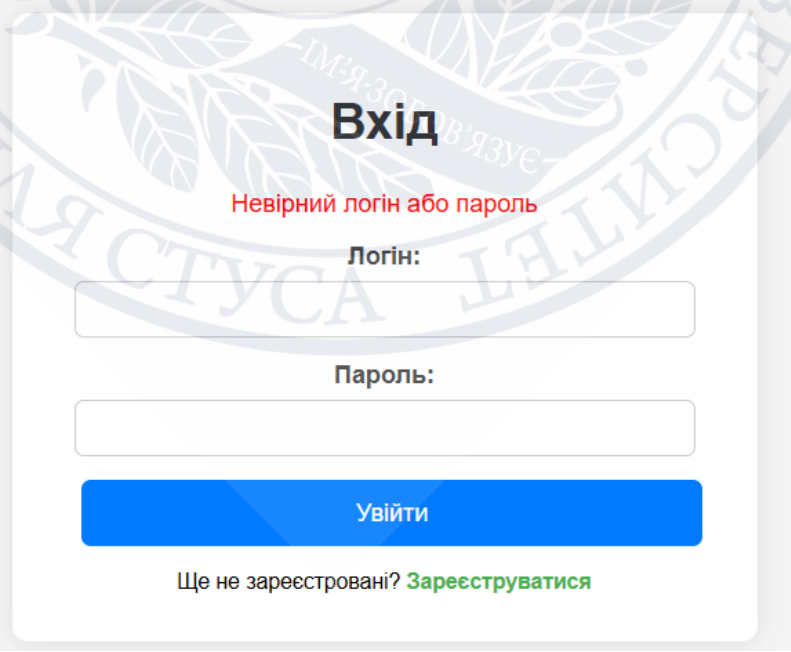

Рисунок 3.21 – Підказка при невірному логіні або паролі

Наступним етапом є перевірка шахової гри та відповіді шахового рушія (рис. 3.22):

Рисунок 3.22 – Перевірка правильності роботи шахового рушія

Наступним кроком потрібно перевірити правильність виконання всіх функцій при проходженні уроків (рис. 3.23):

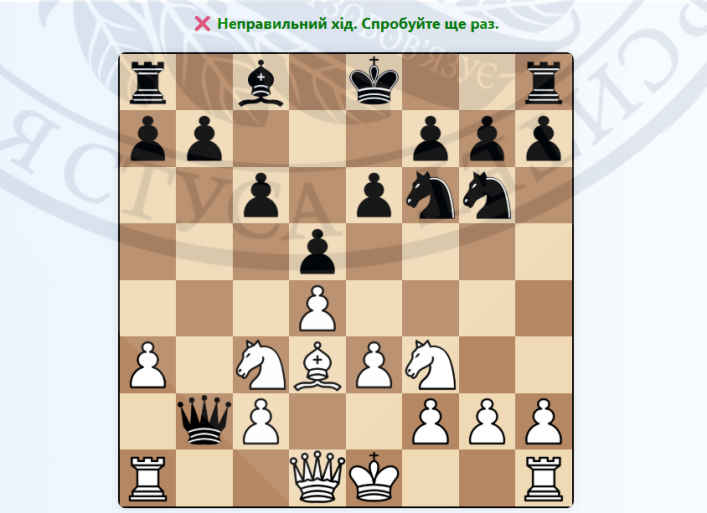

Рисунок 3.23 – Підказка при неправильному виконанні уроку

Останнім моментом, який потрібно перевірити чи оновлюється прогрес пройдених уроків та додаються нові партії у профілі (рис. 3.24):

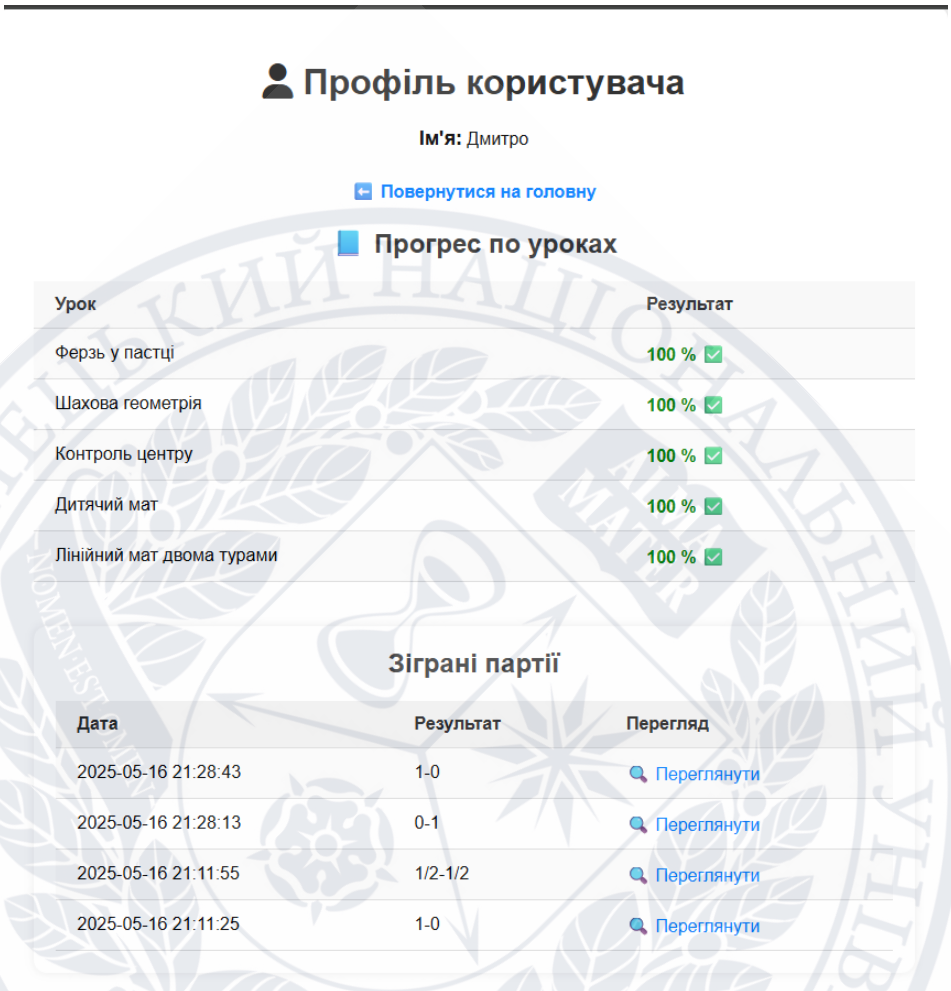

Рисунок 3.24 – Оновлення прогресу та поява нових партій

Під час тестування вебдодатку не було виявлено помилок при авторизації, у роботі шахового рушія, при проходженні уроків, оновленні прогресу та зберігання партій. Єдиним мінусом є невелика кількість уроків для проходження, але це не є вагомою проблемою, оскільки для початківців можливо створити сотні нових уроків за невеликий термін, оскільки всі вони будуть в один чи два ходи.

Таким чином вебдодаток повністю відповідає всім потребам і готовий до користування, а подальша робота над ним лише збільшить інтерес до нього.

ВИСНОВКИ

Під час виконання даної роботи було розроблено навчальний вебдодаток для гри в шахи, що надає можливість не тільки грати в шахи проти рушія, а також навчатися шахам при проходженні уроків та аналізі зіграних партій. За результатами роботи можна зробити наступні висновки:

1. Розглянуте поняття гейміфікації навчального процесу та які наслідки воно може нести, при застосуванні його до створення навчальних вебдодатків. Визначено орієнтацію гейміфікації в навчальному процесі на усунення таких проблем як-от: кількість винагород, складність завдань, змагальний тип навчання, особистий план навчання. Встановлено, що основними елементами гри, які потрібно додавати до навчального вебдодатку, щоб не відволікати користувача від його основної мети є: бали, бейджі, рівні, місії, винагороди.

2. Досліджено історію шахів та їх роль в формуванні стратегічного бачення аналізу ситуації. Проаналізовано складові їх популярності у різних галузях, з акцентом уваги на взаємодію шахів з навчанням. Це дозволило сформувати бачення предметної області для розробки навчального вебдодатку та конкретизувати постановку задачі для програмування.

3. Аналіз літературних джерел з технологій вебпрограмування дозволив визначити основні технологічні підходи та обґрунтувати вибір архітектури та стеку технологій для розробки вебдодатку. Проаналізовано основні недоліки та переваги таких типів архітектури вебдодатків як: Single Page Application та Multi Page Application.

4. Розроблено навчальний вебдодаток для гри в шахи, який має весь основний функціонал як гра в шахи проти рушія, збереження партій та проходження уроків за відповідною тематикою. При розробці додатку витримані основні функціональні вимоги, які були розділені на 2 групи: користувацького функціоналу та правил шахової гри. В якості імітаційного аналітика гри визначено та розроблено рушія гри, результат дій якого приймається за алгоритмом альфа-бета відсікання.

5. Архітектура вебдодатку включає базу даних, серверну та клієнтську частини. Інтерфейс вебдодатку розроблений з урахуванням потреб аудиторії, додавання мінімальних ігрових елементів, а також є простим та інтуїтивним, що допоможе учням та зацікавить їх при користуванні вебдодатком.

6. Проведене тестування вебдодатку було направлене на виявлення помилок як в логіці, що стосується перевірки шахових ходів та рішень, так і на взаємодію сервера з клієнтом і навпаки. У результаті тестування не було виявлено помилок, проте було знайдено резерви для покращення навчального процесу, а саме додавання більшої кількості уроків та розподіл їх за певними направленнями.

Отже, створений вебдодаток відповідає меті дослідження, а також реалізований з усім необхідним функціоналом для забезпечення гри та навчання шахам.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Янчук Р. Л. Гейміфікація як тренд освіти XXI століття. Сучасні цифрові технології та інноваційні методики навчання: досвід, тенденції, перспективи: матеріали VIII Міжнар. наук.-практ. інтернет-конф. (м. Тернопіль, 11-12 лист., 2021). Тернопіль: ТНПУ ім. В. Гнатюка, 2021. С. 48-50.

2. Саган О.В. Гейміфікація як сучасний освітній тренд. Збірник наукових праць. "Педагогічні науки". 2022. Вип. 100. С. 12-18. DOI: https://doi.org/10.32999/ksu2413-1865/2022-100-2

3. Кірк Б., Кренк Т. Презентація на Loyalty Expo2009. 2009. С. 6. URL: http://www.scribd.com/doc/17718638/Loyalty-Expo-2009-in-Review/ (дата звернення: 15.04.2025).

4. Горбенко Є. А. Особливості підбору та використання інструментів гейміфікації в освітньому процесі. Освіта. Інноватика. Практика: науковий журнал. Том 12, №7. Суми: СумДПУ ім. А. С. Макаренка, 2024. С. 29-35. DOI: https://doi.org/10.31110/2616-650X-vol12i7-004

5. Тимощук Г. В. Гейміфікація освітнього процесу: сучасний погляд. Наукові записки. Серія: Педагогічні науки 216. 2024. С. 328-332. DOI: https://doi.org/10.36550/2415-7988-2024-1-216-328-332

6. Заруцька В. О. Гейміфікація у освітньому процесі вищої школи: переваги та ризики. Міжнародні наукові дослідження: інтеграція науки та практики як механізм ефективного розвитку: матеріали III Міжнар. наук.-практ. конф. (м. Київ, 23-24 квітня, 2021). Київ. 2021. С. 21-30.

7. Грицюк О. С., Бережний Ю. О. Гейміфікація як сучасний засіб навчання в сучасному освітньому середовищі. Реформа освіти в Україні. Інформаційно-аналітичне забезпечення: збірних тез доповідей III Міжнар. наук.-практ. конф. (м. Київ, 28 жовтня, 2021). Київ. 2021. С. 175-177.

8. Констанкевич Л., Радкевич М., Лехіцький Т. Гейміфікація як інноваційний підхід в освітньому процесі. Нова педагогічна думка, №3. 2022. С. 47-51. DOI: https://doi.org/10.37026/2520-6427-2022-111-3-47-51

9. Щербина Д. С., Потапова Н. А. Гейміфікація навчального процесу у шахах для покращення сприйняття матеріалу. Прикладні інформаційні технології: матеріали VI Всеукр. наук.-практ. конф. (м. Вінниця, 22 травня 2025 року). Вінниця. 2025.

10. Davidson H. A. A Short History of Chess. Crown. 2012. 176 с.

11. History of Chess | From Early Stages to Magnus. URL: https://www.chess.com/article/view/history-of-chess (дата звернення: 16.04.2025).

12. Chess Clocks: An Introduction. URL: https://www.chess.com/article/view/an-introduction-to-chess-clocks (дата звернення: 16.04.2025).

13. The Man Who Built The Chess Machine. URL: https://www.chess.com/article/view/the-man-who-built-the-chess-machine (дата звернення: 16.04.2025).

14. Chole V., Gadicha V., Thawakar M. Evolution of artificial intelligence through game playing in chess. In: GHONGE, M. M. et al. (eds.). Data-driven systems and intelligent applications. Boca Raton: CRC Press, 2024, C. 119-136

15. Kuipers M., Prasad R. Journey of artificial intelligence. Wireless Pers Commun. 2022, 123, 3275–3290. DOI: https://doi.org/10.1007/s11277-021-09288-0

16. Adventures with endgame tablebases. URL: https://www.chess.com/blog/Rocky64/adventures-with-endgame-tablebases (дата звернення: 17.04.2025).

17. Nalimov EGTB URL: https://www.k4it.de/index.php?topic=egtb&lang=en (дата звернення: 17.04.2025).

18. Majhi S.G. ‘A brave new world’: Exploring the implications of online chess for the sport post the pandemic. In: Basu, B., Desbordes, M., Sarkar, S. (eds) Sports Management in an Uncertain Environment. Sports Economics, Management and Policy, vol 21. Springer, Singapore. 2023. C. 255-270. DOI: https://doi.org/10.1007/978-981-19-7010-8_11

19. Islam A., Lee W. S., Nicholas A. The effects of chess instruction on academic and non-cognitive outcomes: Field experimental evidence from a developing country.

Jornal of Development Economics, vol 150. 2021. DOI: https://doi.org/10.1016/j.jdeveco.2020.102615

20. FIDE Education Commission continues its development activities around the world. URL: https://www.fide.com/fide-education-commission-continues-its-development-activities-around-the-world/ (дата звернення: 18.04.2025).

21. Black J., Cochran M., Gardner R. A security analysis of the Internet Chess Club. IEEE security & privacy, vol. 4. 2006. С. 46-52. DOI: https://doi.org/10.1109/MSP.2006.2

22. Навчання на Chess.com. URL: https://www.chess.com/learn (дата звернення: 19.04.2025).

23. Навчання на Lichess. URL: https://lichess.org/learn (дата звернення: 19.04.2025).

24. Непийвода М., Рекало О., Мичуда Л., Коробейнікова Т. Web applications technological stack. Modern engineering and innovative technologies. 1. № 22-01. 2022. С. 104-112. DOI: https://doi.org/10.30890/2567-5273.2022-22-01-038

25. Sireteanu N.-A., Homocianu D. Front-End Frameworks for Development of SPA and MPA Web Applications. Department of Accounting, Business Information Systems, and Statistics, Faculty of Economics and Business Administration, Alexandru loan Cuza University, Carol I Blvd., no. 22. 2021. С. 4-7. DOI: https://dx.doi.org/10.2139/ssrn.3987838

26. HTML Tutorial. URL: https://www.geeksforgeeks.org/html-tutorial/ (дата звернення: 20.04.2025).

27. CSS Tutorial. URL: https://www.geeksforgeeks.org/css-tutorial/ (дата звернення: 20.04.2025).

28. JavaScript Tutorial. URL: https://www.geeksforgeeks.org/javascript/ (дата звернення: 20.04.2025).

29. Tadi S. Expanding Web Development Horizons: Integrating WebAssembly with React, Vue, and Angular. Journal of Scientific and Engineering Research. vol. 8. 2021. С. 250-261

30. Python Tutorial. URL: https://docs.python.org/3/tutorial/index.html (дата звернення: 21.04.2025).

31. Fuior, F.. Introduction in Python frameworks for web development. Romanian Journal of Information Technology and Automatic Control-Revista Romana De Informatica Si Automatica. vol. 31, no. 3 .2021. С. 97-108. DOI: https://doi.org/10.33436/v31i3y202108

32. Koch S., Wessels M., Klein D., Johns M., Poster: The risk of insufficient isolation of database transactions in web applications. ACM SIGSAC Conference on Computer and Communications Security. 2023. С. 3576-3578. DOI: https://doi.org/10.1145/3576915.3624394

33. FIDE laws of chess. URL: https://handbook.fide.com/chapter/E012023 (дата звернення: 22.04.2025).

34. Wijayanto, F. Forsyth-Edwards Notation in Chess Game Clustering: A Depth-Based Evaluation. Jurnal Sains, Nalar, Dan Aplikasi Teknologi Informasi. vol. 4, no. 1. 2025. С. 18–25. DOI: https://doi.org/10.20885/snati.v4.i1.3

35. Parashar A., Jha A. K., Kumar M. Analyzing a chess engine based on alpha–beta pruning, enhanced with iterative deepening. Expert Clouds and Applications. vol. 444. Springer. 2022, С. 691–700. DOI: https://doi.org/10.1007/978-981-19-2500-9_51

36. Щербина Д. С., Потапова Н. А. Застосування алгоритму альфа-бета відсікання для оптимізації шахових рішень. Комп'ютерні технології обробки даних: матеріали V Всеукр. наук.-практ. конф. (м. Вінниця, 6 грудня 2024). Вінниця. 2024.

37. Wijayanto, F. Clustering Analysis of Chess Portable Game Notation Text. Jurnal Sains, Nalar, Dan Aplikasi Teknologi Informasi. vol. 3, no. 3. 2024. С. 137–142. DOI: https://doi.org/10.20885/snati.v3.i3.42

38. Документація Flask. URL: https://flask.palletsprojects.com/en/stable/ (дата звернення: 22.04.2025).

39. Документація Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 22.04.2025).

40. Gourley D., Totty B. HTTP: The Definitive Guide. 2002. 635 с.

41. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 23.04.2025).

42. Документація Werkzeug. URL: https://werkzeug.palletsprojects.com/en/stable/utils/ (дата звернення: 23.04.2025).

ДОДАТКИ

Додаток А

```
function renderFEN(fen) {
    document.getElementById('boardWrapper').style.display = 'flex';
    const board = document.getElementById('lessonBoard');
    board.innerHTML = '';
    const rows = fen.split(' ')[0].split('/');
    const files = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'];

    for (let r = 0; r < 8; r++) {
        let file = 0;
        for (let c of rows[r]) {
            if (!isNaN(c)) {
                for (let i = 0; i < parseInt(c); i++) {
                    const square = document.createElement('div');
                    square.className = 'square ' + ((r + file) % 2 === 0 ? 'light' : 'dark');
                    const coord = files[file] + (8 - r);
                    square.dataset.square = coord;
                    square.addEventListener('dragover', e => e.preventDefault());
                    square.addEventListener('drop', handleDrop);
                    board.appendChild(square);
                    file++;
                }
            } else {
                const square = document.createElement('div');
                square.className = 'square ' + ((r + file) % 2 === 0 ? 'light' : 'dark');
                const coord = files[file] + (8 - r);
                square.dataset.square = coord;
                square.addEventListener('dragover', e => e.preventDefault());
                square.addEventListener('drop', handleDrop);

                const color = c === c.toUpperCase() ? 'w' : 'b';
                const type = c.toUpperCase();
                const piece = color + type;

                const img = document.createElement('img');
                img.src = `/static/images/${pieces[piece]}`;
                img.className = 'piece';
                img.setAttribute('draggable', 'true');
                img.setAttribute('alt', type);
                img.dataset.square = coord;
                img.addEventListener('dragstart', handleDragStart);
                square.appendChild(img);
                board.appendChild(square);
                file++;
            }
        }
    }
}
```

ДОДАТОК Б

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____
_____
Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»; що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації; що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів; що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)