Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

РОЗРОБКА ОНЛАЙН-СЕРВІСУ ДЛЯ РОЗВИТКУ ПАМ'ЯТІ

Текстова частина до курсової роботи
за спеціальністю "Комп'ютерні науки та інформаційні технології" 122

Керівник курсової роботи
Канд.фіз.-матем. наук
Гречко А.В.
(прізвище та ініціали)

_____
(підпис)
"____" _____ 2021 р.

Виконала студентка 3 курсу
Болюх І.О.
(прізвище та ініціали)
"____" _____ 2021 р.

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій

ЗАТВЕРДЖУЮ
Зав. кафедри
Малашонок Геннадій Іванович
_____
(підпис)
“____” _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

Студентки Болюх Ірини Олександрівни факультету інформатики 3 курсу
ТЕМА: Розробка онлайн-сервісу для розвитку пам’яті

Вихідні дані: Сервіс містить онлайн-ігри, що сприяють покращенню пам’яті та відповідають індивідуальному прогресові користувача, надає статистику розвитку та порівняння результатів з іншими користувачами.
Зміст текстової частини до курсової роботи:
Календарний план
Вступ
Частина 1: Аналіз предметної області. Постановка завдання курсової роботи
Частина 2: Теоретичні відомості про розробку веб-застосувань та пам’ять
Частина 3: Опис реалізації програмного продукту
Висновки
Список використаної літератури
Додатки
Дата видачі “____” _____ 2020 р.
Керівник_____Завдання отримала _____
(підпис) (підпис)

Тема: Розробка онлайн-сервісу для розвитку пам’яті

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	15.10.2020	
2.	Огляд технічної літератури за темою роботи.	16.11.2020	
3.	Дослідження актуальності проблеми.	20.12.2020	
4.	Вивчення аналогів.	25.12.2020	
5.	Планування функціоналу сайту.	29.12.2020	
6.	Створення проекту та основних сторінок.	26.03.2021	
7.	Реалізація реєстрації та авторизації користувачів, їх збереження в базі даних.	31.03.2021	
8.	Додавання ігор, нарахування балів, початкового тестування рівня розвитку.	27.04.2021	
9.	Відображення статистики, фільтрація ігор відповідно до рівня.	01.05.2021	
10.	Додавання стилів та секції рекомендованих технік покращення пам’яті.	03.05.2021	
11.	Написання текстової частини.	03.05.2021	
12.	Перегляд роботи керівником.	10.05.2021	
13.	Створення презентації.	13.05.2021	
14.	Здача курсової роботи.	17.05.2021	
15.	Захист роботи.	24.05.2021	

Студентка Болюх І.О.

Керівник Гречко А.В.

“____” _____

Зміст

Перелік умовних позначень

1. Когнітивні функції мозку - це здатність розуміти, пізнавати, вивчати, усвідомлювати, сприймати і переробляти зовнішню інформацію [1].
2. Нейронна пластичність – здатність мозку відновлюватись та реструктуризуватися [2].
3. СКБД – система керування базами даних.
4. MVC (Model-view-controller) – архітектурний шаблон програмного забезпечення, що використовується для розробки користувацького інтерфейсу та поділяє логіку програми на три взаємопов’язані частини.
5. LPI (Lumosity Performance Index) – стандартизована шкала сервісу Lumosity, що допомагає порівнювати ігрові бали за пророблені завдання відповідно до того, на які когнітивні здібності вони орієнтовані.
6. Нейрогенез – здатність мозку генерувати нові нейрони шляхом поділу стовбурової клітини на стовбурову та новий нейрон.
7. Шаблонізатор – програма, що дозволяє вставляти дані з серверу в статичні файли та відображати їх як готовий HTML-документ клієнту.
8. JWT – JSON Web Token.

ВСТУП

На сьогоднішній день людина настільки звикла до того, що в Інтернеті можна знайти відповідь майже на будь-яке питання, що запам’ятовування певних речей уже не вважається за потрібне. Проблема розвитку та використання когнітивних можливостей мозку у даний час є доволі актуальною. Зважаючи на те, що пам’ять грає важливу роль у житті людини й впливає на всі сфери її життєдіяльності: покращення її функціонування є важливим завданням у сучасному світі. Також варто зазначити, що з населення планети старішає, а з віком збільшується ризик таких хвороб, як Альцгеймера, Паркінсона, склероз, тощо. Дослідження показали, що однією з передумов і засобів профілактики цих хвороб є повсякденне стимулювання мозку, когнітивні тренування [3]. Тому онлайн-сервіс, що надає можливість стимулювати когнітивні функції мозку та покращувати зв’язки між нейронами завдяки систематичному виконанню вправ, є досить цікавим, корисним та актуальним завданням.

Метою є створення веб-сервісу для розвитку пам’яті, який дозволить користувачам дізнаватися про різні техніки покращення процесу запам’ятовування та допоможе відновити роботу когнітивних функцій мозку.

Завданням є розробка веб-застосунку, який надасть можливість виконувати різні когнітивно-спрямовані ігри, слідкувати за своїм прогресом, обирати ігри відповідно до свого темпу розвитку та мотивувати користувачів розвиватися й порівнювати свої результати з іншими користувачами платформи.

Об’єктом дослідження є розробка онлайн-сервісу для розвитку пам’яті.

Предметом дослідження є аналіз роботи сайтів, які вже є на ринку та забезпечують сервісами, що сприяють розвитку пам’яті.

Веб-застосунок отриманий в результаті буде основою для створення більш складних систем, які передбачатимуть користувачів з різними ролями (контент-менеджер, спеціаліст-консультант і тд.) та більшого набору ігор.

Онлайн-сервіс розроблений на мові JavaScript на платформі Node.js з використанням фреймворку Express.js. Робота з базою даних відбувається за допомогою документо-орієнтованої СКБД MongoDB та з використанням її хмарної версії MongoDB Atlas. Процес написання коду відбувається у середовищі WebStorm та Visual Studio Code.

Текстова частина роботи містить наступне: вступ, три розділи, висновки, список використаних джерел та додатків.

У першому розділі описано аналіз предметної області, висвітлено актуальність проблеми дослідження, проаналізовано розробку схожих веб-застосунків, які є популярними на даний момент, виявлено наявні обмеження та сформульовано постановку задачі, що ґрунтується на проробленому аналізі.

У другому розділі надано теоретичні відомості про основні підходи та дослідження, що стосуються розвитку та роботи пам’яті. Також висвітлено теоретичну інформацію щодо розробки веб-застосувань з використанням архітектурного шаблону MVC, пояснено принцип клієнт-серверної архітектури та розподіл веб-розробки на фронтенд і бекенд.

Третій розділ описує практичну частину розробки проекту. Визначено сутності, що повинні зберігатися у базі даних, сформовано основний функціонал веб-застосунку, надано схематичний опис інтерфейсу користувача, описано, які можливості доступні для залогінених та анонімних юзерів. Продемонстровано алгоритм роботи сайту, що дозволяє зрозуміти головну задачу розробки даного застосунку. Обґрунтовано обрану мову та середовище програмування включно з використаними бібліотеками та фреймворками. Описано реалізовані методи та функції, а також надано зображення результуючого користувацького інтерфейсу. Надано звіт про результати тестування вихідного коду програми.

РОЗДІЛ 1

1.1 Аналіз сучасного стану питання та обґрунтування теми

У час високого розвитку технологій людство звикло до того, що більшість завдань можна доручити на виконання машинам. Інтернет у певній мірі замінює пам’ять, надаючи доступ до безмежної кількості інформації, що спричиняє погіршення когнітивних функцій мозку. Більше того, дослідження стверджують, що випромінювання мобільних телефонів згубно впливають на клітини мозку [4]. Люди лінуються й не надають достатньої уваги тому, що потрібно робити перерву у використанні сучасних гаджетів й намагатися приділити більше часу тому, щоб розвивати себе, використовувати можливості людського тіла.

Саме тому, було вирішено розробити онлайн-сервіс, який дозволить користувачам стимулювати функції головного мозку та дізнаватися корисні й цікаві речі про властивості пам’яті.

1.2 Огляд існуючих аналогів розробки

Марк Стибіч, кандидат наук у галузі здорового способу життя та автор статей онлайн-ресурсу Verywell Mind, сформував список рекомендованих платформ розвитку та покращення роботи пам’яті. До нього ввійшли такі застосунки, як Braingle, Lumosity та Happy Neuron [5].

Braingle надає величезну кількість головоломок, логічних пазлів та ребусів. Користувачі платформи мають можливість поповнювати її новими вправами, залишати коментарі до вже створених та обговорювати їхню складність. Також можна самому обирати, яку задачку розв’язати, використовуючи доступні фільтри (див. рис. 1.2.1). На сторінці перегляду головоломки є функція отримати відповідь, а деякі головоломки пропонують підказки (див. рис. 1.2.2).

З недоліків варто зазначити те, що немає можливості слідкування за своїм прогресом, неможливо відстежити скільки ребусів було успішно відгадано. По суті, користувач самостійно робить завдання й оцінює свої вміння. Варто було б додати певні обмеження у часі для розв'язку головоломки та поле для вводу власної відповіді, яку потім можна було б порівняти з правильною.

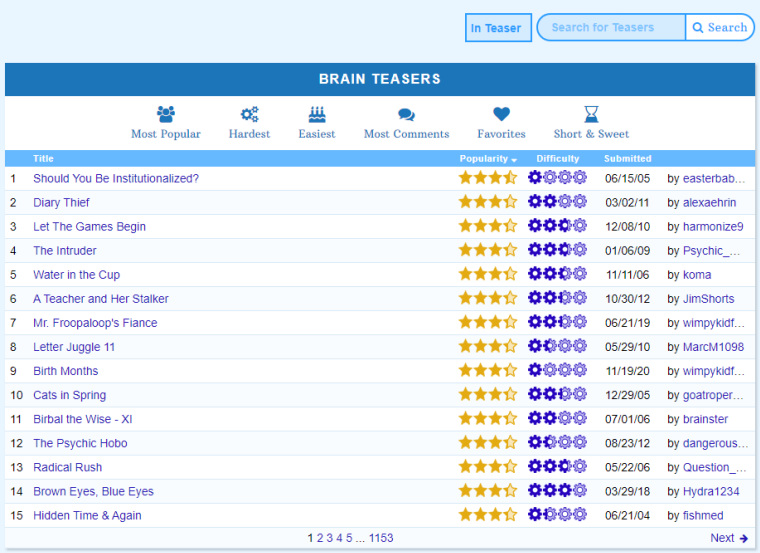


Рис. 1.2.1. Сторінка з доступними головоломками сайту Braingle

Рис. 1.2.2. Приклад головоломок сайту Braingle

Lumosity, на відміну від попереднього сервісу, відкриває для своїх юзерів можливість слідкування за своїм прогресом. Сайт розробив власну

систему оцінки здібностей користувача – LPI (див. рис. 1.2.3). Так клієнт може порівнювати свої успіхи у різних сферах когнітивних здібностей. Вважаю, що система нарахування балів за проходження ігор є хорошою мотивацією для користувачів, оскільки викликає бажання збільшувати свій Lumosity Performance Index та отримувати кращий результат, ніж був до того.

Недоліком цієї платформи було те, що разом з реєстрацією, користувача просять обрати, які сфери роботи мозку він хоче покращити. Від цього вибору залежить побудова усієї програми розвитку. За відгуком Метью Тертона, спеціаліста у сфері навчальної терапії, користувач не може знати напевно, що саме йому потрібно вдосконалити без попередньої консультації чи тестування [6]. Тому якщо налаштувати програму не в тих сферах розвитку, система не дасть очікуваного результату покращення пам'яті.

Згодом цей недолік було усунено, й тепер відразу після реєстрації сервіс пропонує зіграти у тестові ігри, які визначать розвиток головних функцій мозку (див. рис. 1.2.4). Проте це позбавляє користувача певної свободи вибору, адже програма буде пропонувати лише ті ігри, які вона вважатиме за потрібне, ґрунтуючись на результатах початкового тестування. Вважаю, що варто було б зробити відповідні ігри лише рекомендованими і надавати доступ до різних рівнів вправ.

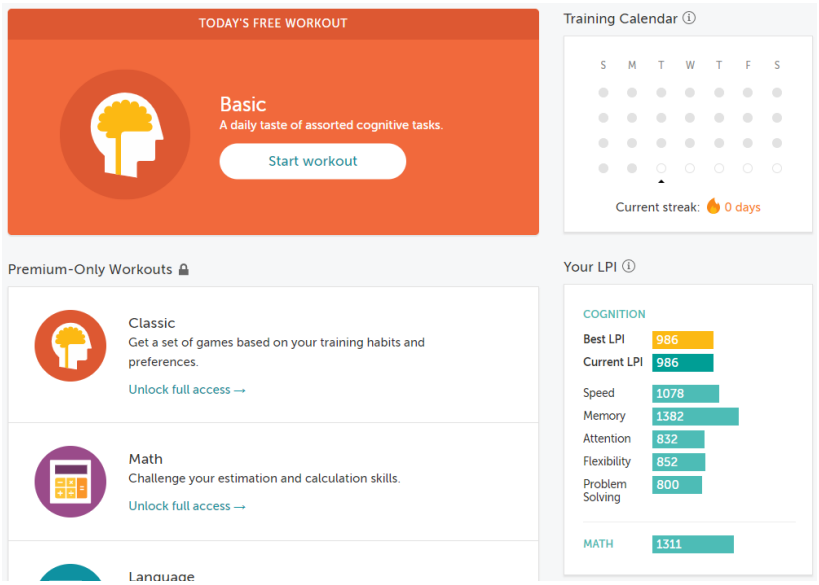


Рис. 1.2.3. Головна сторінка сервісу Lumosity, що відображає прогрес юзера

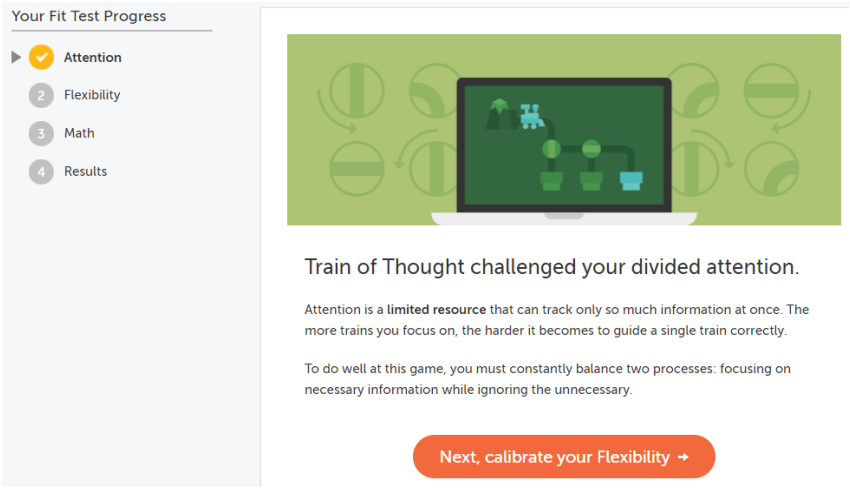


Рис. 1.2.4. Початкове тестування сервісу Lumosity

Веб-сервіс **Happy Neuron** дає можливість самому обирати гру, в яку бажаєш зіграти. Також платформа має рубрику новин, що дозволяє дізнаватись цікаві факти про роботу людського мозку та відкрити для себе більше інформації про когнітивні функції та процеси, що відбуваються у людській голові (див. рис. 1.2.6). Сервіс забезпечує користувача звітом про його прогрес. Спочатку складно розібратися з призначенням усіх кнопок та пунктів меню, що знаходяться на сайті (див. рис. 1.2.5). Функціонал виглядає нагромадженим, хоча по суті не несе особливо важливої інформації. Інтерфейс сайту можна було б спростити для кращого користувацького інтерфейсу. Ще одним недоліком сервісу є те, що немає можливості порівняти свій прогрес з іншими користувачами платформи.

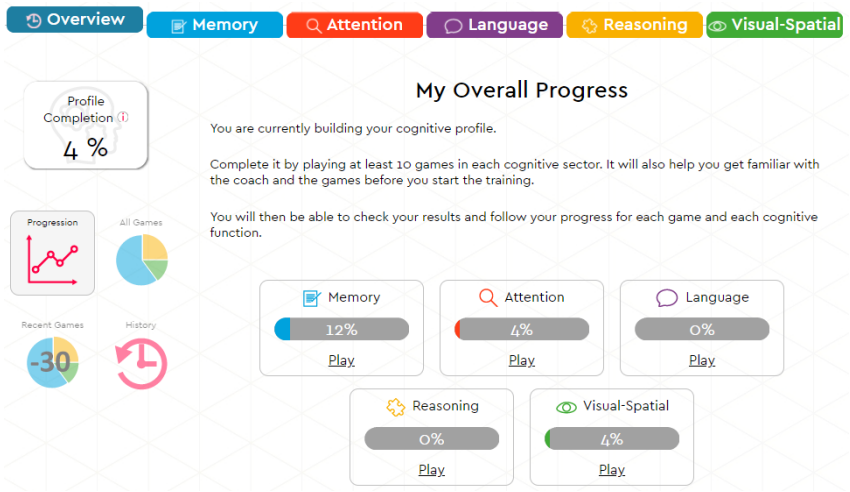


Рис. 1.2.5. Головна сторінка прогресу юзера сервісу Happy Neuron

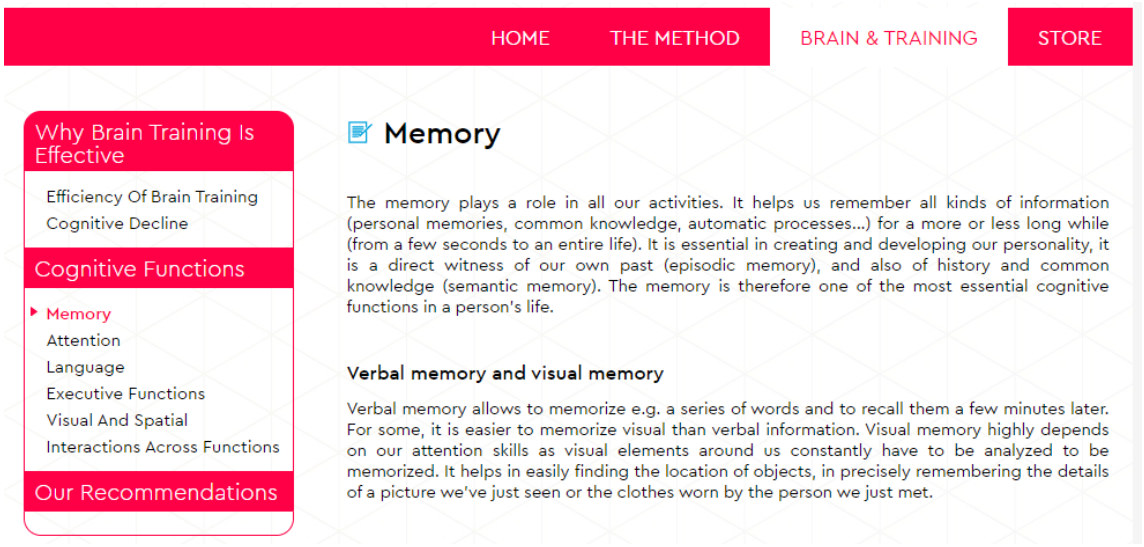


Рис. 1.2.6. Секція новин сервісу Happy Neuron

Отже, з аналізу вищезгаданих веб-застосунків, можна зробити висновок, що варто запропонувати наступне задля усунення певних недоліків:

1. Ігри мають перевірятися та оцінюватися програмою.
2. Користувач повинен мати можливість слідкувати за своїм прогресом.
3. За проходження ігор мають нараховуватися заохочувальні бали.
4. Рейтинг усіх користувачів задля мотивації.
5. Свобода вибору гри водночас з пропонованими завданням відповідно до рівня розвитку користувача.

І одним з найголовніших пунктів є те, що сервіс має забезпечувати не лише іграми, а й цікавою та корисною інформацією, що стосується пам’яті, когнітивних властивостей. Користувачі мають розуміти процеси, що відбуваються у їхньому тілі та вивчати себе задля кращої мотивації розвиватися та вдосконалювати можливості мозку.

1.3 Постановка завдання

У рамках даної курсової роботи були поставлені наступні задачі:

1. Проаналізувати бізнес-логіку веб-застосунків, що вже існують, дослідити їх функціональні можливості, засоби представлення інформації, користувацький інтерфейс.
2. Дослідити предметну область, вивчити матеріали та роботи, що стосуються розвитку та покращення функцій пам’яті.
3. Дослідити підходи до розробки сучасних веб-сайтів. Вибрати актуальні технології для створення онлайн-сервісу.
4. Розробити онлайн-сервіс розвитку пам’яті, що надає користувачам можливість:
   - реєстрації та автентифікації на сайті;
   - визначення початкового рівня зареєстрованого користувача, згідно з яким система рекомендувати виконувати ігри в певному порядку, а також надаватиме рекомендації, щодо налаштування складності гри;
   - вибір гри та нарахування балів за її проходження (кількість балів варіюється відповідно до складності гри);
   - отримання статистики прогресу користувача;
   - формування рейтингу найдосвідченіших користувачів;
   - перегляд щотижневої секції новин, цікавих фактів та порад щодо розвитку та роботи когнітивної системи людини;
   - оновлення, додавання та видалення ігор (для розробника);
   - оновлення щотижневого допису (для контент-менеджера).

РОЗДІЛ 2

2.1 Основи розробки веб-застосувань

Розробка веб-застосунку складається з двох частин: фронтенду та бекенду.

Фронтенд відповідає за макет сторінок, їхні стилі, інтерактивність, взаємодію з клієнтом. Тобто це те, що бачить користувач. Для реалізації фронт енду використовують HTML, CSS та JavaScript.

Бекенд складається ж з: серверу, програми, що відповідає за зв’язок з базою даних та бізнес-логіку сайту, та бази даних, що зберігає в собі потрібні для функціонування сайту дані. Усе це є серверною стороною веб-сайту. Для написання бек енду використовують такі мови, як: Java, JavaScript, Python, PHP, Ruby.

Розробка веб-застосунку ґрунтується на клієнт-серверній архітектурі, яка працює наступним чином (див. рис. 2.1.1). Клієнт надсилає запит на сервер за допомогою Інтернету, використовуючи браузер. У ролі клієнта може бути як комп’ютер, так і телефон, або будь-який інший гаджет. Сервер отримує запит від юзера, шукає (якщо потрібно, то звертається до бази даних) та формує відповідь на нього. Знайдений результат відправляє клієнту.

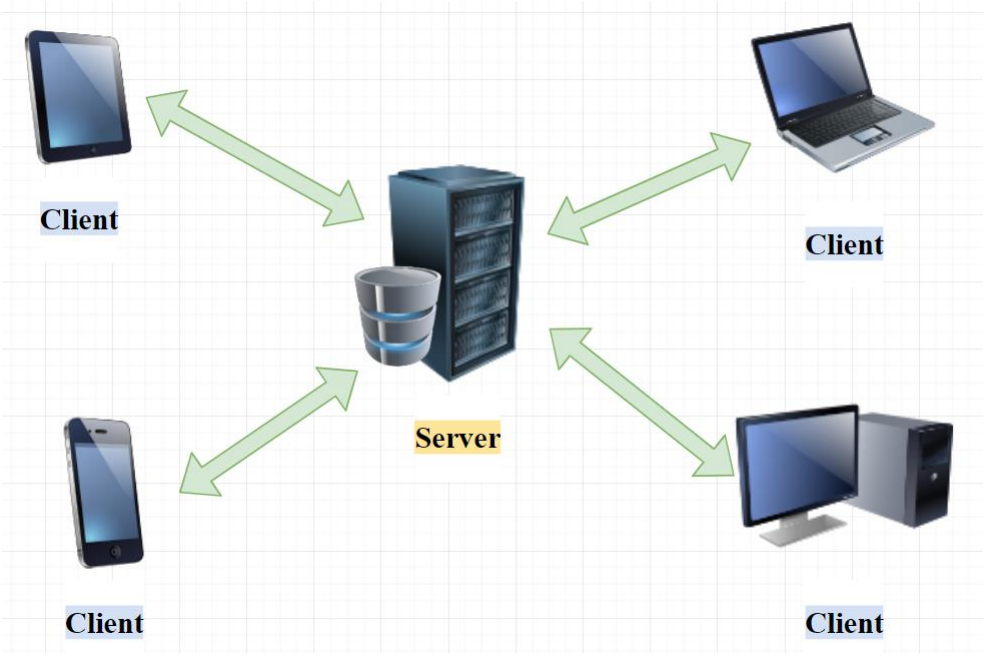


Рис. 2.1.1. Схематичне зображення клієнт-серверної архітектури

MVC підхід до організації коду під час проектування веб-розробки є досить популярним на сьогоднішній день. Ідея полягає в тому, що кожна частина коду має свою мету, яка відрізняється від інших [7]. Відокремивши логіку певних частин коду, процес проектування самої програми стає набагато легшим.

Зображення такого шаблону розробки програмного забезпечення наведено нижче (див. рис. 2.1.2). Model (Модель) відповідає зазвичай за об’єкти, що існують у реальному житті. Вона дозволяє налагодити зв’язок з базою даних та утримує дані, що є необхідними для функціонування застосунку. View (Вид) – код, що відповідає за представлення інформації, яку надає застосунок користувачу. Вид відповідає за те, як клієнт взаємодіє з програмою та як її бачить. Controller (Контролер) налагоджує зв’язок між Моделлю та Видом. Отримавши запит від користувача, він приймає рішення, що робити з ним далі. Наприклад, звертається до Моделі, яка в свою чергу звертається до бази даних та витягує звідти певну інформацію, що була запрошена юзером. Далі Модель віддає цю інформацію Контролеру, який передає отримані дані на View. Вид же відобразить відповідь на запит користувача у браузері, де той зможе його переглянути.

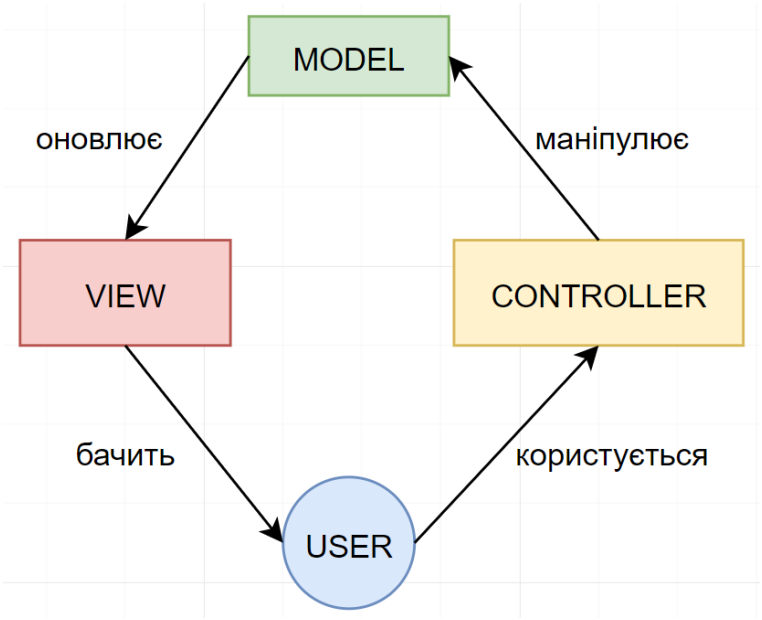


Рис. 2.1.2. Схема шаблону проектування MVC

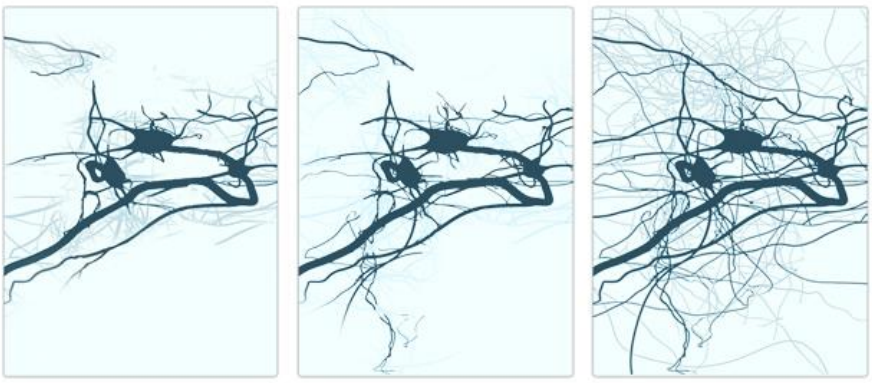
2.2 Властивості пам'яті та способи її покращити

Мозок людини має таку особливу здатність, як нейронна пластичність. Ця особливість полягає в тому, що нервова система може змінюватись у свої структурі під впливом різних чинників, що приходять з навколишнього середовища. Ці зміни можуть бути як позитивними, так і негативними. Саме ця здатність людського організму дозволяє людям відновити роботу мозку після тяжких травм та допомагає зменшити наслідки таких хвороб як Паркінсона, Альцгеймера, дислексія, розсіяний склероз і т.д. [2].

Впродовж усього життя людство вчиться та дізнається нові речі, поповнюючи свою базу знань новими поняттями та асоціаціями. З кожним поповненням мозок встановлює нові зв'язки між нейронами. Варто зазначити, що нейрони поєднані між собою синапсом. З кожним новим засвоєнням інформації або ж практикою й використанням уже наявних знань, цей синапс, зв'язок між нейронами, посилюється. У розглянутій статті про нейронну пластичність [2] наведено зрозумілий приклад: коли людина намагається розпізнати пташку, між нейронами відбувається новий зв'язок. Нейрони, що відповідають за зір - фіксують колір пташки, ті що за слух – звучання її співу, інші – назву. Таким чином, задіяні нейрони утворюють ланцюжок зв'язку, і з кожною такою спробою розпізнати пташку, електричний сигнал проходить по цьому ланцюжку більш швидко та ефективно, тобто зв'язок стає тіснішим та кращим. На цьому прикладі можна помітити, що практика позитивно впливає на ефективність роботи мозку, адже використовується властивість нейронної пластичності для покращення зв'язків між нейронами (див. рис. 2.2.1).

Дослідження та опитування показали, що не зважаючи на вікові особливості та пережиті травми, люди, що займаються розумовою діяльністю мають кращу нейронну пластичність [2].

Внаслідок стабільного виконання вправ, можна налагодити нейронні зв'язки та покращити як робочу пам'ять, так і інші когнітивні здібності мозку.



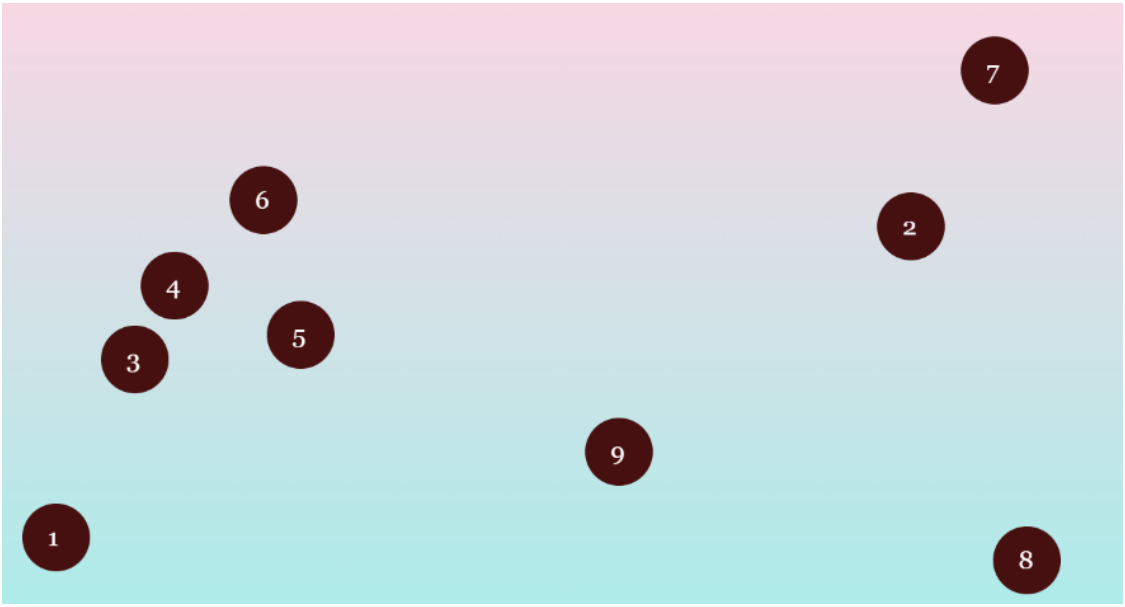
2.2.1 Вплив когнітивних тренувань на зв’язок нейронів

Робоча (оперативна або буферна) пам’ять – це процеси, що дозволяють нам зберігати інформацію, яка використовується короткочасно для виконання різного роду задач. Вона відповідає за розпізнання мови, математичних обрахунків, навчання, мислення, читання [8]. Людина використовує її, коли, для прикладу, почула на вулиці пісню, а потім намагається знайти її в Інтернеті й пригадує почуті слова, або ж коли отримує нові знання й намагається їх осмислити та співставити уже з раніше відомою нам інформацією. До речі, саме останній приклад відіграє важливу роль, оскільки саме цей процес дозволяє нам постійно навчатись, зберігати нову інформацію.

Для того, щоб перевірити об’єм робочої пам’яті використовують правило, що було сформульовано у 1956 році видатним американським психологом Джорджом Міллером – “Магічне число сім, плюс-мінус два”. Завдяки ряду проведених експериментів дослідник встановив, що мозок середньостатистичної людини здатен утримати в своїй оперативній пам’яті 5 односкладових слів, 7 букв алфавіту, 8 десяткових чисел та 9 двійкових чисел [9]. Звідси випливає, що це в середньому 7 плюс-мінус 2 інформаційні елементи.

Послідовний тест – це ще одне завдання, яке дозволяє оцінити здатність тимчасово утримувати дані в оперативній пам’яті та одночасно користуватися ними [10]. Його суть полягає в тому, щоб за певний проміжок

часу запам’ятати як можна більше чисел із серії та уникнути помилок при відтворенні цієї послідовності. Один із варіантів цього тесту: запам’ятати місце знаходження рандомно розкиданих кружечків, на яких написані цифри (див. рис. 2.2.2). За декілька секунд надписи на кружечках зникнуть, і завданням того, хто проходить тест, стає: натискати на кружечки у послідовності, що залежить від цифр, які були написані на них.



2.2.2 Тест на послідовність

Дослідження, проведені у сфері когнітивної психології людини, що були орієнтовані на вивчення процесів запам’ятовування, показали, що підвищена мотивація позитивно впливає на продуктивність пам’яті [11]. Також було виявлено, що за наявності кінцевої мети, необхідно мати проміжні цілі. За таких умов є більші шанси досягнути успіху. Саме тому задля достатньої мотивації було прийнято рішення передбачити у веб-застосунку рейтинг користувачів та різні рівні розвитку, що в певному розумінні виступають проміжними цілями для юзерів сервісу.

РОЗДІЛ 3

3.1 Аналіз технічного завдання

Веб-застосунок орієнтований на взаємодію з зареєстрованими користувачами. Юзер має можливість зареєструватися, пройти тест для визначення початкового рівня розвитку пам’яті, грати в розвиваючі ігри, проходити тести, виконувати завдання, накопичувати бали, спостерігати за своєю статистикою, порівнювати свої здобутки з іншими користувачами платформи. Анонімні користувачі мають лише доступ до топу користувачів сервісу та секції порад щодо покращення пам’яті. Також у системі будуть присутні користувачі з такими ролями, як контент-менеджер, що оновлюватиме секцію новин та розробник, що матиме можливість додавати нові ігри. Онлайн-сервіс передбачає інтерфейс контент-менеджера, розробника, зареєстрованого та анонімного користувачів.

Технічні вимоги користувачів до функціональних можливостей сервісу:

Анонімний користувач:

1. Перегляд головної сторінки сайту.
2. Доступ до перегляду топу юзерів.
3. Перегляд секції з порадами щодо покращення пам’яті, інформації щодо її властивостей.
4. Можливість зареєструватися.

Зареєстрований користувач (User):

1. Автентифікація.
2. Проходження тесту на визначення об’єму робочої пам’яті.
3. Вибір ігор відповідно до власного рівня розвитку.
4. Можливість обрати складність гри.
5. Пошук на сторінці ігор за рівнем складності та назвою гри.
6. Накопичувальна система балів за проходження ігор та виконання завдань.
7. Перегляд топу користувачів.

8. Перегляд власної статистики за день, місяць, рік від дня реєстрації.
9. Доступ до перегляду секції з новинками у сфері когнітивної психології, порад щодо покращення пам’яті та фактів щодо когнітивних функцій мозку.

Контент-менеджер (Manager):

1. Автентифікація.
2. Можливість редагувати тижневу секцію новин (змінювати, додавати або видаляти: картинку, текст, відео, заголовок новини).

Розробник (Developer):

1. Автентифікація.
2. Можливість додавати нові ігри.
3. Редагування уже існуючих ігор, зміна їх опису, рівня складності, картинки, посилання на гру.
4. Видалення ігор.

Специфікація вимог до даних:

Користувач

Зареєстрований користувач додається до бази даних. Про юзера зберігається його поштова скринька – логін, хешований за допомогою функції bcrypt пароль, нікнейм, що базується на введеній електронній пошті, його ід (ідентифікатор, що генерується при створенні користувача), початкова кількість балів, що ґрунтується на пройденому відразу після реєстрації тесті, рівень (статус) користувача та його роль (User, Manager, Developer).

Користувач з роллю User має змогу самостійно зареєструватися на сайті вказавши пошту та пароль, та пройшовши вступний тест. Користувачі з роллю Manager та Developer можуть бути додані лише розробником системи. Також юзери можуть заходити і виходити за допомогою інтерфейсу онлайн-сервісу.

Гра

Створюється користувачем з роллю Developer. Доступ до її перегляду мають усі зареєстровані користувачі сайту. Developer може змінювати її та видаляти. Гра містить такі поля: назва, опис, рівень (Усі рівні, Початківець, Середній, Досвідчений), посилання на картинку, посилання на сторінку з грою.

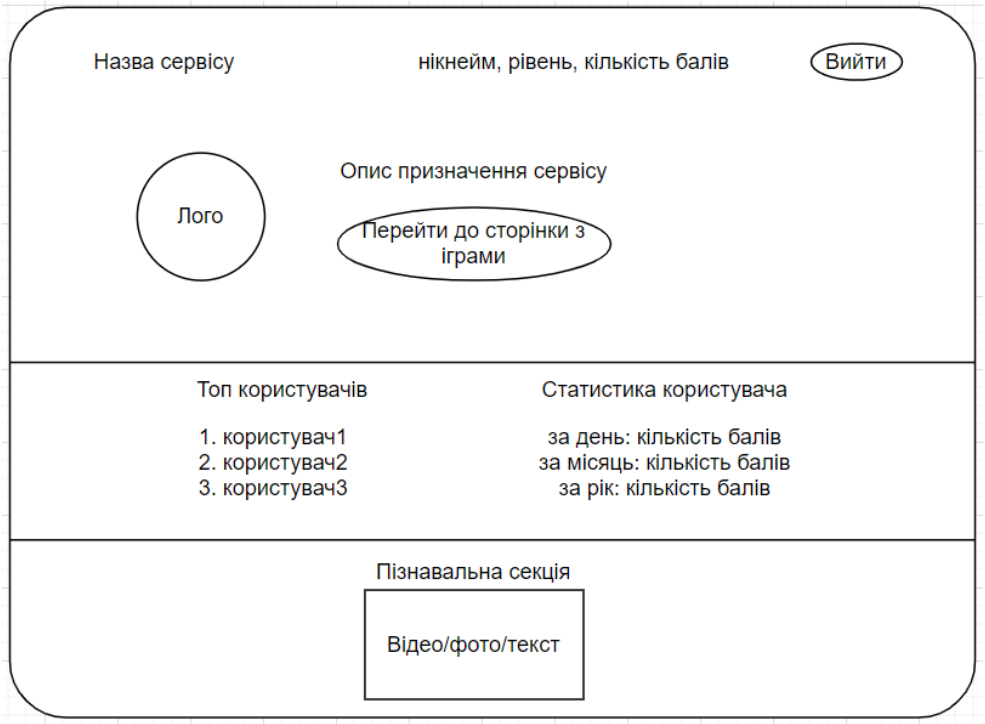
Допис

Тижнева секція новин додана розробником системи. Може оновлюватися лише користувачем з роллю Manager. Доступ до її перегляду мають усі користувачі системи. Допис містить поля: заголовок, опис, посилання на зображення, посилання на відео, дата оновлення.

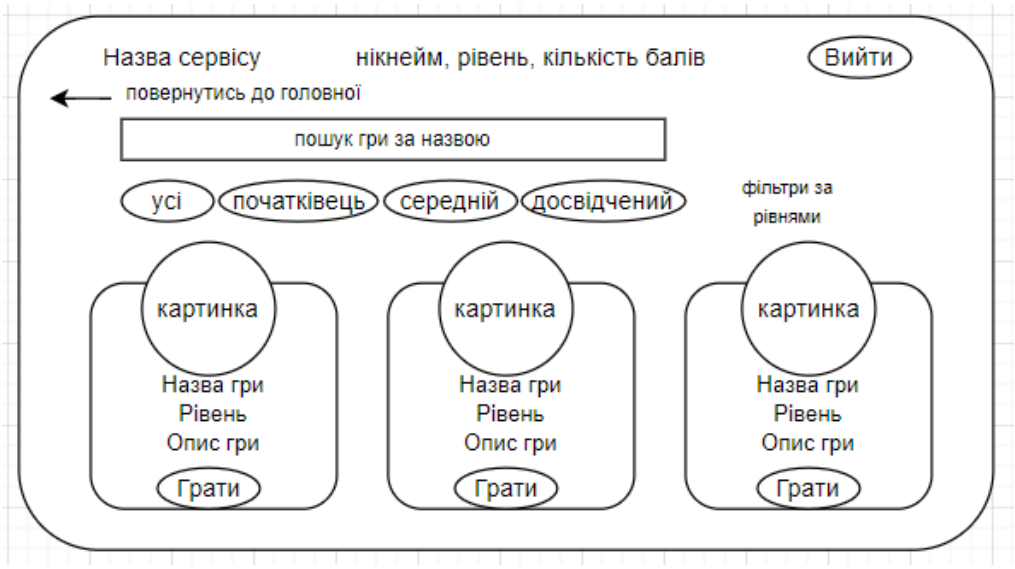
Інтерфейси сторінок веб-застосунку:

Перед розробкою сервісу та верстkою сторінок було створено схеми, що представляють функціонал сайту та розташування відповідних компонентів сторінки на екрані користувача. Вони стали підґрунтям для реалізації користувацького інтерфейсу та способу представлення інформації у браузері юзера.

Головна сторінка:



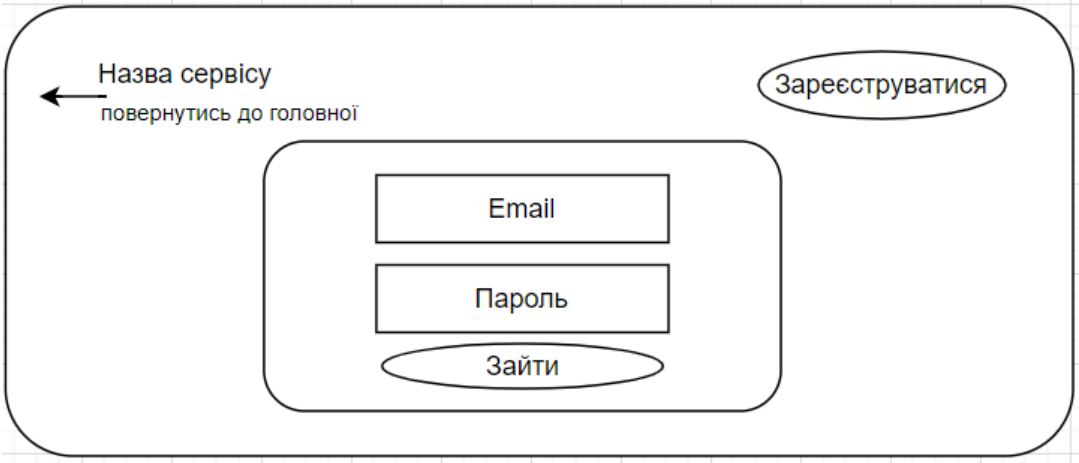
Сторінка з іграми:



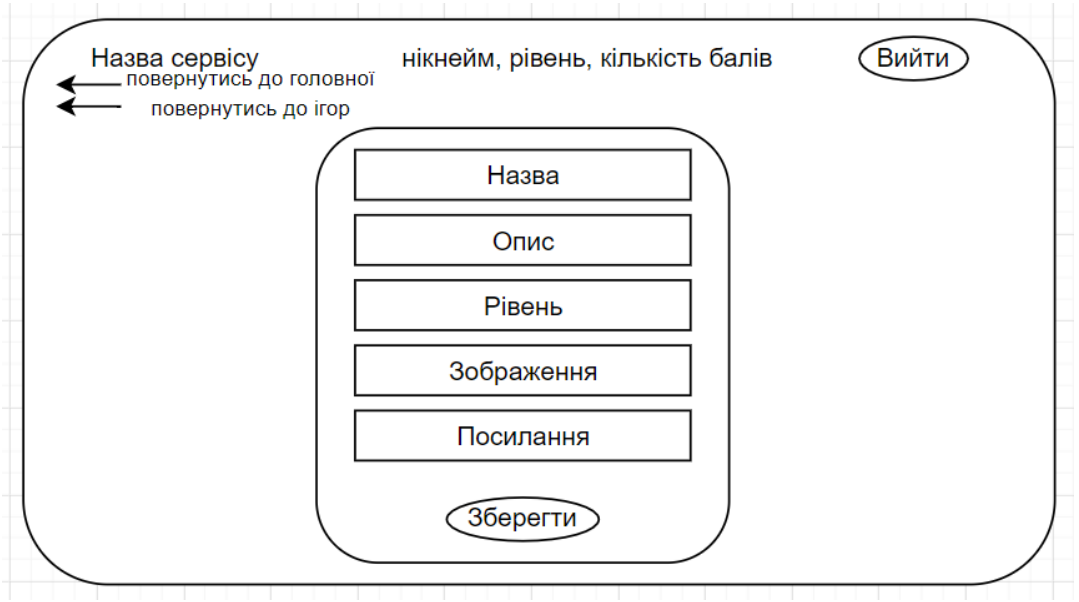
Сторінка конкретної гри:

Реєстрація:

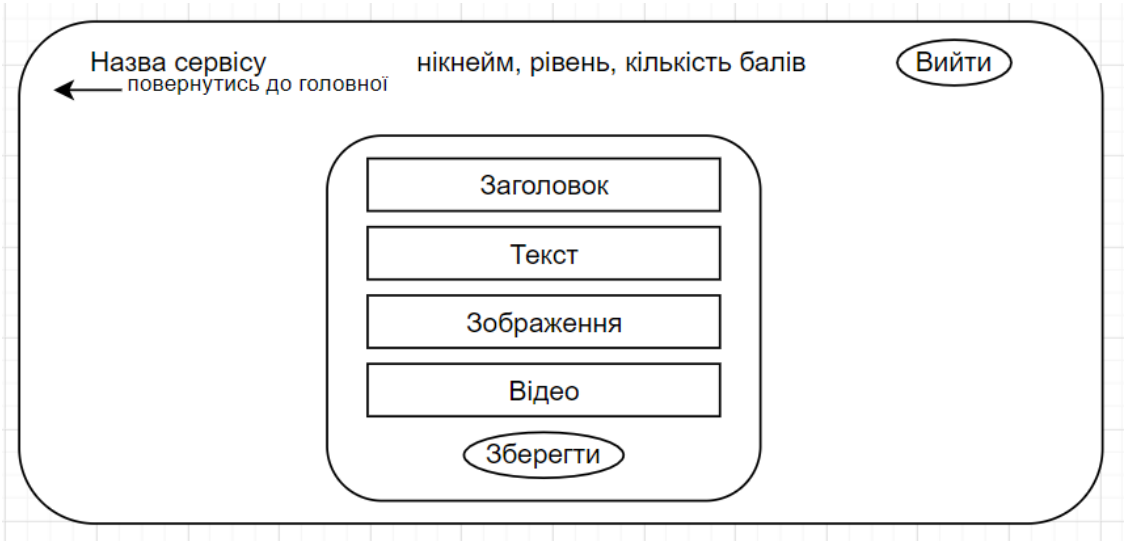
Вхід:



Сторінки редагування та додавання ігор:



Сторінка редагування допису:



3.2 Обґрунтування алгоритму й структури програми

Алгоритм роботи веб-застосунку представлений за допомогою блок-схеми (див. рис. 3.2.1).

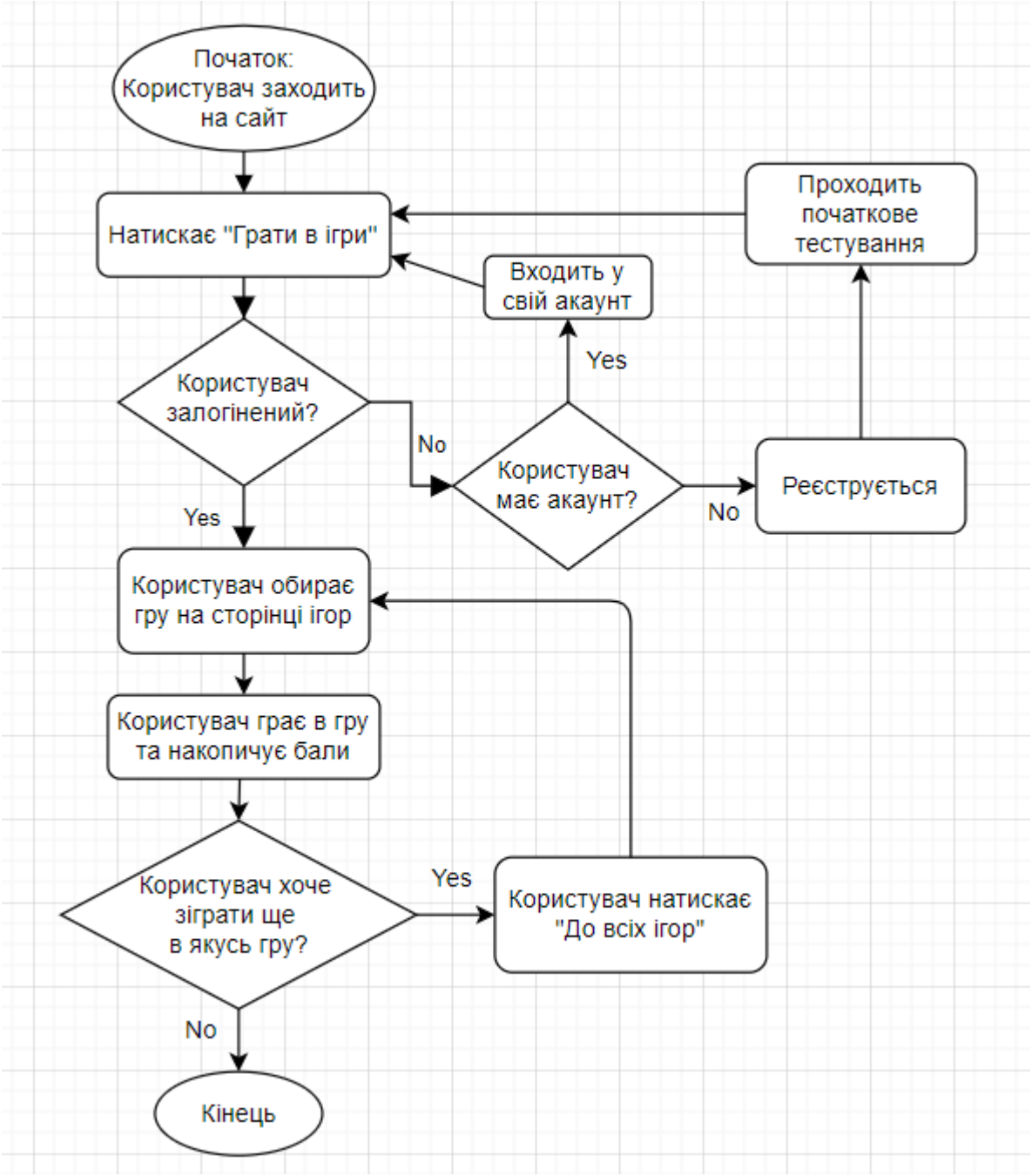


Рис. 3.2.1 Блок-схема роботи веб-застосунку

Основними модулями програми є реєстрація, проходження тестування, автентифікація, вибір гри, ігри та нарахування балів за них.

3.3 Обґрунтування вибору засобів розробки

Для розробки даного онлайн-сервісу було прийнято рішення використовувати JavaScript та реалізувати застосунок за допомогою платформи Node.js. Це міжплатформенне середовище виконання JavaScript написане на C++, що дозволяє писати серверний код для динамічних веб-сторінок та має ще певний ряд плюсів, які вплинули на зроблений вибір:

- Асинхронність

Дозволяє виконувати декілька процесів одночасно не блокуючи основний потік [12]. Наприклад, якщо потрібно звернутись до бази даних сервера і чекати поки буде отримано відповідь, у той же час можна обробити якісь інші запити, виконати інші задачі.

- Доступність

Безкоштовна платформа з відкритим доступом до вихідного коду.

- Простота та зрозумілість

Оскільки використовується синтаксис JavaScript, то для тих, хто мав досвід раніше з цієї мовою програмування, не буде складно користуватися даною платформою. Також в Інтернеті у відкритому доступі є достатньо покрокових інструкцій, в яких розібрано та пояснено основні підходи до розробки веб-застосунків.

- JavaScript Google V8

Рушій, який компілює JavaScript відразу в машинний код, що дозволяє виконуватися програмі набагато швидше й ефективніше.

- Насичена вбудована бібліотека

Стандартна бібліотека забезпечує великою кількістю вбудованих функцій, що є необхідними під час розробки веб-застосунку.

- NPM, зовнішні бібліотеки та готові модулі

NPM (Node Package Manager) – менеджер пакетів, що встановлюється разом з Node.js та дозволяє швидко й зручно підвантажувати необхідні

бібліотеки та модулі, яких на сьогоднішній день існує досить багато. Вони спрощують розробку, оскільки надають готові рішення до спільних проблем.

Для реалізації бекенд частини проекту було обрано фреймворк Express.js, що був написаний для Node.js. Він спрощує процес створення веб-аплікації, оскільки надає уже готові рішення, а також допомагає організувати структуру проекту за MVC-підходом. Express.js відповідає насамперед за маршрутизацію сайту, сесії, HTTP-запити та обробку помилок, тобто основні речі, які повинні відбуватися на сервері.

Переваги використання даного фреймворку:

- Легко встановити за допомогою команди npm та налаштувати. Розгорнута документація, що надає достатньо інформації для розуміння та вивчення [13]. Текстові інструкції та відео-пояснення доступні у широкому виборі.
- Досить просто налаштувати маршрутизацію сайту, що базується на назвах HTTP-запитів та URL-ів.
- Оскільки Node.js підтримує цілий ряд шаблонізаторів таких як Jade, EJS, Handlebars, Nunjucks і тд., Express.js дозволяє швидко налаштувати їх використання у застосунку.
- Легко налаштувати та визначити middleware. Для прикладу, створити middleware, що буде відповідати за обробку помилок або ж перевірку автентифікації користувача.
- Має метод, який дозволяє встановити зв’язок з такими базами даних, як MySQL, MongoDB, Redis.

Для збереження даних про зареєстрованих користувачів було обрано використовувати документо-орієнтовану систему керування базами даних MongoDB, оскільки вона зазвичай використовується при розробці застосунку з використанням Express.js та є досить потужною у своїх можливостях, адже зберігає файли JSON-подібному форматі, так званому BinaryJSON. Було

використано хмарну версію сервісу – MongoDB Atlas, яка є зрозумілою у своєму функціоналі та дозволяє виконувати маніпуляції з даними.

Задля зручної взаємодії з базою даних було використано ODM (Object Data Modeling) бібліотеку Mongoose, яка забезпечує таким функціоналом як – зв'язок між об'єктами програмного коду і представленням цих об'єктів у базі даних MongoDB, валідація схем об'єктів, що зберігаються у БД, пошук, оновлення та видалення об'єктів з БД. Також ця бібліотека забезпечує додатковим функціоналом, таким як, наприклад, створення віртуальних полів об'єкту, які можуть бути використаними у програмі, але не зберігатися безпосередньо у базі даних. Моделі реалізовані на основі Mongoose-схеми і є частиною MVC-підходу проектування.

Реалізація інтерфейсу відбувалася за допомогою JavaScript, CSS та HTML. Конкретно для написання розмітки веб-сторінки було використано EJS-шаблонізатор, який дозволив використовувати змінні та дані передані з серверу. Це полегшило перевірку на роль користувача та відображення відповідних елементів сторінки, а також використання часткових елементів зменшило обсяг написаного коду. Задля динамічності було використано AJAX-підхід, що допоміг реалізувати оновлення певних елементів сторінки. Також для стилізації сторінок було використано скриптову метамову SASS, а саме синтаксис – SCSS, який надає можливість використання змінних, міксинів, циклів, успадкування та принципу вкладеності, що теж в свою чергу спрощує написання та обсяг коду. Для процесу автентифікації та полегшення ідентифікування користувача було використано JWT, який зберігається в куках клієнтського браузера впродовж певного визначеного періоду та дозволяє користувачу виконувати запити, що потребують автентифікації.

Для середовища розробки було обрано WebStorm 2020.2.1, програмний продукт компанії JetBrains. Вбудований термінал цього IDE дозволяє у швидкому доступі виконувати необхідні команди, можливість налаштування

конфігурацій полегшує запуск проекту та потрібних скриптів. Також IDE надає доступ до встановлення необхідних плагінів, які допомагають у написанні коду забезпечуючи розробника підказками щодо синтаксису програми. Можливий пошук по проекту, швидке перейменування та розумне видалення, інтегрований функціонал контролю версій Git. Оскільки інтерфейс програми досить зрозумілий, WebStorm є зручним у користуванні. Крім того, завдяки підтримці JavaScript, HTML, CSS та шаблонізаторів одночасно – це IDE дозволяє писати як серверну, так і клієнтську частину програми. Саме ці переваги були основними при виборі середовища розробки.

3.4 Опис розробки програми

Побудова застосунку є реалізацією клієнт-серверної архітектури. Спілкування між клієнтом та сервером відбувається завдяки HTTP-протоколу. Клієнт надсилає запити типу GET, коли намагається отримати якісь дані з серверу та запити типу POST, коли відправляє певні дані на сервер.

Для структури проекту було використано MVC-підхід. Тобто програма має файли, що грають роль модель, контролерів та представлень і знаходяться у папках з відповідними назвами.

Для початку розглянемо папку **models**, у якій знаходяться усі файли, що відповідають за представлення екземплярів бази даних у об’єктному коді. У випадку цього веб-застосунку це – User.js, Post.js та Game.js. Саме ці три сутності зберігатимуться в MongoDB. Для прикладу розглянемо модель користувача, **User.js**. Для реалізації моделі користувача було використано Mongoose згаданий вище. Його функціонал полегшує користування базою даних. Mongoose допоміг налаштувати валідацію даних (див. рис. 3.4.1) та відкрив можливість використовувати віртуальні поля (див. рис. 3.4.2).

```
const userSchema = new mongoose.Schema( definition: {
  email: {
    type: String,
    required: [true, 'Будь ласка, введіть мейл.'],
    unique: true,
    lowercase: true,
    validate: [isEmail, 'Будь ласка, введіть коректний мейл.']
  },
  password: {
    type: String,
    required: [true, 'Будь ласка, введіть пароль.'],
    minLength: [8, 'Пароль має складатися не менше, ніж з 8 знаків.']
  },
  role: {type: String...},
  score: {type: Number...},
  dayCreated: {type: Date...}
});
```

Рис. 3.4.1 Модель користувача

```
// method to get User name
userSchema.virtual( name: 'name').get(function() {
  return this.email.substring(0, this.email.lastIndexOf("@"));
});

// method to check if Manager
userSchema.virtual( name: 'isManager').get(function() {
  return (this.role === roles.Manager);
});
```

Рис. 3.4.2 Віртуальні поля моделі користувача

Для налаштування зв’язку з MongoDB було використано метод connect() у файлі, що відповідає за створення та налаштування серверу – **app.js** (див. рис. 3.4.3). Усі змінні середовища зберігаються у файлі **.env**, який доступний лише розробнику, оскільки має паролі та важливі змінні оточенні, що не мають бути у відкритому доступі.

```
const PORT = process.env.PORT || 3000;
// database connection
const dbURI = 'mongodb+srv://brain-training:' + process.env.MONGO_ATLAS_PW
            + '@braintraining.p4zyl.mongodb.net/' + process.env.MONGO_ATLAS_NM;
mongoose.connect(dbURI, options: { useNewUrlParser: true,
                                   useUnifiedTopology: true,
                                   useCreateIndex:true }) Promise<Mongoose>
        .then((result : Mongoose ) => app.listen(PORT)) Promise<Server>
        .catch((err) => console.log(err));
```

Рис. 3.4.3 Під’єднання до хмарного сервісу MongoDB

Для представлень було використано EJS-шаблонізатор (див. рис. 3.4.4). Файли, що відповідають за вид знаходяться у папці **views**.

```
app.set('view engine', 'ejs');
```

Рис. 3.4.4 Налаштування шаблонізатора застосунку

Даний шаблонізатор дозволяє скоротити написання коду, адже дозволяє вставляти уже готові частини коду за допомогою **includes**, використовувати цикли for та умовні оператори if/else. Прикладом використання можливостей шаблонізаторів є сторінка з іграми, код якої знаходить у файлі **games.ejs**. Передані з сервера дані використані для заповнення сторінки. Код перевіряє чи є користувач залогіненим, його рівень співставляє з рівнем гри, що наявна у масиві ігор, і якщо рівні співпали або якщо гра доступна для всіх рівнів, то виводить її на екран (див. рис. 3.4.5). Якщо ж роль користувача буде Developer, то йому буде виведено усі ігри масиву разом з потрібними функціями для редагування, видалення та додавання ігор.

```
<% if(games.length > 0) { %>
    <% for (let game of games) { %>
        <% if(user && (!developer) && (game.level === user.status || game.level === "Усі рівні")) { %>
            <li id="<%= game.title %>" class="games__game <%= game.level %>">
                <img src="/images<%= game.imageLink %>" alt="">
                <h4><%= game.title %></h4>
                <h5><%= game.level %></h5>
                <p><%= game.description %></p>
                <a href="<%= game.pageLink %>" class="btn btn-game">Грати</a>
            </li>
        <% } %>
```

Рис. 3.4.5 Код відображення ігор на сторінці games

У проекті наявна папка файлів, що відповідають за маршрутизацію – **routes**. Для прикладу **authRouts.js** містить усі шляхи, що стосуються автентифікації користувача (див. рис. 3.4.6).

```
const {Router} = require('express');
const authController = require('../controllers/authController');

const router = Router();

router.get( path: '/signup', authController.getSignup);
router.get( path: '/login', authController.getLogin);
router.post( path: '/signup', authController.postSignup);
router.post( path: '/login', authController.postLogin);
router.get( path: '/logout', authController.logout);

module.exports = router;
```

Рис. 3.4.6 Шляхи, що відповідають за автентифікацію

Деякі шляхи вимагають автентифікації користувача, тому в них наявне використання middleware, що відповідає за перевірку того, чи автентифікований користувач. Прикладом цього є шлях переходу до ігор, що знаходиться у файлі **gamesRoutes.js** (див. рис. 3.4.7).

```
const { requireAuth } = require('../middleware/authMiddleware');

router.get('/games', requireAuth, (req, res) => res.render( view: 'games'));
```

Рис. 3.4.7 Використання middleware для перевірки автентифікації

У цьому випадку використовується middleware – **requireAuth** (див. рис. 3.4.8).

```
const requireAuth = (req, res, next) => {
  const token = req.cookies.jwt;

  if (token){
    jwt.verify(token, process.env.JWT_SECRET, options: (err, decodedToken) => {
      if (err) {
        console.log(err.message);
        res.redirect('/login');
      }
      else {
        console.log(decodedToken);
        next();
      }
    });
  }
  else {
    res.redirect('/login');
  }
}
```

Рис. 3.4.8 Middleware, що перевіряє чи автентифікований користувач

У випадку, коли користувач не залогінений, його перекидує на сторінку логінування. Якщо ж він все-таки автентифікувався, то відбувається виклик next() і продовжується виконання наступних дій – а саме рендеринг сторінки з іграми – games.

Контролери, що відповідають за обробку запитів від клієнта містяться у папці **controllers**. Прикладом контролера буде файл **gamesController.js**. Для того щоб оновити кількість балів користувача після проходження певної гри на сервер надсилається POST-запит з отриманими за гру балами (див. рис. 3.4.9).

```
let xhr = new XMLHttpRequest();
xhr.open( method: "POST", url: "/update-score", async: true);
xhr.setRequestHeader( name: 'Content-Type', value: 'application/json');
xhr.send(JSON.stringify( value: {
    score: roundedScore
}));
```

Рис. 3.4.9 Код файлу **game-sequence.js**

Тоді ідентифікуємо користувача за допомогою JWT, що зберігається у куках та асинхронно оновлюємо поле користувача з балами у базі даних (див. рис. 3.4.10).

```
module.exports.scoreUpdate = (req, res, next) =>{
    const {score} = req.body;
    const token = req.cookies.jwt;

    if (token){
        jwt.verify(token, process.env.JWT_SECRET, options: async (err, decodedToken) => {
            if (err) {
                console.log(err.message);
                next();
            }
            else {
                console.log(decodedToken);
                let user = await User.findById(decodedToken.id);
                let newScore = user.score + score;
                await User.updateOne( filter: { _id: user.id}, update: { score: newScore });
                next();
            }
        });
    }
}
```

Рис. 3.4.10 Контролер, що відповідає за оновлення балів **gamesController.js**

Ще один з важливих контролерів - **statisticsController.js**, який відповідає за відображення статистики та ґрунтується на даних отриманих з БД можна знайти в додатках (див. Додаток А).

3.5 Створення об'єктів і розробка головної програми

Створення та оновлення об’єктів в базі даних відбувається за допомогою Mongoose. З реєстрацією на сайті користувач буде доданий до бази даних. Також відразу після реєстрації він проходить тест на визначення рівня обсягу пам’яті, після проходження якого, його кількість балів оновлюється та визначається рівень підготовленості користувача.

Для зберігання паролів користувачів та перевірки відповідності під час автентифікації було використано bcrypt. Її функції дозволяють забезпечити зберігання паролів у хешованому вигляді, а також збільшують стійкість до атак, оскільки використовують salt (сіль) (див. рис. 3.5.1).

```
//do this before new user was saved to db
userSchema.pre( method: 'save',  fn: async function(next){
  const salt = await bcrypt.genSalt();
  this.password = await bcrypt.hash(this.password, salt);
  next();
});

// method to login User
userSchema.statics.login = async function(email, password){
  const user = await this.findOne({email}); // find by email in db
  if (user){
    const auth = await bcrypt.compare(password, user.password);
    if (auth){
      return user;
    }
    throw Error('wrong password');
  }
  throw Error('wrong email');
}
```

Рис. 3.5.1 Методи моделі User, що відповідають за паролі

3.6 Опис файлів даних та інтерфейсу програми

На кожній сторінці сайту (окрім сторінки з тестом) відображається header та footer. **Header** містить назву сайту, що діє як лінк, та завжди може перенести користувача до головної сторінки **home.** Також **хедер** має кнопки залогінитись та зареєструватися, нікнейм залогіненого користувача, кількість набраних балів та рівень розвитку. У **footer** вказано рік створення, організацію та авторів ілюстрацій сайту.

З кліком на кнопку реєстрації, перед користувачем відкривається сторінка, що вимагає вводу електронної скриньки та паролю (див. рис. 3.6.1).

```
<form>
  <h2>Sign up</h2>
  <label for="email">Email</label>
  <input type="text" name="email" required/>
  <div class="email error"></div>
  <label for="password">Password</label>
  <input type="password" name="password" required/>
  <div class="password error"></div>
  <button>Sign up</button>
</form>
```

Рис. 3.6.1 Форма реєстрації користувача

```
form.addEventListener( type: 'submit', listener: async (e) => {
    e.preventDefault();
    emailError.textContent = '';
    passwordError.textContent = '';
    const email = form.email.value;
    const password = form.password.value;
    try{
        const res = await fetch( input: '/signup', init: {
            method: 'POST',
            body: JSON.stringify( value: {email, password}),
            headers: {'Content-Type': 'application/json'}
        });
        const data = await res.json();
        if (data.errors){
            emailError.textContent = data.errors.email;
            passwordError.textContent = data.errors.password;
        }
        //redirecting if successfully signed up
        if (data.user){
            location.assign( url: '/test');
        }
    }
    catch(err){
        console.log(err);
    }

});
```

Рис. 3.6.2 Відправка форми на сервер

Після заповнення форми, користувач відправляє її на сервер (див. рис. 3.6.2). У випадку неправильно введених даних сервер повертає помилки, що були виявленні при створенні нового акаунта, ці помилки будуть відображені в інтерфейсі користувача. Якщо ж усі дані були введені правильно, то користувача перенаправляє на сторінку з тестом. Ті ж дії відбуваються і з формою логінування. Помилки з неправильно введеним паролем або логіном будуть виведені на екран юзера. Помилки обробляються на сервері (див. рис. 3.6.3).

```
const errorChecking = (err) => {
  console.log(err.message, err.code);
  let errors = { email: '', password: ''};

  //incorrect email or password when logging
  if (err.message === 'wrong email'){
    errors.email = 'Incorrect email.';
  }
  if (err.message === 'wrong password'){
    errors.password = 'Incorrect password.';
  }

  //check for duplicate emails
  if (err.code === 11000){
    errors.email = 'This email is already registered.';
    return errors;
  }

  //validation errors
  if (err.message.includes( value: 'user validation failed')){
    Object.values(err.errors).forEach(({properties}) => {
      errors[properties.path] = properties.message;
    })
  }

  return errors;
}
```

Рис. 3.6.3 Перевірка даних форми у файлі authController.js

На головній сторінці home доступні: кнопка, що дозволяє перейти на сторінку з усіма іграми (може бути активована лише за умови автентифікації користувача), опис та призначення сайту, статистика користувачів та топу найбільш успішних юзерів сервісу, секція з новинками, техніками та

цікавими фактами, що стосується сфери когнітивних функцій мозку та пам’яті зокрема. Для користувача з роллю Manager також буде доступна функція редагування допису.

На сторінці з іграми відображаються ігри доступні на сервері. Ті ігри, що підходять для всіх рівнів розвитку або ж мають кастомізовані налаштування складності будуть відображатися для всіх. Ті ігри, що мають більший рівень складності, відображитумься відповідно для тих юзерів, які накопичили більшу кількість балів і мають більше досвіду практики. В коді це перевіряється за допомогою можливостей ejs. Для користувача з роллю Developer будуть відображатися додаткові можливості, а саме видалення, редагування та додавання ігор (див. рис. 3.6.4).

```
<% if(developer) { %>
    <li class="games__game <%= game.level %>" id="<%= game.title %>">
        <img src="/images<%= game.imageLink %>" alt="">
        <h4><%= game.title %></h4>
        <h5><%= game.level %></h5>
        <p><%= game.description %></p>
        <a href="<%= game.pageLink %>" class="btn btn-game">Грати</a>
        <a href="/update-game/<%= game._id %>" class="btn btn-game"><i class="fa fa-pencil" aria-hidden="true"></i></a>
        <form class="games__delete" action="/delete-game/<%= game._id %>" method="POST">
            <button type="submit" class="btn btn-game"><i class="fa fa-trash" aria-hidden="true"></i></button>
        </form>
    </li>
<% } %>
```

Рис. 3.6.4 Відображення додаткових ігор для користувача Developer

На сторінках ігор, що містять кастомізовані налаштування будуть висвічуватись рекомендовані параметри гри (див. рис. 3.6.5).

```
<label for='timeLimit'>Time limit(1-5)</label>
    <% if (user.status === 'Beginner') { %>
        <p>Recommended for you: 2</p>
    <% } else if (user.status === 'Average') { %>
        <p>Recommended for you: 1</p>
    <% } else { %>
        <p>Recommended for you: 2</p>
    <% } %>
<input type='number' id='timeLimit' name='timeLimit' min='1' max='5'>
```

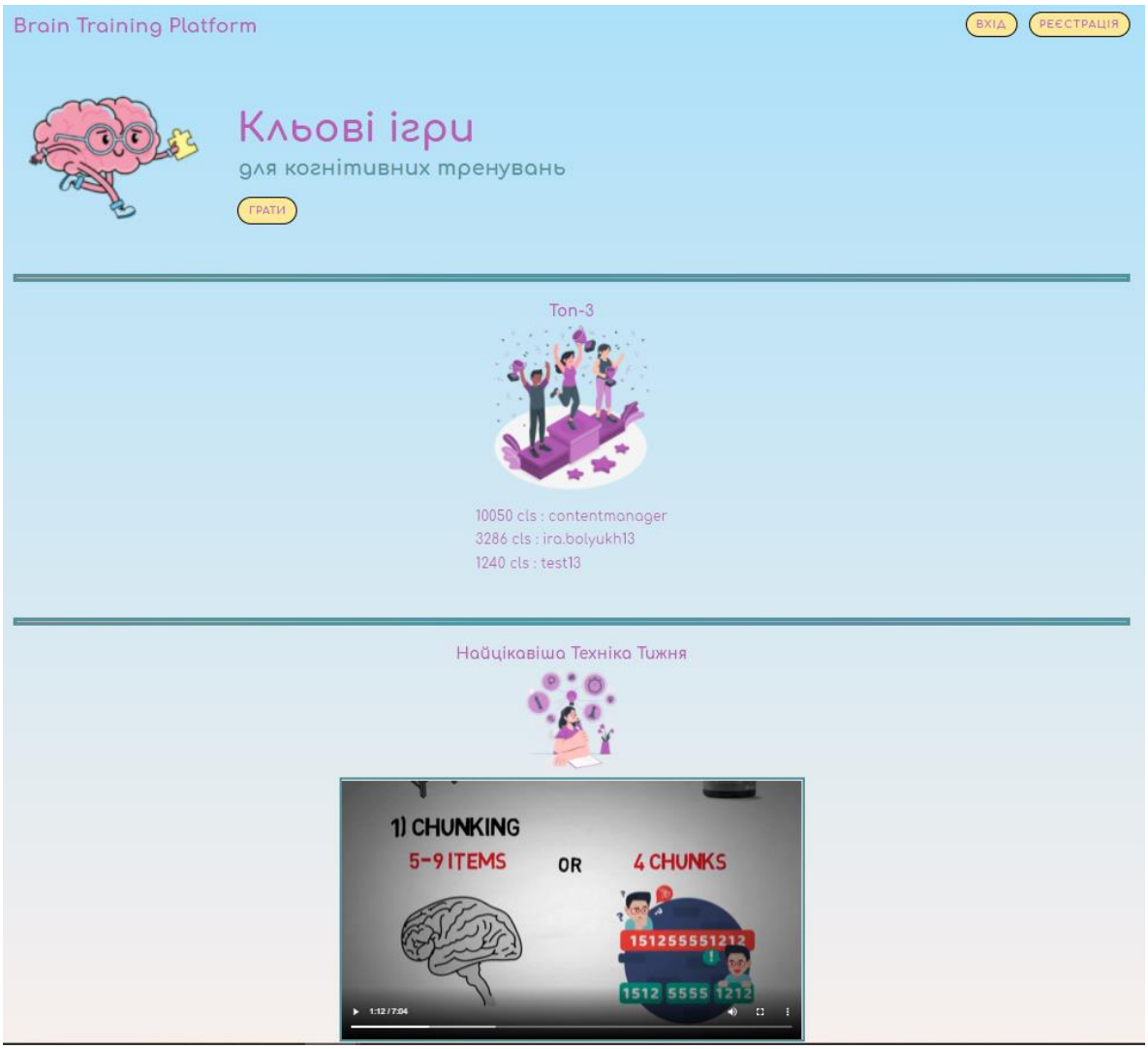
Рис. 3.6.5 Рекомендаційні параметри гри

Реалізація ігор відбувається у папці **games**. У ній представлений код інтерфейсу ігор, сам функціонал написаний на JavaScript та зберігається у папці **public/js**.

Після проходження гри, перед користувачем висвічується повідомлення, яке сповіщає про кількість набраних балів за гру, а також надсилається POST-запит для оновлення набраних балів користувача, що відображаються у верхній правій частині екрану (див. Додаток Б).

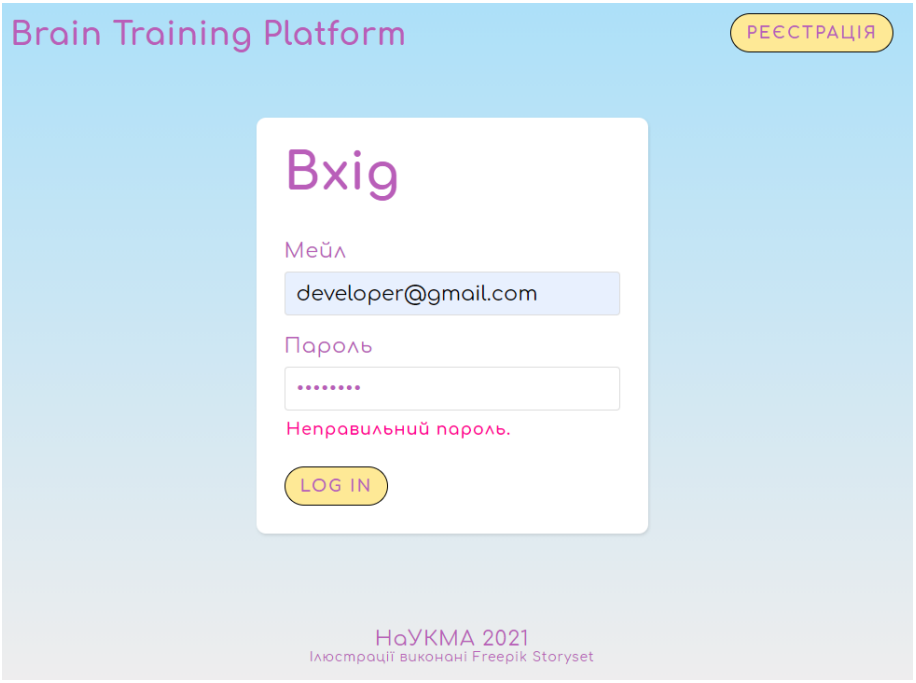
3.7 Інструкція користувача

Для початку перед користувачем відкривається головна сторінка сайту. До її перегляду мають доступ як автентифіковані так і анонімні юзери (див. рис. 3.7.1).



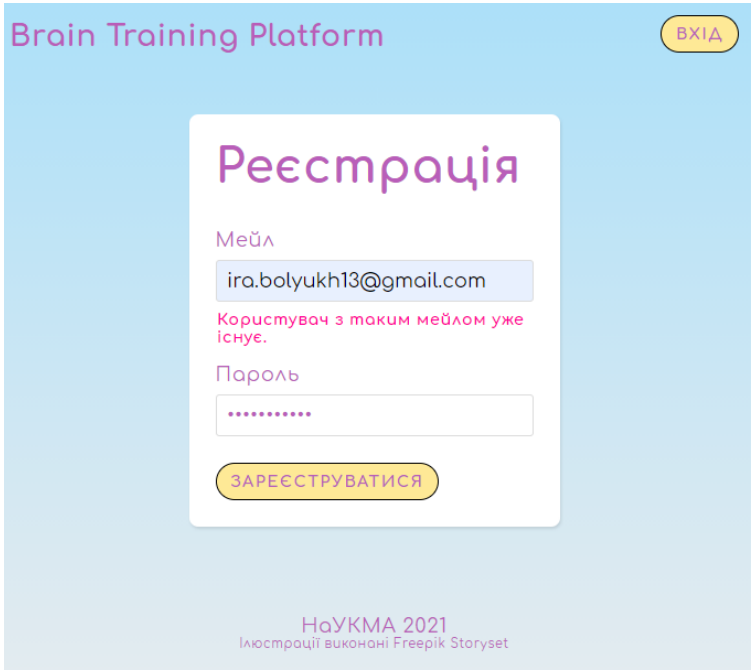
3.7.1 Вигляд головної сторінки сайту для анонімних юзерів

Якщо анонімний користувач спробує натиснути кнопку “Грати”, його буде перенаправлено на сторінку логінування (див. рис. 3.7.2). Відбуватиметься перевірка введених даних. Якщо пароль або мейл були введені неправильно, то користувача буде повідомлено про помилку.



3.7.2 Сторінка входу

Якщо користувач ще не створював акаунту, він зможе зареєструватися (див. рис. 3.7.2). Також відбуватиметься перевірка на правильність даних.



3.7.3 Сторінка реєстрації

Відразу після реєстрації користувача буде перенаправлено на сторінку з тестом. Цей тест є необхідним до проходження, оскільки він допоможе визначити можливості користувача, відповідно до яких здійснюватиметься рекомендація ігор. Інструкція по проходженню тесту наявна на сторінці самого тесту (див. рис. 3.7.4).

3.7.4 Сторінка з тестом

Після того, як користувач запам’ятав слова він намагається їх пригадати та записати у тому ж порядку в наступному кроці (див. рис. 3.7.5).

3.7.5 Сторінка з тестом

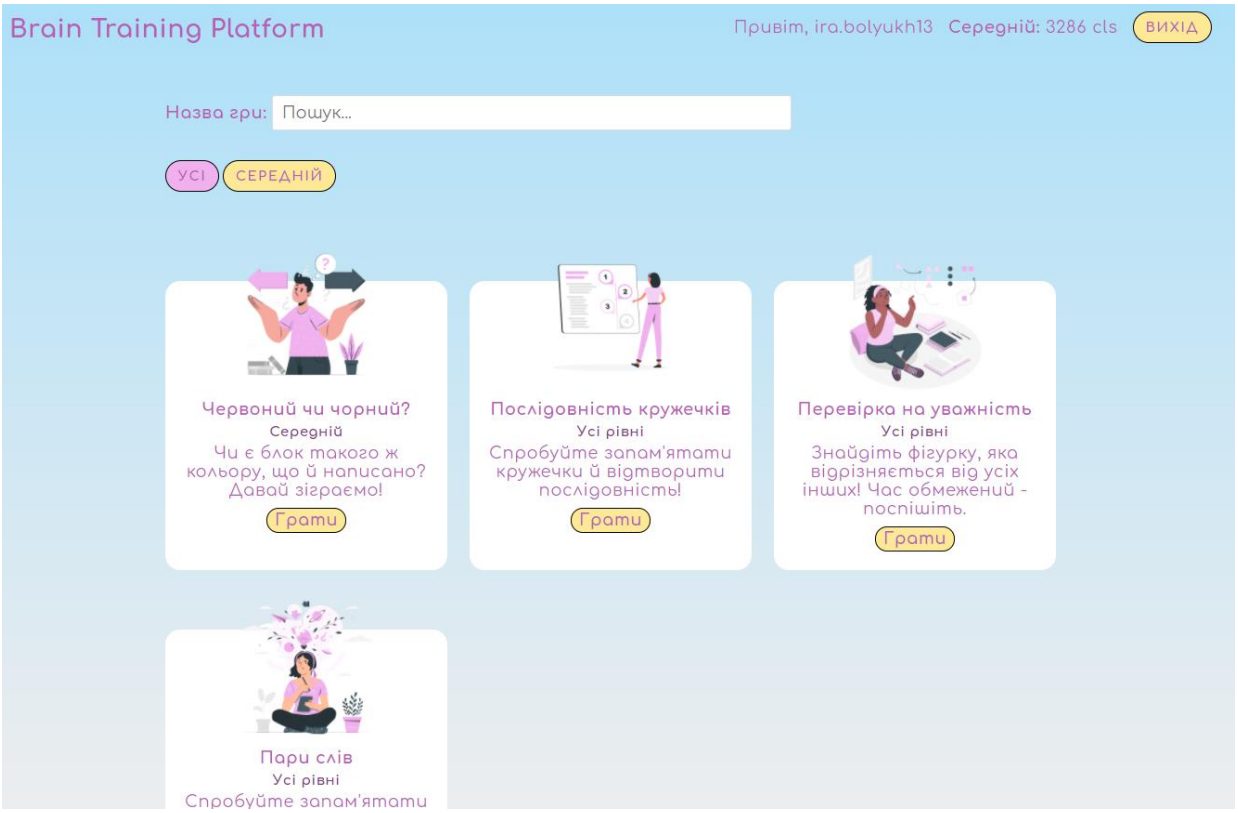
Далі користувач отримає результати тесту відповідно до яких, йому буде нараховано початкову кількість балів та визначено його рівень (якщо користувач запам’ятав менше 5 слів, то він буде “Початківець” з 10 балами, якщо від 5 до 9, то - “Середній” з 1000 балів, у випадку запам’ятовування більше 9 - “Досвідчений” з 10000 балів) (див. рис. 3.7.6). З натиском на кнопку “Почати”, користувач опиниться на головній сторінці уже як зареєстрований користувач.

Рис. 3.7.6 Результат тесту на обсяг пам’яті

Для автентифікованого користувача головна сторінка матиме дещо інший вигляд (див. рис. 3.7.7). Для нього відображатиметься його статистика, також у верхній правій частині екрану завжди відображатиметься його нікнейм, кількість набраних балів на поточний момент та рівень розвитку. Також на всіх сторінках буде доступна можливість вилогінитись з натиском кнопки “Вихід”. Якщо ж користувач схоче повернутися до головної сторінки, то достатньо буде натиснути на назву сайту – “Brain Training Platform”.

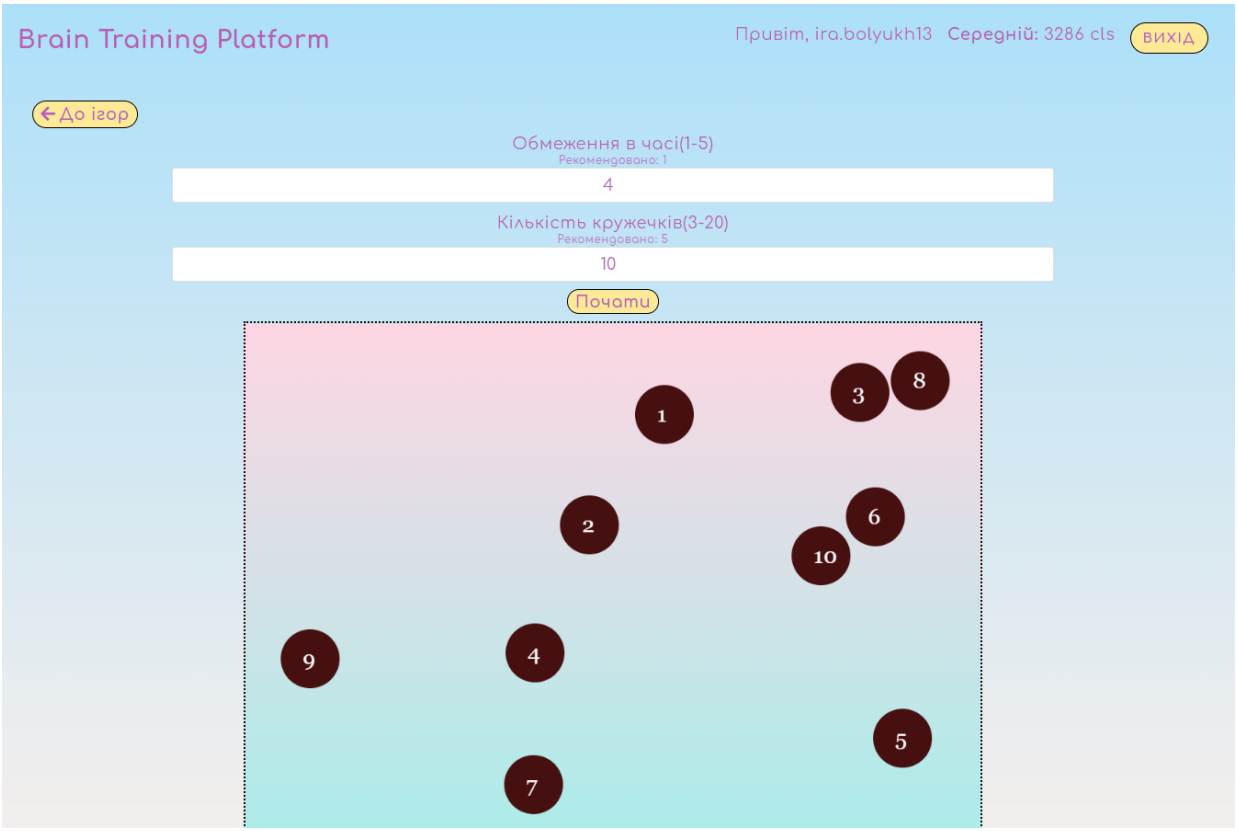
3.7.7 Головна сторінка автентифікованого юзера

Після натиску на кнопку “Грати”, за умови автентифікації, користувач переміститься до сторінки усіх ігор (див. рис. 3.7.8). На цій сторінці йому будуть доступні ігри, що відповідають його рівню та ігри, що підходять будь-якому рівню користувачів (фільтр з назвою “Усі”), оскільки мають кастомізовані параметри. Також на цій сторінці будуть доступні фільтри за рівнем, що спрощують пошук потрібної гри. А також можна здійснювати пошук гри за її назвою.



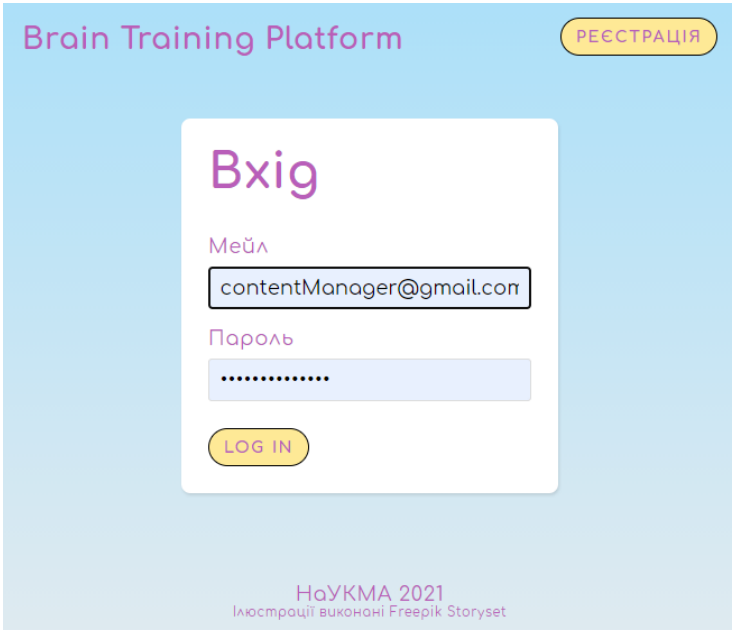
3.7.8 Сторінка з іграми

Після того як користувач натисне “Грати” на певній грі, його буде переміщено безпосередньо на сторінку з грою. Для прикладу гра на встановлення послідовності – “Послідовність кружечків”. Ця гра є доступною для всіх рівніх користувачів, оскільки вона передбачає кастомізовані параметри. Перед початком гри користувач самостійно визначає її параметри, а саме: скільки секунд він зможе бачити цифри на кружечках, що відповідають за порядок натискання відповідних кружечків, та число кружечків, які йому потрібно буде запам’ятати. Після встановлення параметрів гра почнеться з натисненням кнопки ‘Почати”. Сервіс передбачає рекомендації відповідно до поточного рівня користувача (див. рис. 3.7.9). Користувач може припинити гру та повернутися до вибору гри натиснувши на кнопку “До ігор”. Після проходження гри його кількість балів автоматично оновиться.



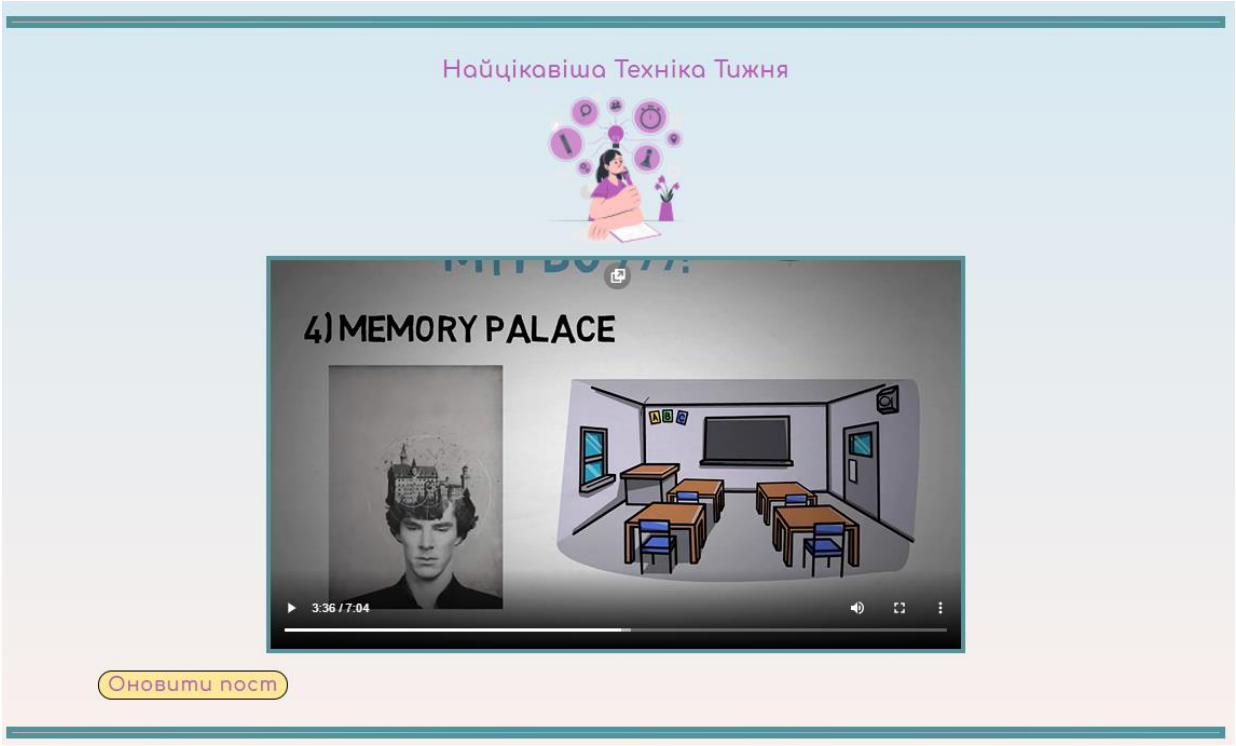
3.7.9 Гра “Послідовність кружечків”

Користувач з роллю Manager має відповідний мейл та пароль, що дозволяють йому ввійти в систему під своєю роллю (див. рис. 3.7.10).



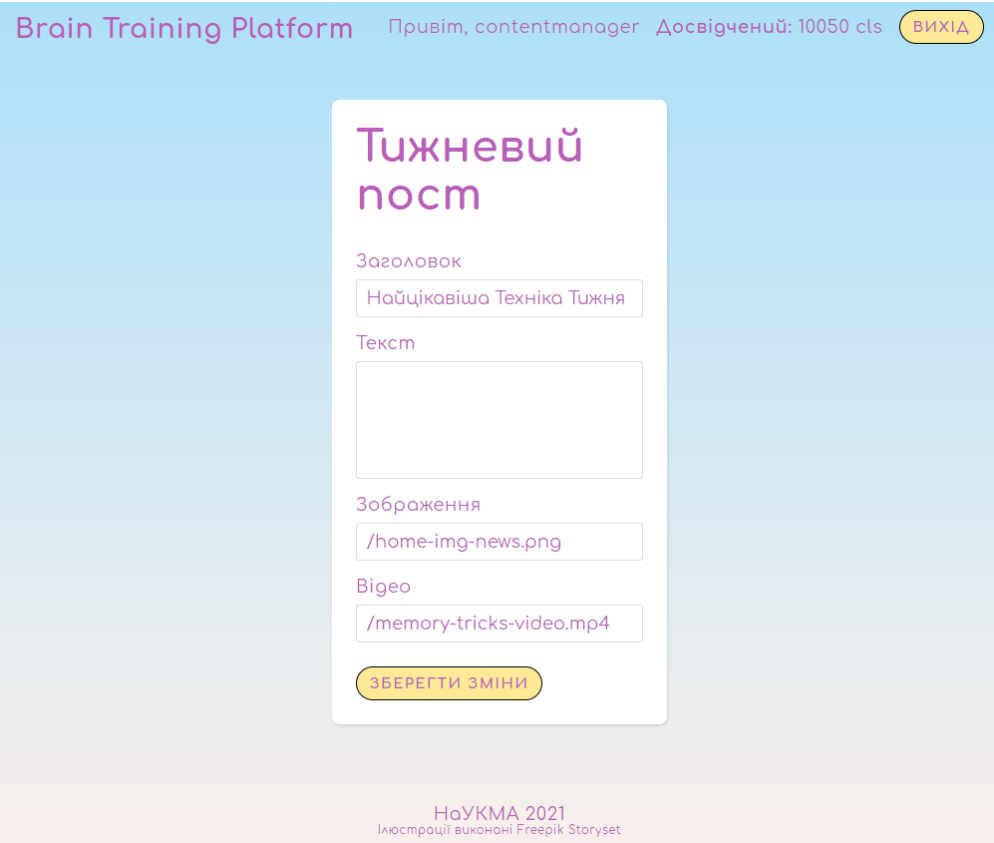
3.7.10 Вхід Manager

Контент-менеджер має усі ті права, що і звичайний залогінений користувач плюс право на редагування тижневої секції новинок на головній сторінці сайту (див. рис. 3.7.11). Ця функція є доступною лише користувачу з роллю Manager.



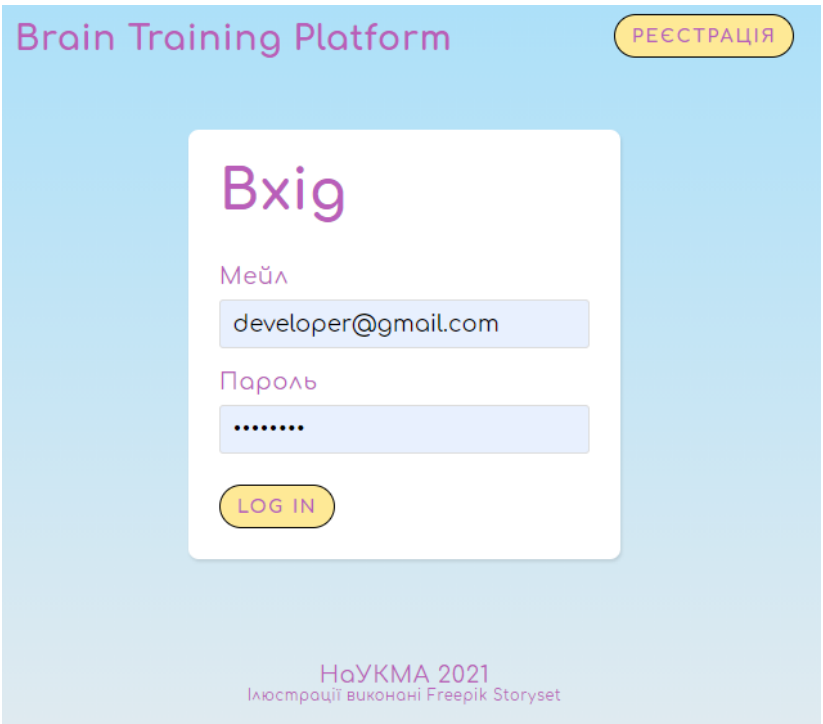
3.7.10 Функція редагування допису на головній сторінці сайту

На сторінці редагування поста автоматично по дефолту підвантажуються дані, які уже є в дописа. Контент-менеджер у свою чергу може змінити щось, додати або прибрати (див. рис. 3.7.11).



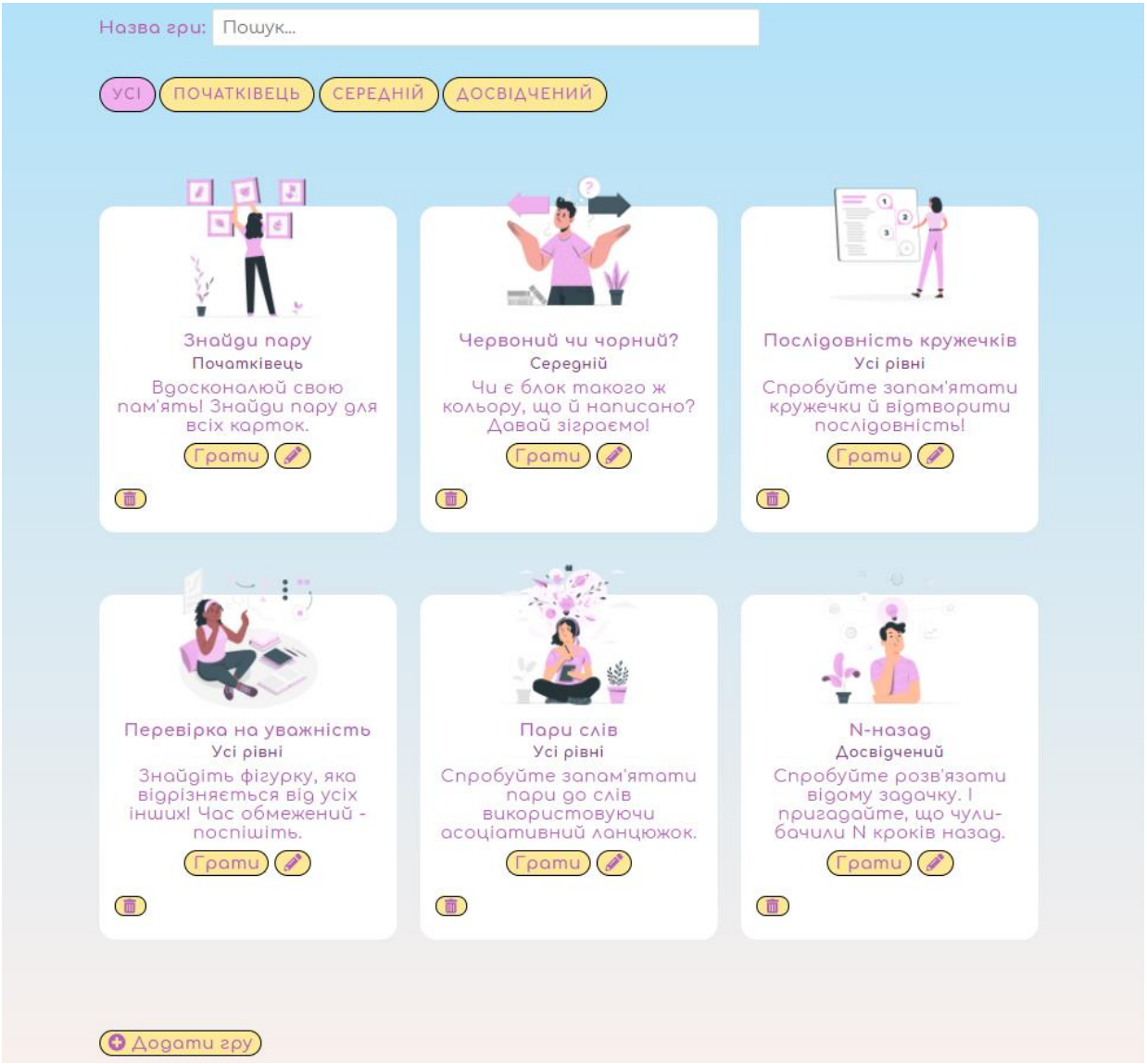
3.7.11 Редагування допису

Користувач з роллю Developer має відповідний мейл та пароль, що дозволяють йому ввійти в систему під своєю роллю (див. рис. 3.7.12).



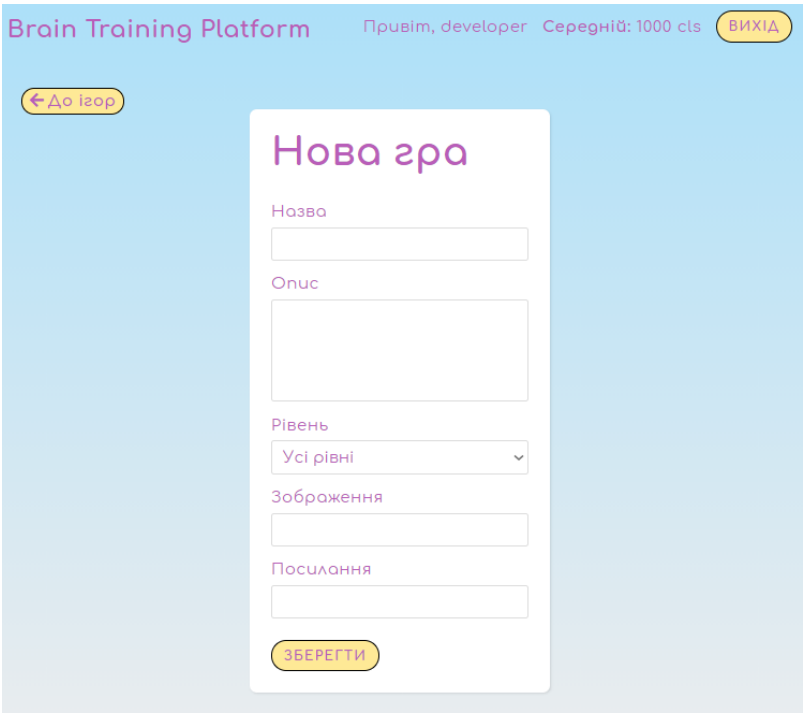
3.7.12 Вхід Developer

Developer має усі ті права, що і звичайний залогінений користувач плюс право на додавання, редагування та видалення ігор, що знаходяться на сторінці з усіма іграми (див. рис. 3.7.13). Ця функція є доступною лише користувачу з роллю Developer. На сторінці з іграми він має доступними усі ігри, що існують в системі.



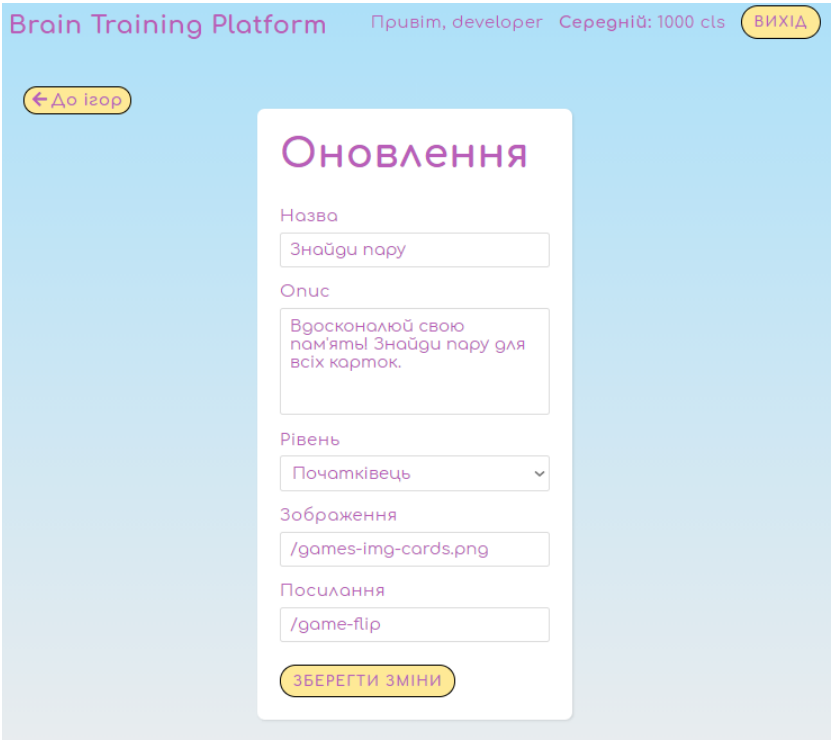
3.7.13 Інтерфейс сторінки з іграми для користувача з роллю Developer

На сторінці з додаванням ігор Developer має обов’язково вказати назву, опис, рівень (вибрати з випадаючого списку), зображення та посилання на гру. Якщо він передумає додавати гру, то може натиснути “До ігор” та повернутися до списку усіх ігор (див. рис. 3.7.14).



3.7.14 Додавання нової гри

На сторінці редагування гри старі параметри будуть вказані уже по дефолту. Developer має можливість їх змінити та зберегти зміни або повернутись до сторінки з усіма іграми (див. рис. 3.7.15).



3.7.15 Оновлення інформації гри, що вже існує

Висновки

Отже, було реалізовано онлайн-сервіс розвитку пам’яті, що відповідає поставленим на початку вимогам, які базувалися на аналізі оглянутих сервісів-аналогів, що існують на сьогоднішній день на ринку послуг. Також було ретельно досліджено предметну область: розглянуто основне призначення пам’яті та когнітивних функцій мозку загалом, вивчено теоретичні відомості та способи покращення пам’яті, вплив та ефективність регулярного виконання практичних вправ. Відбулося ознайомлення з підходами до розробки клієнт-серверних застосунків, які базуються на проектуванні коду за шаблоном MVC. Також було використано одні з найпопулярніших фреймворків для розробки веб сервісів, досліджено та використано функції, що вони пропонують.

Отриманий в результаті веб-застосунок дозволяє користувачам виконувати практичні завдання, що позитивно впливають на розвиток їхніх когнітивних здібностей, слідкувати за своєю статистикою, порівнювати свої досягнення з досягненнями інших користувачів платформи та дізнаватися цікаву пізнавальну інформацію про когнітивні функції мозку та техніки покращення їх функціонування. Також онлайн-сервіс передбачає користувачів з наступними ролями: контент-менеджер, що оновлюватиме секцію пізнавальної інформації та розробника, що додаватиме нові ігри, редагуватиме та видалятиме ті, що вже існують.

Розроблений сайт може бути вдосконаленим у майбутньому: можна додати нові групи користувачів (для прикладу спеціаліста-консультанта, що може надавати поради у сфері когнітивної психології), розширити набір доступних ігор, надати можливість користувачам залишати відгуки на ігри та вибирати гру за популярністю. Відповідно до цих пунктів передбачено монетизацію сайту за додаткові послуги та більший вибір ігор.

Список використаної літератури

1. Стаття про когнітивні функції мозку [Електронний ресурс] https://www.pfizermed.com.ua/public/medical-content/brain-cognitive-functions
2. Нейронна пластичність та когнітивність [Електронний ресурс] https://www.cognifit.com/ru/brain-plasticity-and-cognition
3. 1. Could Brain Training Prevent Dementia?. Brain Train Bugle. 2016. Vol. 2, no. 9. https://www.csusb.edu/sites/default/files/SeptemberBugle_2016-Can-Brain-Training-Prevent-Dementia.pdf.
4. Стаття про влив сучасних технологій на пам’ять людини [Електронний ресурс] https://medconfer.com/en/node/1763
5. Стаття журналу Verywell Mind з рейтингом найкращих сайтів для розвитку мозку [Електронний ресурс] https://www.verywellmind.com/top-websites-and-games-for-brain-exercise-2224140
6. Науково-обґрунтований відгук про сервіс Lumosity [Електронний ресурс] http://www.getyourbreakthrough.com/blog/bid/337603/Lumosity-Brain-Games-A-Review
7. Стаття про MVC [Електронний ресурс] https://www.codecademy.com/articles/mvc
8. Робоча пам’ять [Електронний ресурс] https://www.cognifit.com/science/cognitive-skills/working-memory
9. Miller G. A. The magical number seven, plus or minus two: Some limits on our capacity for processinginformation. Washington D.C : Psychological review, 1956. http://www2.psych.utoronto.ca/users/peterson/psy430s2001/Miller%20GA%20Magical%20Seven%20Psych%20Review%201955.pdf

10. Тест на послідовність [Електронний ресурс] https://www.cognifit.com/cognitive-assessment/battery-of-tests/wom-asm-test/sequential-test
11. Максименко С. Д., Пасічник І. Д. Когнітивна психологія в контексті дослідження пам’яті людини. Наукові записки Національного університету "Острозька академія". Серія : Психологія і педагогіка. 2012. № 20. С. 3–8. https://eprints.oa.edu.ua/1447/1/Pasichnuk_130912.pdf
12. Асинхронність в JavaScript [Електронний ресурс] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing
13. Express.js [Електронний ресурс] https://expressjs.com
14. Використання Canvas [Електронний ресурс] https://flaviocopes.com/canvas/

Додаток А. Код контролеру, що відповідає за статистику

```
const User = require('../models/User');
const jwt = require('jsonwebtoken');

module.exports.my_statistic = (req, res, next) =>{
 const token = req.cookies.jwt;

 // getting user from current session and displaying User statistic
 if (token){
  jwt.verify(token, process.env.JWT_SECRET, async (err, decodedToken) => {
   if (err) {
     console.log(err.message);
     next();
   }
   else {
    await User.findById(decodedToken.id).then(doc => {
      if (doc) {
       const date1 = doc.dayCreated.getTime();
       const date2 = Date.now();
       let days = Math.round((date2 - date1) / (1000 * 3600 * 24));
       let months = Math.round((date2 - date1) / (1000 * 3600 * 24 * 30));
       let years = Math.round((date2 - date1) / (1000 * 3600 * 24 * 30 *12));
      let scorePerDay = (!days) ? "Day has not passed yet." :    (Math.round(doc.score / days * 10)/10) + " cls";
      let scorePerMonth = (!months) ? "Month has not passed yet." : (Math.round(doc.score / months * 10)/10) + "
cls";
      let scorePerYear = (!years) ? "Year has not passed yet." :  (Math.round(doc.score / years * 10)/10) + " cls";
      res.status(200).json({
                          "scorePerDay" : scorePerDay,
                          "scorePerMonth" : scorePerMonth,
                          "scorePerYear" : scorePerYear
                      });
    } else {
          res.status(404).json({
                              message: "No user found"
                  });
    }
  }).catch(err => {
         console.log(err);
         res.status(500).json({
                              error: err
                  });
          });
         next();
       }
     });
  }
}

module.exports.top = async (req, res, next) => {
   await User.find().sort('-score').then(doc => {
       if (doc) {
          res.status(200).json(doc);
       } else {
          res.status(404).json({
             message: "No users found"
          });
       }
   }).catch(err => {
      console.log(err);
      res.status(500).json({
          error: err
```

```
        });
    });
    next();
}
```

Додаток Б. Код гри на відновлення послідовності

```
const canvas = document.querySelector('#canvas');
const c = canvas.getContext('2d');
const timeLimit = document.querySelector('#timeLimit');
const circlesAmount = document.querySelector('#amount');
const btn = document.querySelector('#start');
const circleRadius = 40;

let h = canvas.height = innerHeight - 200;
let w = canvas.width = innerWidth - 375;
let circleArray;
let mouseCoords = { x: undefined, y: undefined };
let hasWon = false;
let isClicked = false;
let foundCirclesArray = [];
let scored;

//почати гру з натиском на кнопку старт
btn.addEventListener('click', () => {
    foundCirclesArray = [];
    init();
    displayTime(circleArray);
});

//позначити кружечок знайденим по кліку
canvas.addEventListener('click', (e) => {
    let canvasCoords = canvas.getBoundingClientRect();
    mouseCoords.x = e.x - canvasCoords.left;
    mouseCoords.y = e.y - canvasCoords.top;
    isClicked = true;
})

//встановлює час впродовж якого користувач може бачити розташування кружечків
function displayTime(arr) {
    setTimeout(() => {
        for (let i = 0; i < arr.length; i++) {
            arr[i].hide();
        }
    }, timeLimit.value * 1000);
}

function distance(x1, y1, x2, y2) {
    return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
}

//кружечки
function makeCircle(x, y, identifier) {
        this.x = x,
        this.y = y,
        this.color = '#471010',
        this.textColor = '#FFFFFF',
        this.identifier = identifier;
```

```
    this.draw = () => {
        c.beginPath();
        c.arc(this.x, this.y, circleRadius, 0, Math.PI * 2, false);
        c.fillStyle = this.color;
        c.fill();
        c.fillStyle = this.textColor;
        c.font = '30px Georgia';
        c.fillText(this.identifier, this.x - 10, this.y + 10);
    }

    this.hide = () => {
        this.color = '#d68686';
        this.textColor = '#d68686';
    }

    this.update = () => {
        if (distance(mouseCoords.x, mouseCoords.y, this.x, this.y) <= circleRadius && isClicked === true) {
            // знайдені кружечки позначаємо у масиві як тру
            foundCirclesArray[this.identifier - 1] = true;
            this.color = '#000000';
            this.textColor = '#ffffff';
        }
        this.draw();
    }
}

//розприділяємо кружечки по канвасу так, щоб вони не накладались одне на одного
function init() {
    circleArray = [];

    for (let i = 0; i < circlesAmount.value; i++) {
        let x = Math.floor(Math.random() * (w - circleRadius * 2) + circleRadius),
            y = Math.floor(Math.random() * (h - circleRadius * 2) + circleRadius),
            n = i + 1;

        if (i !== 0) {
            for (let j = 0; j < circleArray.length; j++) {
                if (distance(x, y, circleArray[j].x, circleArray[j].y) - circleRadius * 2 < 0) {
                    x = Math.floor(Math.random() * (w - circleRadius * 2) + circleRadius),
                        y = Math.floor(Math.random() * (h - circleRadius * 2) + circleRadius);

                    j = -1;
                }
            }
        }
        circleArray.push(new makeCircle(x, y, n))
    }
}

//анімація канвасу за допомогою requestAnimationFrame
function animate() {

    if(parseInt(circlesAmount.value) === foundCirclesArray.length && (Object.values(foundCirclesArray).length
=== foundCirclesArray.length)){
        console.log(foundCirclesArray);
        hasWon = true;
    }

    // якщо гру було виграно, то відправляємо формочку для оновлення результатів користувача
    if(hasWon){
        scored = (circlesAmount.value * 10) / timeLimit.value;
        let roundedScore = Math.round(scored);
```

```
        if(alert('You have scored '+ roundedScore)){}
        else {
            let xhr = new XMLHttpRequest();
            xhr.open("POST", "/update-score", true);
            xhr.setRequestHeader('Content-Type', 'application/json');
            xhr.send(JSON.stringify({
                score: roundedScore
            }));
            // почекаємо, щоб оновився рахунок
            setTimeout(() => {
                window.location.reload();
            }, 1000);
        }
    }
    else{
        requestAnimationFrame(animate);
    }
    c.clearRect(0, 0, w, h);

    //гра починається спочатку, якщо було натиснуто неправильний кружечок
    if (foundCirclesArray.length > 0) {
        for (let i = 0; i < foundCirclesArray.length; i++) {
            if (foundCirclesArray[i] !== true) {
                foundCirclesArray = [];
                init();
                displayTime(circleArray);
            }
        }
    }

    for (let i = 0; i < circleArray.length; i++) {
        circleArray[i].update();
    }

    isClicked = false;

}
init();
displayTime(circleArray);
animate();
```

Додаток В. Код серверу

```
const express = require('express');
const mongoose = require('mongoose');
const morgan = require('morgan');
const bodyParser = require('body-parser');
const dotenv = require('dotenv').config();
const authRoutes = require('./routes/authRoutes');
const gamesRoutes = require('./routes/gamesRoutes');
const statisticsRoutes = require('./routes/statisticsRoutes');
const postRoutes = require('./routes/postRoutes');
const cookieParser = require('cookie-parser');
const { requireAuth, checkUserAndGetLocals } = require('./middleware/authMiddleware');

const app = express();
const PORT = process.env.PORT || 3000;

// middleware
```

```
app.use(express.static('public'));
app.use(express.json());
app.use(cookieParser());

// logs
app.use(morgan('dev'));

// view engine
app.set('view engine', 'ejs');

// database connection
const dbURI = 'mongodb+srv://brain-training:' + process.env.MONGO_ATLAS_PW
            + '@braintraining.p4zyl.mongodb.net/' + process.env.MONGO_ATLAS_NM;
mongoose.connect(dbURI, { useNewUrlParser: true,
                          useUnifiedTopology: true,
                          useCreateIndex:true })
       .then((result) => app.listen(PORT))
       .catch((err) => console.log(err));


app.use(bodyParser.urlencoded({extended:false}));
app.use(bodyParser.json());

// routes
app.use((req, res, next) => {
   res.header('Access-Control-Allow-Origin', '*');
   res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');
   if(req.method === 'OPTIONS'){
      res.header('Access-Control-Allow-Methods', 'GET, PUT, POST, PATCH, DELETE');
      return res.status(200).json({});
   }
   next();
});

app.get('*', checkUserAndGetLocals); //to set user for ejs-templates
app.get('/', (req, res) => res.render('home'));
app.get('/test', requireAuth, (req, res) => res.render('test'));
app.use(postRoutes);
app.use(authRoutes);
app.use(gamesRoutes);
app.use(statisticsRoutes);
```

Додаток Г. Клієнтський код

Головна сторінка home.ejs

```
<%- include('partials/header'); -%>

<header>
  <div class="brain">
    <img src="/images/brain.png" alt="">
  </div>
  <div class="headings">
    <h2>Кльові ігри</h2>
    <h3>для когнітивних тренувань</h3>
    <a href="/games" class="btn">Грати</a>
  </div>
</header>
```

```
<hr>

<div class="container row">
  <div class="top col">
    <h3>Топ-3</h3>
    <img src="/images/home-img-top.png" alt="">
    <div class="top__info" id="top">
    </div>
  </div>
  <% if (user) { %>
  <div class="statistic col">
    <h3>Моя статистика</h3>
    <img src="/images/home-img-statistic.png" alt="">
    <div class="statistic__info" id="my-statistic">
    </div>
  </div>
  <% } %>
</div>

<hr>

<div class="container news">
  <% if (post) { %>
    <h3> <%= post.header %></h3>
    <img src="/images<%= post.imageLink %>" alt="">
    <div class="news__info">
      <%= post.description %>
    </div>
    <video controls>
      <source src="/videos<%= post.videoLink %>" type="video/mp4">
      Пробачте, Ваш браузер не підтримує це відео.
    </video>
  <% } %>
</div>
<% if (user && post) { %>
  <% if (manager) { %>
    <div class="container">
      <br>
        <a href="/update-post/<%= post._id %>" class="btn btn-game">Оновити допис</a>
    </div>
  <% } %>
<% } %>

<hr>

<script src='/js/home.js'></script>
<%- include('partials/footer'); -%>
```

Код, що оновлює дані сторінки home.js

```
function fetchdata(){
   $.ajax({
       url: '/top',
       type: 'GET',
       headers: {
           "Content-Type":"application/json"
       },
       success: (response) => setTop(response),
       complete: function(data){
           setTimeout(fetchdata,10000);
```

```
        }
    });
}

function setStatistic(data) {
    let $statDiv = $("#my-statistic");
    $statDiv.empty();
    $statDiv.append("<p>У день: " + data.scorePerDay + "<p>" +
        "<p>У місяць: " + data.scorePerMonth + "<p>" +
        "<p>У рік: " + data.scorePerYear + "<p>"
    );
}

function setTop(data) {
    let $appShow = $("#top");
    $appShow.empty();
    if(data.length > 2){
        for(let i = 0; i < 3; i++){
            $appShow.append(
                "<p>" + data[i].score
                + " cls : " + getName(data[i].email) + "</p>"
            );
        }
    }
    else{
        data.forEach((player) => {
            $appShow.append(
                "<p>" + player.score
                + " cls : " + getName(player.email) + "</p>"
            );
        });
    }
}

function getStatistic() {
    $.ajax({
        url: '/my-statistic',
        type: 'GET',
        headers: {
            "Content-Type":"application/json"
        },
        success: (response) => setStatistic(response),
        complete: function(data){
            setTimeout(getStatistic,5000);
        }
    });
}

function getName(email){
    return email.substring(0, email.lastIndexOf("@"))
}

$(document).ready(function(){
    fetchdata();
    getStatistic();
});
```

Додаток Д. Діаграма класів