Міністерство освіти і науки України
Прикарпатський національний університет імені Василя Стефаника
кафедра комп’ютерних наук та інформаційних систем

БАКАЛАВРСЬКА РОБОТА

на тему: Гейміфікована система підтримки вивчення англійської мови для дітей

Виконав: студент 4 курсу гр.ІСТ-41
Спеціальності 126 Інформаційні системи та технології
Сорока Ростислав Ігорович
Керівник: к.т.н., старший викладач, Ізмайлов Артем Вікторович
Рецензент: старший викладач Никорак Я.Я

м. Івано-Франківськ – 2024

АНОТАЦІЯ

Ця робота присвячена розробці веб-сервісу для вивчення англійської мови за допомогою гейміфікації. Вона складається зі вступу, трьох розділів, висновків та списку використаних джерел і додатків. Розглядається застосування різноманітних ресурсів та інтерактивних методів для покращення ефективності навчання. Дослідження містить 16 рисунків, 3 таблиці, 15 посилань та 2 додатки.

Вона також включає розділи технічного дизайну, програмування та тестування. Розроблено веб-сервіс, який поєднує вивчення англійської мови з гейміфікацією.

Автор є призером Міжнародної науково-технічної конференції здобувачів вищої освіти та молодих вчених “Комп’ютерні науки, інформаційні технології та системи управління”.

Ключові слова: Англійська мова, Веб-сервіс, Гейміфікація, Інтерактивні методи навчання, Тестування.

ABSTRACT

This work is dedicated to the development of a web service for learning the English language using gamification. It consists of an introduction, three chapters, conclusions, a list of references, and appendices. The use of various resources and interactive methods to improve the effectiveness of learning is considered. The research includes 16 figures, 3 tables, 15 references, and 2 appendices. It also includes sections on technical design, programming, and testing. A web service has been developed that combines learning the English language with gamification.

The system uses interactive methods to enhance learning. The work covers all stages of development, including technical and pedagogical aspects of learning.

The author is a laureate of the International Conference of Young Scientists 2024.

Keywords: english language, web service, gamification, interactive learning methods, testing.

ЗМІСТ

ВСТУП

Актуальність: В сучасному світі вивчення іноземних мов стає все більш важливою складовою освіти та особистісного розвитку. Проблема ефективного засвоєння англійської мови залишається актуальною в умовах швидкого змінного інформаційного середовища. Створення веб-сервісу для вивчення англійської є стратегічно важливим кроком, оскільки враховує сучасні тенденції в освіті та використання технологій [1].

Гейміфікація в освіті, зокрема у вивченні мов, значно підвищує мотивацію та залученість учнів. Впровадження елементів гри в навчальний процес робить його більш захопливим та інтерактивним. Завдяки гейміфікації можна створювати умови для постійного виклику та заохочення, що стимулює учнів до регулярного навчання та досягнення нових результатів. Веб-сервіс з елементами гейміфікації може включати різноманітні ігрові завдання, рівні складності, системи нагород та змагань, що сприяє більш ефективному засвоєнню матеріалу та підтримці високого рівня мотивації. Таким чином, гейміфікація є важливим інструментом, який допомагає адаптувати навчальний процес до потреб сучасних учнів та забезпечує цікавий підхід до вивчення англійської мови [1].

Мета дослідження є розробка гейміфікованої системи для вивчення англійської мови у формі веб-сервісу,спрямованого на покращення якості та ефективності навчання.

Предметом дослідження є гейміфікованована система для вивчення англійської мови у формі веб-сервісу, **об'єктом** — процес вивчення англійської мови за допомогою ігрових додатків. **Методи дослідження** включають аналіз літературних джерел, тестування та апробацію розробленого веб-сервісу.

Задача полягає в необхідності розробки засобу для навчання, який би стимулював активне засвоєння англійської мови та вирішував проблему низького рівня мотивації учнів до вивчення іноземної мови.

У першому розділі здійснюється аналіз предметної галузі та існуючих додатків, що стосуються вивчення англійської мови. Визначаються їхні переваги

та недоліки, проводиться дослідження та порівняння різних аспектів навчання англійської мови.

У другому розділі проводиться аналіз різноманітних технологій з метою визначення найбільш перспективних методів та технологій для ефективної реалізації поставленого завдання. Результатом цього аналізу є формування кінцевого стеку технологій, який оптимально відповідає потребам та вимогам проекту. У цьому розділі також розглядається архітектура проекту, в якій описуються основні компоненти системи, їх взаємодія та функціональні можливості.

У третьому розділі, присвяченому практичній реалізації поставленої задачі, проводиться виведення кодової реалізації та тестування програмного продукту. Детальний аналіз програмного коду дозволяє висвітлити ключові аспекти та архітектурні рішення. У цьому контексті також виконується серія тестів для перевірки функціональності та ефективності розробленого програмного рішення. Особлива увага зосереджується на виявленні можливих покращень та виправленні виявлених дефектів.

РОЗДІЛ 1.АНАЛІЗ ПРЕДМЕТНОЇ ГАЛІЗУ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз Методик Вивчення Англійської Мови

1.1.1 Актуальність вивчення англійської мови в сучасному світі

Вивчення англійської мови в дитячому віці має велике значення для подальшого освітнього й кар'єрного розвитку. У сучасному світі, де глобалізація відіграє ключову роль, володіння англійською мовою стає необхідністю. Аналіз предметної галузі полягає в дослідженні педагогічних підходів та методик, що використовуються для навчання англійської мови дітей [1]. Традиційні методи, такі як викладання з підручників та виконання вправ, порівнюються з інноваційними підходами, такими як ігрові методики та використання інтерактивних технологій.Цей аналіз допомагає зрозуміти, які підходи до вивчення англійської мови є найбільш ефективними для дітей та як можна вдосконалити існуючі методики [2].

1.1.2 Традиційні методи навчання

Традиційні методи навчання, такі як викладання з підручників та проведення уроків,мають свої **переваги** [3]:

1.Структурований підхід: Традиційні методи надають структурований навчальний план, що допомагає учням організувати своє навчання.

2.Академічна підготовка: традиційні методи надають важливі академічні знання та навички, такі як граматика та лексика.

Проте є й **недоліки**:

1.Нецікавий підхід: Для деяких учнів традиційні методи можуть бути нудними та мало захоплюючими.

2.Мало практики: Часто ці методи мало зосереджені на практичному використанні мови в реальних ситуаціях.

1.1.3 Гейміфіковані методи навчання

Гейміфіковані методи навчання стають все більш популярними у сучасному освітньому середовищі [3]. Вони використовують елементи гри та інтерактивності для стимулювання мотивації та покращення результатів навчання [3]. Гейміфікація має наступні **переваги**:

1.Захоплюючий підхід: Гра як елемент гейміфікації робить навчання більш захоплюючим та цікавим для учнів, спонукаючи їх активно залучатися до процесу вивчення.

2.Мотивація до досягнення цілей: Елементи змагання, винагороди та досягнення мотивують учнів до активності та досягнення поставлених цілей.

3.Підвищення зацікавленості: Гра дозволяє навчальному матеріалу представлятися у більш привабливому та легкозасвоюваному форматі, що збільшує зацікавленість учнів у вивченні [3].

4.Індивідуалізований підхід: Гейміфіковані методи навчання можуть бути адаптовані до індивідуальних потреб учнів, забезпечуючи більш ефективний процес навчання [3].

Проте є і **недоліки**

1.Перенасичення гри: Надмірне використання ігрових елементів може відволікати учнів від основного навчального матеріалу. Вони можуть зосереджуватися більше на грі, ніж на самих знаннях.

2.Нерівномірний розподіл уваги: Гейміфікація може призвести до того, що учні концентруються лише на досягненні нагород чи балів, ігноруючи важливі, але менш захоплюючі аспекти навчання.

3.Мотиваційні ризики: Винагороди та змагальність можуть спонукати до короткострокової мотивації, яка зникає після завершення гри або отримання винагороди. Це може негативно вплинути на внутрішню мотивацію учнів до навчання.

1.1.4 Інтерактивні технології

Застосування інтерактивних технологій, таких як мобільні додатки та онлайн-ресурси, стає все більш популярним методом вивчення англійської мови [3].

Вони мають такі **переваги**:

1.Гнучкість: Інтерактивні технології можуть бути доступними для вивчення в будь-який час та в будь-якому місці.

2.Індивідуалізація: Вони дозволяють учням навчатися у зручному для них темпі, з урахуванням їхніх індивідуальних потреб [3].

Проте існують такі **недоліки**:

1.Відсутність взаємодії з викладачем: Використання інтерактивних технологій може призвести до відсутності особистого взаємодії з викладачем, що може бути важливим для деяких учнів.

2.Потребують доступу до технологій: Не всі учні можуть мати доступ до необхідних технологій для вивчення англійської мови в цьому форматі.

Загальний Висновок:

У порівнянні з іншими методологіями вивчення англійської мови, гейміфіковані підходи виявляються більш привабливими та ефективними у залученні учнів до процесу навчання, підвищенні їхньої мотивації та поліпшенні результатів. Гейміфікована система підтримки вивчення англійської мови для дітей має потенціал стати найбільш ефективним інструментом для досягнення успіху у вивченні мови [3].

1.2 Переваги та недоліки доступних гейміфікованих інструментів для вивчення англійської мови

1.2.1 Загальний аналіз

Використання технологій у вивченні мови передбачає застосування різних електронних інструментів та програмних рішень для покращення процесу вивчення мови [4].

1.Мобільні додатки для вивчення мови:Існує низка мобільних додатків для вивчення мови на смартфонах і планшетах. Ці додатки включають практику вимови, інтерактивні завдання, аудіо- та відеоматеріали.

2.Онлайн-платформи для дистанційного навчання: Технологія дистанційного навчання дозволяє учням проходити онлайн-заняття, виконувати завдання та взаємодіяти з викладачами та іншими учнями за допомогою відеоконференцій та платформ електронного навчання [4].

3.Інтерактивні вправи та ігри: Технології дозволяють створювати цікаві інтерактивні вправи та ігри для вивчення мови, які ефективно розвивають граматику, словниковий запас, навички читання та аудіювання.

4.Автоматизована перевірка граматики та вимови: веб-сервіси та додатки можуть автоматично перевіряти граматику учнів і надавати зворотній зв'язок та пропозиції для покращення навичок [4].

Такі технології не лише полегшують доступ до навчальних ресурсів, але й роблять процес вивчення мови більш цікавим, ефективним і корисним для учнів.

1.2.2 Додаток LearnEnglish Kids

Аналіз існуючих рішень показав, що більшість з них є іграми, але без користі для вивчення англійської мови, тобто гра є своєрідною "винагородою" за вивчення певного уроку.Ілюстративним прикладом є LearnEnglish Kids – сайт з симуляторами та завданнями (рис. 1.1) [5].

Рисунок 1.1 - Інтерфейс веб-сайту LearnEnglish Kids

У випадку з цим додатком сам процес навчання не є гейміфікований, гейміфікувалися лише додаткові ігри, не пов'язані з вивченням англійської мови.

Переваги:

- Додатковий мотиваційний фактор:

Можливість зіграти в гру може стати додатковим мотиваційним фактором для дітей і спонукати їх активно вивчати англійську мову, щоб отримати доступ до ігор.

- Навчальні ігри:

Якщо ігри, доступні в додатку, спрямовані на розвиток таких навичок, як логіка, творчість і співпраця, вони можуть сприяти загальному розвитку дітей.

Недоліки:

- Відсутність гейміфікації в навчальних модулях:

Відсутність гейміфікації в самому навчальному процесі може зробити його менш цікавим, оскільки діти можуть відчути відсутність стимулюючого елементу.

- Обмежена глибина знань:

Через обмежену глибину знань у деяких розділах, LearnEnglish Kids менше підходить для дітей з більш високим рівнем знань.

1.2.3 Додаток Funbrain

Наступне рішення, яке було проаналізовано - це веб-додаток FunBrain (рис. 1.2). Ця платформа має ті ж недоліки, що й попередній проаналізований застосунок. А саме, ігри є лише додатковою частиною і не дають жодної користі для вивчення англійської мови [6].

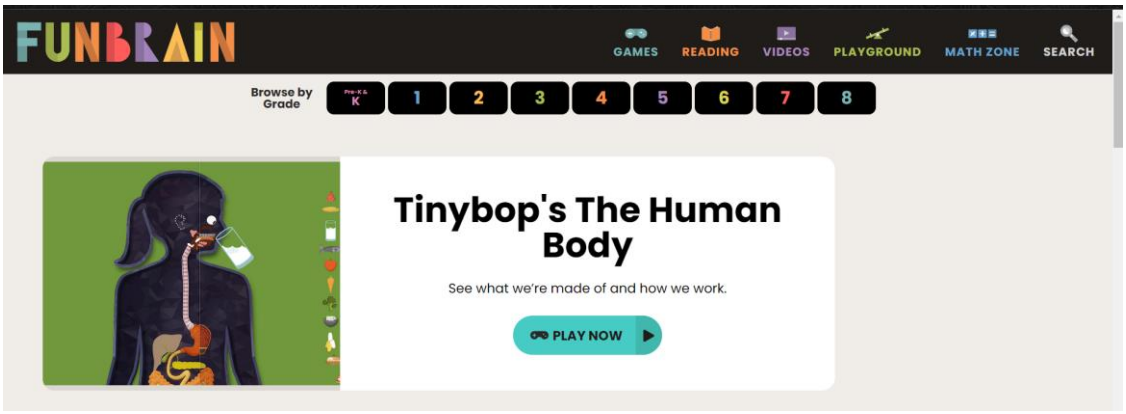


Рисунок 1.2 - Інтерфейс веб-додатку Funbrain

Приклади дослідження також не є інтерактивні, наприклад, у класі читання учні читають лише комікси, і цей процес не гейміфікований (рис. 1.3).

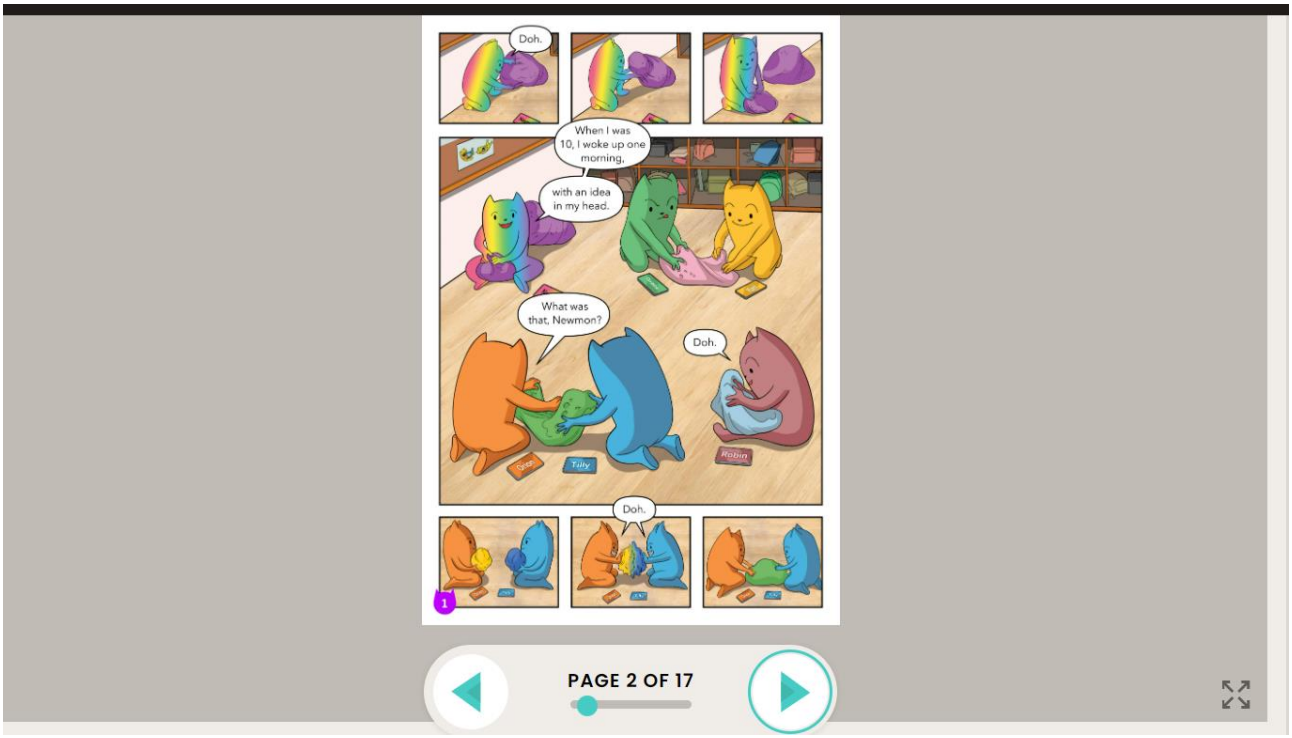


Рисунок 1.3 - Інтерфейс коміксу FunBrain

Переваги:

- Підвищена мотивація за рахунок активностей:

Ігри можуть додатково стимулювати інтерес до вивчення англійської, особливо якщо ваша дитина любить читати мангу та інші цікаві заняття.

- Підходить для різних вікових груп

FunBrain може бути адаптований для різних вікових груп і дозволяє дітям на різних етапах розвитку знаходити цікаві та відповідні завдання.

Недоліки:

- Відсутність гейміфікації в навчальних модулях:

Оскільки в самому навчальному процесі відсутня гейміфікація, користувачі можуть вважати процес навчання непривабливим, оскільки йому бракує стимулюючих елементів. А також у учня можливо швидко спаде інтерес до цього додатку.

- Відсутність управління контентом:

Через відсутність власної системи управління контентом деякі навчальні матеріали можуть не відповідати стандартам батьків та вчителів.

1.2.4 Додаток WordClouds

Наступним проаналізованим додатком був веб-додаток WordCloud.com (рис. 1.4).

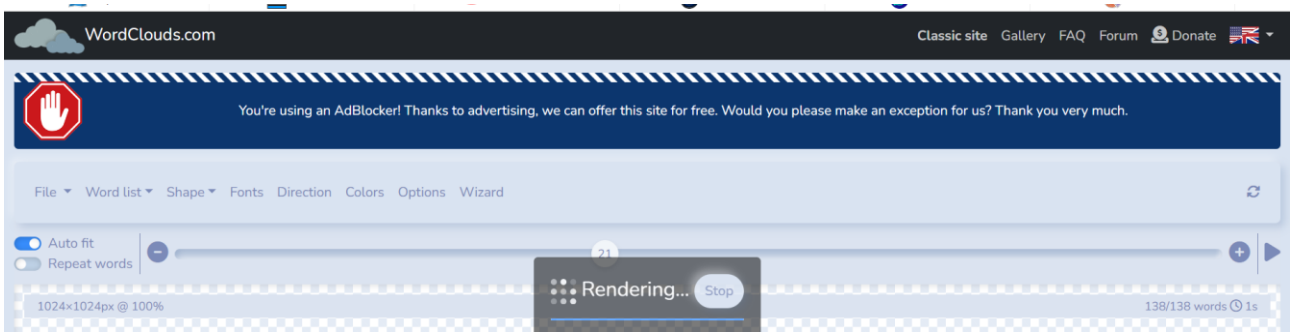


Рисунок 1.4 - Інтерфейс головної сторінки WordClouds

Процес вивчення англійської мови полягає лише у створенні картинок, з яких будуються списки слів (рис. 1.5) [7].

Рисунок 1.5 - Інтерфейс самого застосунку

Переваги:

- Простота використання:

WordClouds пропонує інтуїтивно зрозумілий і простий у використанні інтерфейс, який дозволяє створювати хмари слів користувачам, які не є експертами.

- Безкоштовний сервіс:

WordClouds пропонує безкоштовний доступ до базового функціоналу, що робить його доступним для широкого кола користувачів.

Недоліки:

- Відсутність реєстрації та зберігання:

Немає можливості зберігати створені WordClouds, що може ускладнити їх повторне використання та організацію.

- Вузький спектр можливостей:

У порівнянні з іншими сервісами, WordClouds пропонує вузький спектр опцій та налаштувань.

Аналіз існуючих методів вивчення англійської мови показав, що багато рішень пропонують ігрові елементи, але вони не обов'язково спрямовані на активне вивчення мови. Деякі ігри функціонують лише як "винагорода" після завершення уроку і не передбачають додаткового вивчення мови під час

гри.Гейміфікація в цих рішеннях є,але вона не є центральним прийомом для навчання,тобто гра як елемент а не як метод.

1.3 Постановка завдання.

Завдання полягає в розробці інтерактивного веб-сервісу для вивчення англійської мови. Веб-сервіс буде складатись з гри “Шаради” та ігрового платформеру,який розрахований на гравців різних вікових груп дітей.

Основні завдання.

1.Розробка інтерактивного інтерфейсу: створити привабливий та зручний інтерфейс, щоб користувачі могли легко взаємодіяти з грою.

2.Механізм генерації завдань: Створити механізм генерації слів та фраз для шарад, що забезпечить різноманітність та цікавість.

3.Оцінювання та підрахунок балів: визначення правильності чи неправильності відповідей гравців, підрахунок балів та управління статистикою.

4.Розробка платформеру: реалізація різних механів,створення середовища,створення візульної частини гри та гемплею.

5.Адаптивність і сумісність: забезпечення адаптивності до різних пристроїв і операційних систем та сумісності з різними веб-браузерами.

Висновки до першого розділу

У цьому розділі було проведено детальний аналіз предметної області, пов'язаних з вивченням англійської мови, та сформульовано постановку задачі щодо розробки електронного засобу з використанням гейміфікації.

У результаті обгрунтовано розробку системи підтримки вивчення англійської мови . У наступному розділі розглянуто деталі та реалізація додатку для досягнення поставленої мети та завдань.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ АРХІТЕКТУРИ СИСТЕМИ ТА ВИБІР ТЕХНОЛОГІЙ

2.1 Вибір технологій

2.1.1 Аналіз технологій для реалізації Frontend складової

Для реалізації клієнтської частини необхідно використати один з JavaScript фреймоворків [8].

Для порівнняння було обрано 2 фреймворки такі як **React** та **Vue.js**

Таблиця порівняння:

Таблиця 2.1 – Результати статистичного дослідження обох фреймворків

Особливості	**React**	**Vue**
Синтаксис	JSX(JavaScriptXML)	HTML-подібний шаблон
Стан	Зберігається у власних змінних	Має власну систему реактивності
Компоненти	Функціональні та класові	В основному тільки функціональні
Управління станом	Сторонні бібліотеки такі як Redux,MobX	Має вбудований механізм-Vuex
Компактність	Легкий і компактний	Менш здатний для великих проєктів
Популярність	Широке використання	Швидко набирає популярність
Властивості	Відмінна оптимізація	Простота в освоєнні

Переваги та недоліки:

React:

- **переваги:**

  -Широко використовується в промислових проектах.

  -Велика кількість сторонніх бібліотек для різних задач [9]

  -Ефективна віртуалізація DOM забезпечує високу продуктивність.

- **недоліки:**

-Вимагає використання сторонніх бібліотек для керування станом.

-Часто використовувані шаблони потребують додаткового коду [10].

Vue:

- **переваги:**

-Простий у вивченні та використанні, особливо для початківців.

-Має вбудовану систему управління станом (Vuex).

-Потужний механізм реактивності для автоматичного оновлення даних.

- **недоліки:**

-Менш ефективний для великих проектів порівняно з React.

-Менша кількість готових бібліотек та плагінів порівняно з React.

Висновок:

Обидва фреймворки, React і Vue, мають свої переваги та недоліки, і вибір між ними залежить від конкретних потреб проекту, рівня досвіду розробника та особистих вподобань. Однак, вибір на користь **React** є більш обґрунтованим для проектів, які вимагають високої продуктивності та масштабованості. Завдяки своїй гнучкості, великій екосистемі та активній спільноті, React забезпечує ефективне управління складними інтерфейсами та є оптимальним рішенням для великих проектів. Vue, натомість, часто обирається для менших проектів або для початківців через його легкість вивчення та використання.

2.1.2 Аналіз технологій для реалізації Backend частини

Для реалізації серверної частини проекту необхідно використати один з Python фреймоворків фреймоворків [11].

Для порівнняння було обрано Flask та Django

Порівняльна таблиця:

Таблиця 2.2 – Результати статистичного дослідження обох фреймворків

Особливості	Flask	Django
Маштабованість	Підходить для невеликих та середніх проектів, більш гнучкий	Підходить для великих та складних проектів, має вбудовану адміністративну панель
Складність	Простий та легкий для вивчення	Більше складний та потужний, вимагає більше часу для освоєння
Архітектура	Мікрофреймворк, не має вбудованих компонентів, кожен функціонал додається окремо за потребою	Монолітний фреймворк з вбудованими компонентами, такими як ORM, адміністративна панель тощо
Безпека	Не має вбудованих заходів безпеки, вимагає вручну додавати необхідні заходи	Має вбудовані заходи безпеки, такі як вбудована система аутентифікації та авторизації
Швидкодія	Швидкий в порівнянні з Django	Може бути повільнішим через більший обсяг функціоналу
Спільнота	Має меншу,але активну спільноту	Має велику та активну спільноту,що сприяє розвитку

Переваги та недоліки:

Flask:

- **Переваги:**

-Простота вивчення та використання, особливо для початківців [12].

-Ідеально підходить для швидкого створення прототипів та малих проектів.

-Гнучкий підхід до розробки, дозволяє вибирати та використовувати лише необхідні компоненти.

- **Недоліки:**

-Вимагає додаткових заходів безпеки, таких як власноручна настройка.

-Менш підходить для великих та складних проектів через відсутність вбудованих компонентів.

Django:

- **Переваги:**

-Вбудована адміністративна панель для швидкого створення та керування вмістом.

-Має вбудовані заходи безпеки та аутентифікації, що полегшує розробку безпечних додатків.

-Ідеально підходить для великих та складних проектів завдяки вбудованому функціоналу.

- **Недоліки:**

-Більша складність вивчення та розгортання порівняно з Flask.

-Може бути надто монолітним для деяких проектів, що ускладнює підтримку та розширення.

Висновок:

Обидва фреймворки, Flask і Django, мають свої переваги та недоліки, і вибір між ними залежить від конкретних потреб проекту, рівня досвіду розробника та особистих вподобань. Однак, вибір на користь **Flask** є більш обґрунтованим для цього проекту. Flask часто обирається для невеликих та середніх проектів або там, де важлива швидкість розробки, завдяки своїй легкості та гнучкості. Django, з іншого боку, найбільше підходить для великих

та складних проектів, де важлива вбудована функціональність та безпека. Flask дозволяє швидко створювати та розгортати проекти, що є ключовим фактором для нашого проекту.

2.1.3 Аналіз технологій для реалізації гри в проекті

Для реалізації ігри в проекту було обрано JS Canvas. JavaScript Canvas - це HTML-елемент, який дає можливість малювати графіку за допомогою JavaScript. Він дозволяє створювати анімацію, графіку, ігри та інші візуальні ефекти без використання сторонніх бібліотек. Canvas надає низькорівневий доступ до пікселів на екрані, що робить його потужним інструментом для створення ігор [13].

Аспекти вибору Canvas:

1.Простота використання: Canvas дозволяє малювати на екрані будь-яку графіку з допомогою JavaScript, що робить його доступним для більшості розробників.

2.Низькорівневий доступ: Відсутність абстракцій над пікселями дозволяє повністю контролювати відображення на екрані, що корисно при створенні складних ігрових ефектів.

3.Висока продуктивність: Canvas використовує апаратне прискорення для малювання графіки, що забезпечує високу продуктивність навіть у великих іграх.

Для порівняння було обрано SVG(Scalable Vector Graphisc).

Таблиця 2.3 – статистичне порівняння обох інструментів

Особливості	Canvas	SVG
Тип графіки	Растрова(піксельна)	Векторна
Структура	Імперативна	Декларивна
Взаємодія з DOM	Мало зв’язаний з DOM	Інтегрований з DOM
Продуктивність	Зазвичай краща для складних ігор	Краща для мало інтерактивних малюнків

Анімація	Можлива,але потребує JavaScript коду	Простіше досягнення анімації за допомогою CSS

Переваги і недоліки:

Canvas:

.Переваги:

-Ідеально підходить для складних анімацій та ігор.

-Забезпечує кращу продуктивність для складних графічних ефектів.

-Дозволяє малювати будь-які графічні елементи з допомогою JavaScript.

.Недоліки:

-Роздільна здатність залежить від розмірів вікна браузера, що може - призвести до втрати якості при масштабуванні.

-Вимагає складнішого коду для досягнення складних ефектів порівняно з SVG.

SVG:

.Переваги:

-Простий для маніпулювання та анімації окремих елементів.

-Краще масштабується без втрат якості, оскільки базується на векторах.

-Зручний для створення статичних або малоінтерактивних малюнків.

.Недоліки:

-Менш підходить для складних анімацій та ігор.

-Може викликати проблеми з продуктивністю для складних графічних ефектів.

Висновок:

Canvas і SVG - це два різних підходи до малювання графіки веб-додатками. **Canvas** краще підходить для складних анімацій та ігор, оскільки дозволяє малювати растрову графіку та маніпулювати пікселями, в той час як SVG

зручний для створення статичних або малоінтерактивних малюнків, оскільки базується на векторах та забезпечує краще масштабування без втрат якості.

2.1.4 Підходи гейміфікації в додатку

Гейміфікація, особливо у формі тестів, може бути дуже ефективним методом для вивчення слів з кількох причин:

1.Залучення: Гра або тест надає можливість взяти участь в активності, яка може бути цікавою і захоплюючою, що збільшує мотивацію до вивчення слів.

2.Контекстуалізація: Гра або тест може створити контекст, де слова використовуються в реальних або симульованих ситуаціях, що полегшує їх запам'ятовування та розуміння.

3.Нагороди та досягнення: Системи нагород, досягнень та лідерства можуть бути використані для стимулювання учнів, збільшуючи їхню мотивацію до вивчення та досягнення кращих результатів.

4.Повторення: Повторення є ключовим для запам'ятовування слів. Гейміфіковані тести можуть пропонувати циклічні завдання або періодичні випробування, що змушують учнів регулярно повторювати вивчений матеріал.

5.Адаптація до індивідуальних потреб: Деякі гейміфіковані платформи можуть адаптуватися до індивідуальних потреб користувачів, надаючи додаткові вправи або пояснення там, де вони необхідні.

Отже, використання гейміфікації, зокрема тестів, для вивчення слів може забезпечити ефективний і захопливий спосіб навчання, який стимулює учнів до активності та допомагає їм краще розуміти та запам'ятовувати матеріал.

Використання гейміфікації у формі платформера для вивчення слів може мати кілька переваг:

1.Інтерактивність: Платформери надають можливість взаємодії з матеріалом через рух персонажа, що може зробити навчання більш захоплюючим та привабливим для учнів.

2.Візуалізація: Графіка та анімація в платформерах можуть допомогти візуалізувати слова та їх значення, що полегшує їх розуміння та запам'ятовування.

3.Контекстуалізація: Платформер може створити віртуальне середовище, де слова використовуються в різних ситуаціях або контекстах, що сприяє їх засвоєнню.

4.Розвиток навичок: Гра у стилі платформера може допомогти розвивати різноманітні навички, такі як швидкість реакції, координація рухів, проблемне мислення та стратегічне планування.

5.Мотивація досягнень: Впровадження елементів гейміфікації, таких як бонусні рівні, віртуальні нагороди та лідерські дошки, може стимулювати учнів до активної участі та досягнення кращих результатів.

Отже, використання платформера як форми гейміфікації для вивчення слів може бути ефективним методом, що сприяє активному навчанню, розвитку навичок та збільшенню мотивації учнів.

2.2 Архітектура веб-додатку

Архітектура веб-додатку є ключовим аспектом його розробки, визначаючи структуру та взаємодію між різними компонентами системи. Від ефективного дизайну архітектури залежить продуктивність, масштабованість та безпека додатку. Вибір правильної архітектури дозволяє забезпечити оптимальну роботу користувацького інтерфейсу, надійне управління даними та ефективне виконання бізнес-логіки. Завдяки розподіленій архітектурі можна досягти високої стійкості системи до навантажень і забезпечити легкість інтеграції з іншими сервісами [14].

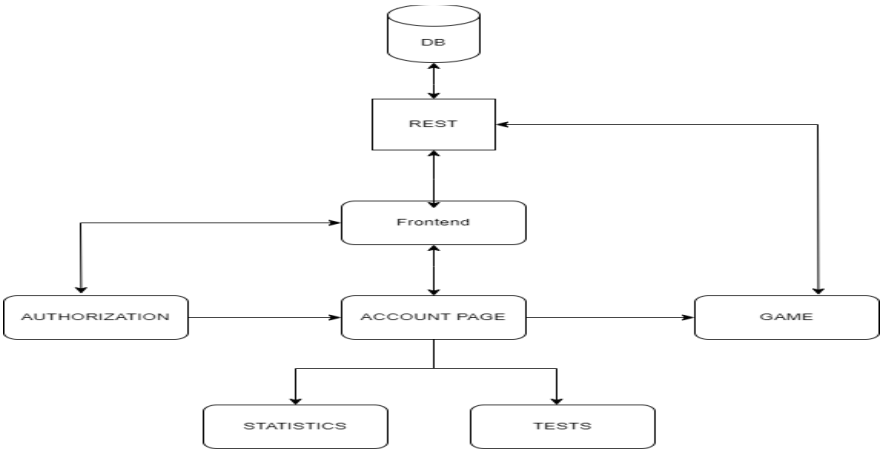
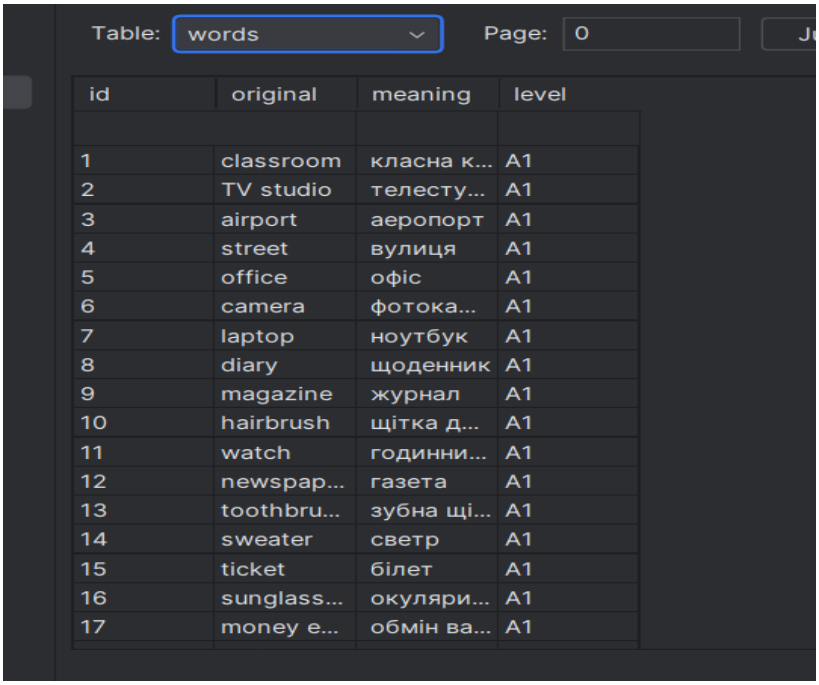


Рисунок 2.1 Інтерфейс архітектури додатку

DB (DataBase) - це база даних, яка зберігає таблиці, такі як: test_cache, test_question, user, word, які в свою чергу містять різні поля. Наприклад, структура таблиці слів (див. Рисунок 2.2).



id	original	meaning	level
1	classroom	класна к...	A1
2	TV studio	телесту...	A1
3	airport	аеропорт	A1
4	street	вулиця	A1
5	office	офіс	A1
6	camera	фотока...	A1
7	laptop	ноутбук	A1
8	diary	щоденник	A1
9	magazine	журнал	A1
10	hairbrush	щітка д...	A1
11	watch	годинни...	A1
12	newspap...	газета	A1
13	toothbru...	зубна щі...	A1
14	sweater	светр	A1
15	ticket	білет	A1
16	sunglass...	окуляри...	A1
17	money e...	обмін ва...	A1

Рисунок 2.2 – Інтерфейс таблиці words

На цій таблиці містяться різні поля та атрибути, такі як оригінальне слово, його переклад і рівень англійської мови,які є важливими складовими для реалізації наступного компонента,а саме гри “Шаради”.

Компонент REST виконує роль серверної частини, яка забезпечує обробку запитів від клієнта та взаємодію з базою даних та іншими компонентами системи. Він функціонує як посередник, приймаючи HTTP-запити з фронтенду, обробляючи їх та надсилаючи відповідні відповіді. REST-сервер також відповідає за виконання CRUD-операцій з базою даних, а також за інтеграцію з іншими сервісами, включаючи ігровий компонент.

Фронтенд є клієнтською частиною веб-застосунку, яка відповідає за інтерфейс користувача та взаємодію з сервером через REST-запити. Фронтенд надає користувачам можливість аутентифікації, реєстрації, проходження тестів та перегляду статистики. Процес аутентифікації включає два сценарії: для зареєстрованих користувачів, які вводять свій логін та пароль для входу, та для нових користувачів, які реєструються, вводячи логін та пароль двічі для

підтвердження. Після успішної аутентифікації користувач отримує доступ до персоналізованих функцій веб-застосунку.

Сторінка Облікового Запису є персональною сторінкою користувача, де зберігаються всі його дані. Ця сторінка інтегрована з фронтендом і відповідає за передачу даних користувача до бази даних для збереження та подальшої обробки. Вона забезпечує користувача інформацією про його активність, включаючи результати тестів та інші статистичні дані.

Компонент Тести дозволяє користувачам проходити різні тести, результати яких зберігаються в компоненті Статистика. Після завершення тесту результати надсилаються через REST-запит на Сторінку Облікового Запису, яка, в свою чергу, передає ці дані на сервер для збереження в базі даних. Таким чином, забезпечується централізоване зберігання та обробка результатів тестування.

Компонент Статистика відповідає за збереження та обробку результатів тестів користувачів. Цей компонент дозволяє аналізувати та візуалізувати дані, надаючи користувачам можливість переглядати свою успішність та прогрес. Інтеграція з базою даних забезпечує постійний доступ до історичних даних користувача.

Ігровий компонент є окремим модулем, що дозволяє користувачу грати в гру. Цей компонент взаємодіє з сервером для отримання необхідних даних, таких як вибір англійського слова з бази даних. Він функціонує автономно, проте використовує REST-запити для інтеграції з іншими частинами системи, забезпечуючи інтерактивність та динамічність ігрового процесу.

Таким чином, архітектура веб-застосунку складається з кількох взаємопов’язаних компонентів, кожен з яких виконує специфічні функції та забезпечує взаємодію між користувачем, сервером та базою даних. Кожен компонент має чітко визначені завдання та інтерфейси для інтеграції, що дозволяє створювати модульну та масштабовану систему [15].

Висновки до другого розділу:

Проведено детальний аналіз наявних інструментів та технологій для реалізації електронних інструментів та створення архітектури системи.

Виявлено, що вибір технології має відбуватися з урахуванням особливостей проекту та функціональних вимог.

Результати цього аналізу будуть використані для подальшої розробки електронних засобів та для визначення конкретних інструментів і технологій для кожної частини проекту.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Розробка веб-додатку

3.1.1 Розробка бази даних

Для реалізації бази даних, було обрано MySql та SQLAlchemy.

SQLAlchemy - це бібліотека для роботи з базами даних у Python, яка надає зручний і потужний спосіб взаємодії з реляційними базами даних. Вона дозволяє використовувати об'єктно-реляційне відображення (ORM), щоб взаємодіяти з базою даних через об'єкти Python, а також використовувати вирази SQL для складання запитів [16].

Через ORM створено клас User та інші, кожен з яких має свій функціонал(рисунок 3.1). До прикладу клас User,створений для зберігання логіну та пароля користувача, клас Words для збереження слів, для подальшого використання їх в тестах, клас TestCache для зберігання прогресу користувача.

```python
class User(db.Model):
    id: Mapped[int] = mapped_column(db.Integer, primary_key=True)
    username: Mapped[str] = mapped_column(db.String(80), unique=True, nullable=False)
    password_hash: Mapped[str] = mapped_column(db.String(128), nullable=False)


class Words(db.Model):
    id: Mapped[int] = mapped_column(db.Integer, primary_key=True)
    original: Mapped[str] = mapped_column(db.String(80),  nullable=False, unique=True)
    meaning: Mapped[str] = mapped_column(db.String(80), nullable=False)
    level: Mapped[EnglishLevel] = mapped_column(Enum(EnglishLevel), nullable=False)


12 usages
class TestCache(db.Model):
    id: Mapped[str] = mapped_column(primary_key=True)
    user_id: Mapped[int] = mapped_column(nullable=False)
    questions_count: Mapped[int] = mapped_column(nullable=False)
    responses_range: Mapped[str] = mapped_column(nullable=False)
    questions: Mapped[str] = mapped_column(default='')
    level: Mapped[EnglishLevel] = mapped_column(Enum(EnglishLevel), nullable=False)
    start_time: Mapped[datetime.datetime] = mapped_column(nullable=False)
    deadline: Mapped[datetime.datetime] = mapped_column(nullable=False)
    state: Mapped[TestState] = mapped_column(Enum(TestState), nullable=False)
```

Рисунок 3.1 – інтерфейс реалізації таблиць в базі даних

Завдяки цьому елементу коду, отримуєм готові таблиці.Таблиці містять логін та пароль ,який було захешовано.Також було використано Enum і ForeignKey.

Enum:

Enum використовується для створення перераховуваних типів даних у базі даних. Перераховуваний тип дозволяє вам обмежити значення поля до конкретного набору допустимих варіантів. Наприклад, якщо у вас є стовпчик для рівня англійських слів (наприклад, початковий, середній, високий), ви можете використати Enum для обмеження можливих значень цього стовпчика лише до цих трьох рівнів.

ForeignKey:

ForeignKey використовується для встановлення зв'язку між таблицями у базі даних. Коли ви визначаєте зовнішній ключ (foreign key) у моделі, ви вказуєте на стовпчик у поточній таблиці, який посилається на первинний ключ (primary key) іншої таблиці. Це створює зв'язок між цими двома таблицями, який можна використовувати для виконання операцій з'єднання та звернення до даних із зв'язаних таблиць.

Отже, Enum використовується для обмеження допустимих значень стовпців, а ForeignKey - для встановлення зв'язків між таблицями у базі даних. Вони обидва допомагають управляти структурою та обмеженнями даних у базі даних за допомогою SQLAlchemy.

3.1.2 Розробка frontend частини

Спочатку була розроблена фронтенд-частина, тобто генерація компонентів. Для прикладу розглянемо код, який створює нижню частину сайту(код наведено в додатку А). React-компонент, призначений для відображення нижнього колонтитула веб-сайту, використовуючи бібліотеку Material-UI для стилізації та позиціонування елементів.

Нижній колонтитул складається з двох частин, розділених сіткою. Перша частина містить інформацію про авторські права, яка вказує на те, що всі права захищені до 2024 року; друга частина містить політику конфіденційності, умови використання та посилання на соціальні мережі.

Нижній колонтитул також має темний фон (backgroundColour: '#201E28FF') і прилипає до нижньої частини екрана (position: 'sticky'). Це робить їх видимими і доступними при прокручуванні сторінки вниз.

Більшість сторінок у проекті створено за цим алгоритмом.

Наступним компонентом є фронтенд частина коду яка відповідає за авторизацію(рисунок 3.2).

```
8
9   const ActionTypes = {
10      LOGIN: 'LOGIN',
11      LOGOUT: 'LOGOUT'
12  }
13
14
15  const authReducer = (state, action) => {
16      switch (action.type) {
17          case ActionTypes.LOGIN:
18              localStorage.setItem(AuthItem, action.payload)
19              const tokenData = jwtDecode(action.payload)
20              return {
21                  ...state,
22                  user: {
23                      name: tokenData.sub,
24                      fresh: tokenData.fresh
25                  }
26              };
27          case ActionTypes.LOGOUT:
28              localStorage.removeItem(AuthItem)
29              return {
30                  ...state,
31                  user: null
32              };
33          default:
34              return state
35      }
36  }
```

Рисунок 3.2 – Інтерфейс авторизації

Цей алгоритм працює наступним чином:

1.Створення контексту користувача (UserContext):

UserContext створюється за допомогою React.createContext(null). Цей контекст використовується для зберігання інформації про поточного користувача та методів для управління його авторизацією.

2.Дії для зміни стану авторизації (ActionTypes та authReducer):

Об'єкт ActionTypes визначає різні дії, які можуть бути виконані, такі як увійти в систему або вийти з неї.

Функція authReducer приймає поточний стан та дію, і повертає новий стан відповідно до виконаної дії.

3.Контекстний постачальник (UserContextProvider):

Компонент UserContextProvider використовується для надання доступу до контексту користувача всім компонентам вашого додатку. Він використовує useReducer для управління станом авторизації, а також містить методи login та logout для управління процесом авторизації користувача.

Код також використовує useEffect для перевірки, чи є користувач авторизованим при завантаженні сторінки, і автоматично виконує вхід, якщо відповідний токен зберігається в локальному сховищі.

Цей код допомагає управляти станом авторизації користувача в додатку, зберігаючи та оновлюючи інформацію про авторизованого користувача та забезпечуючи методи для входу та виходу з системи. Це дозволяє іншим компонентам додатку отримувати доступ до цих даних та здійснювати відповідні дії, наприклад, відображати вміст, який доступний лише авторизованим користувачам.

3.1.3 Розробка backend частини

Спочатку розглянемо функцію, яка є парсером, що відповідає за оброблення слів із сайту з збереженням їх в базу даних(рисунок 3.3).

```python
def make_parser(english_level: EnglishLevel) -> typing.Callable:

    def wrapper() -> None:

        url = f'https://school.karpaty.info/materials/english/vocabulary/gen_{english_level}_vocabulary'
        response = requests.get(url)
        logging.info(f'Parsing url {url}')
        soup = bs4.BeautifulSoup(response.text, features: 'html.parser')
        tags = soup.select('ol > li')

        for tag in tags:

            try:
                logging.info(f'Adding word `{tag.text}` to db')
                original, translation = tag.text.split( sep: '-', maxsplit=1)
                db.session.add(
                    Words(
                        original=original,
                        meaning=translation,
                        level=english_level,
                    )
                )
                db.session.commit()

            except sqlalchemy.exc.IntegrityError:
                logging.warning(f"Word already exists - `{tag.text}`. Skipping")
                db.session.rollback()

    wrapper.__name__ = wrapper.__qualname__ = uuid.uuid4().hex
```

Рисунок 3.3 – Інтерфейс парсеру

Цей Python-скрипт призначений для аналізу інформації з веб-сайту, що містить словник англійської мови, і збереження даних у базі даних.

Основні етапи виконання скрипта

Імпорт бібліотек і модулів:

- Використовуються різні бібліотеки, в тому числі logging для ведення логів, bs4 для розбору HTML, запити для взаємодії з сайтом, а також модулі з власного додатку для роботи з Flask і SQLAlchemy.

- Створіть контекст додатку та створіть таблиці в базі даних:

- Забезпечує правильну взаємодію з додатком Flask і створює таблиці в базі даних.

Основні функції синтаксичного аналізатора:

- Створює специфічні англомовні функції синтаксичного аналізатора, розбирає дані і зберігає їх у базі даних.

Синтаксичний аналіз і зберігання даних.

- Аналізує HTML-сторінки сайту, знаходить слова та їхні переклади і зберігає цю інформацію в базі даних. Слова, які вже є в базі даних, пропускаються.

Налаштування системи логування:

- Встановлення параметрів системи журналювання, таких як рівень журналювання, формат і дата.

Вийти з контексту програми:

- Закриває контекст програми після того, як дані було проаналізовано і збережено у базі даних.

Цей скрипт можна використовувати для автоматизації поповнення бази даних англійськими словами заданого рівня володіння мовою.

Наступний фрагмент є кодом для генерації самого тесту (код наведено в Додатку Б).

TestManager взаємодіє з базою даних, моделлю користувача та моделлю тесту. Основний алгоритм роботи можна описати наступним чином

Створення тесту:

- Перевірити, чи є у користувача вже активний тест.

- Створити новий тест з випадковою кількістю запитань і відповідей.

- Встановити час початку, час закінчення та інші параметри.

Отримання тесту користувача:

- Отримує активний тест для вказаного користувача.

- Повертає None, якщо тесту немає.

Отримати всіх тестів користувача:

- Отримує всі тести для вказаного користувача у зворотному порядку за часом запуску.

- Повертає список тестів (до останніх 40).

- Повертає наступне питання користувача:

- Повертає активні вікторини користувача: повертає список активних вікторин користувача у зворотному порядку за часом початку.

- Повертає наступне питання або Ні, якщо тест завершено.

Відповісти на останнє запитання:

- Знаходить активний тест і встановлює відповідь на останнє запитання.

Завершити тест:

- Знаходить активний тест.

- Встановити статус тесту (СКАСОВАНО або ЗАВЕРШЕНО).

Основна ідея полягає в тому, щоб дати користувачеві можливість пройти тест, відповісти на нього і отримати результат. Клас дозволяє створювати, відстежувати і завершувати тести для кожного користувача окремо.

Також є головний файл бекенду, саме він відповідає за всі запити і взаємодію між різними елементами програми(рисунок 3.4)

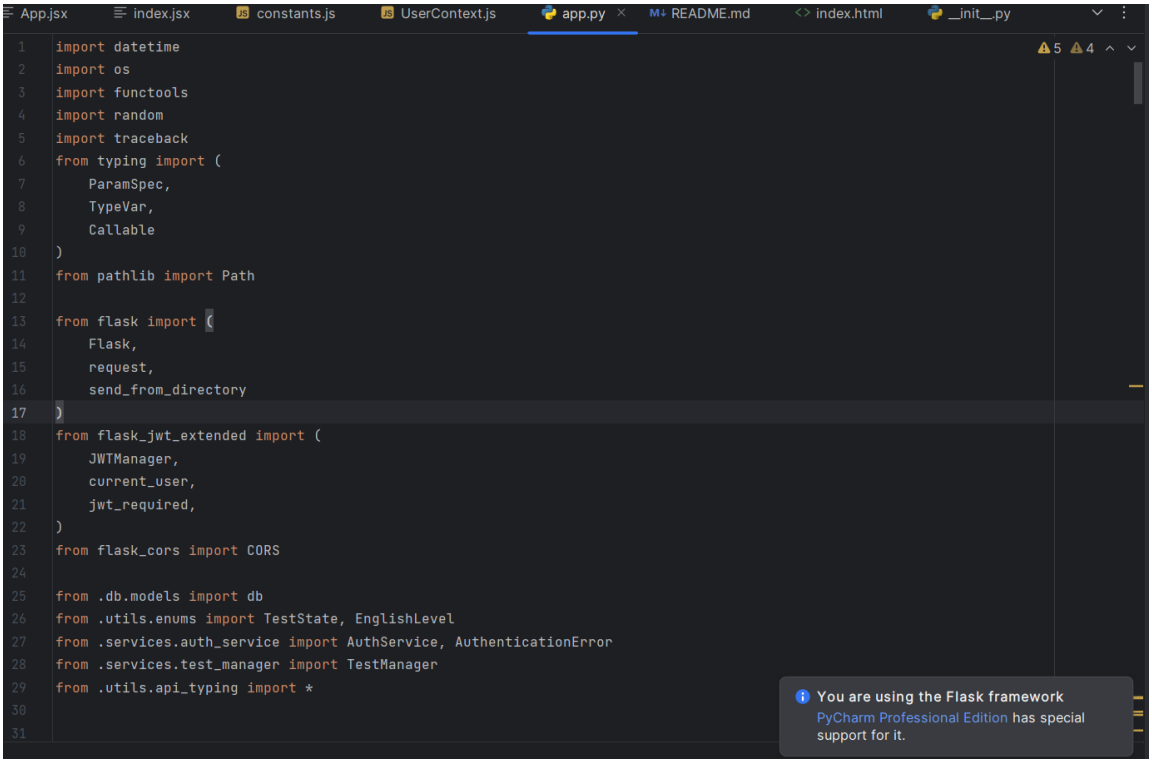


Рисунок 3.4– інтерфейс головного файлу

Алгоритм роботи:

Імпорти модулів та класів:

У цьому розділі імпортуються необхідні залежності для побудови веб-додатку. Це включає модулі Flask, SQLAlchemy для роботи з базою даних, Flask-JWT-Extended для автентифікації та авторизації, а також різні інші модулі, які використовуються в додатку.

Конфігурація додатку Flask:

Налаштовується додаток Flask, вказуючи параметри підключення до бази даних, секретний ключ для підпису JWT-токенів та інші конфігураційні параметри. Також створюєте необхідні директорії для зберігання логів.

Декоратори та функції маршрутизації:

У цьому розділі визначаються функції, які викликаються при зверненні до певних URL-адрес. Кожна функція обробляє HTTP-запити та відповідає на них відповідним чином. Декоратори використовуються для виконання певних дій перед або після виклику функції, таких як перевірка авторизації або обробка помилок.

Функції та декоратори:

У цьому розділі визначені корисні функції та декоратори, які використовуються в інших частинах додатку. Наприклад, декоратори catch_missing_field, catch_auth_error або catch_unexpected використовуються для обробки різних видів помилок та надання адекватних відповідей.

Структура проекту:

Крім коду, зазначеного вище, у проекті є ще багато інших компонентів, таких як моделі бази даних, сервіси (наприклад, AuthService, TestManager) для роботи з цими моделями, а також фронтенд, який розміщений у папці frontend.

Логування та обробка помилок:

Використовується логування, щоб фіксувати помилки та непередбачені ситуації, які можуть виникнути в процесі виконання додатку. Логи розділені на різні файли залежно від контексту (наприклад, authentication.log для автентифікації, tests.log для тестів тощо), що допомагає спростити їх подальшу обробку.

Взаємодія з базою даних:

Використовується SQLAlchemy для взаємодії з базою даних SQLite, створюєте та оновлюєте таблиці за допомогою моделей, які ви визначаєте, а також виконуєте різні операції з даними, такі як створення нових користувачів, входження в систему, відповіді на питання тестів тощо.

Отже, веб-додаток побудований з урахуванням кращих практик розробки веб-додатків з використанням Flask та SQLAlchemy. Він має структурований та модульний код, який полегшує розширення та підтримку в майбутньому.

3.1.4 Розробка гри

Гру реалізовано через canvas. До прикладу наступний фрагмент коду(рисунок 3.5),відповідає за генерацію ящика, в якій міститься саме слово.

```
let bbox = ThePlayer.boundingBox
let playerX = bbox.centerX
let playerY = bbox.centerY
let centerX = this.x * 8 + TILE_SIZE / 2
let centerY = this.y * 8 + TILE_SIZE / 2

if (distanceSquared(centerX, centerY, playerX, playerY) < 140) {
    this.needRender = false
    console.log('Chest hit !')
    fetch('http://localhost:5000/random_word')
        .then(res => {
            return res.json()
        })
        .then(async json => {
            CurrentWords.push(json['data'])
            this.currentText = `${json['data']['word'].toUpperCase()} - ${json['data']['meaning'].toUpperCase()}`
        })
}
}

render() {

    if (!this.needRender) {

        if (this.currentText !== null) {
            const text = TheGraphics.measureText(this.currentText)
            const oldStyle = TheGraphics.fillStyle
            TheGraphics.fillStyle = 'gray'
            TheGraphics.fillRect(this.x * 8 - 2, this.y * 8 - 11, text.width + 2, 15)
            TheGraphics.fillStyle = oldStyle
            TheGraphics.fillText(this.currentText, this.x * 8, this.y * 8)
```

Рисунок 3.5 -Інтерфейс алгоритму створення ящика

Основні елементи цього коду:

Імпорти:

В цьому розділі імпортуються зображення ящику, клас Tile, а також різні глобальні змінні, константи та допоміжні функції, що використовуються в коді.

Конструктор:

Визначено конструктор класу Chest, який ініціалізує об'єкт ящику з вказаною позицією x та y. Встановлюються початкові значення для властивостей

needRender (логічне значення, що вказує, чи потрібно рендерити ящик) та currentText (текст, що відображається поблизу ящику).

Метод step():

Цей метод викликається на кожному кроці анімації гри. Він перевіряє, чи гравець знаходиться в непосредній близькості до ящика (за допомогою функції distanceSquared для обчислення відстані між гравцем та ящиком). Якщо гравець наблизився до ящика, то відбувається запит до сервера за випадковим словом, яке додається до глобального масиву CurrentWords, а також встановлюється значення currentText, щоб відображати текст поруч із ящиком.

Метод render():

Цей метод відповідає за відображення ящика на екрані. Якщо ящик не потрібно рендерити (значення needRender дорівнює false), то виводиться текст поруч із ящиком (якщо він був встановлений). В іншому випадку рендериться зображення ящика з використанням графічного двигуна.

Цей клас відповідає за логіку та візуалізацію ящиквів у грі, а також за взаємодію з ними, коли гравець знаходиться поруч.

Настуним чудовим прикладом є фрагмент коду,який відповідає за фінал рівня(код наведений в додатку В).

Основні елементи цього коду:

Імпорти:

Цей розділ містить імпорти констант, глобальних змінних, а також різних утилітних функцій та класів, що використовуються у коді.

Змінні:

Визначені змінні index та colors для випадкового вибору кольору для кожної точки. Колір кожної точки визначається як випадковий елемент з масиву кольорів.

Клас Point представляє точку, яка рухається навколо мети. Він має властивості goalX та goalY, що визначають мету точки, а також координати та швидкість переміщення по осях x та y. Також використовуються випадкові значення для визначення початкової позиції та швидкості руху кожної точки.

Константи NUM_POINTS та GOAL_RADIUS визначають кількість точок, що оточують мету, та радіус області, яка визначає, коли гравець досяг мети.

Метод step() класу Point (код наведений в додатку В):

Цей метод викликається на кожному кроці анімації гри. Він оновлює координати кожної точки та відтягує їх до центру мети за допомогою гравітації. Якщо рівень вже завершено (ThePlayer.finishedLevel дорівнює true), то змінна gravity зменшується, що веде до того, що точки відстороняються від мети.

Метод step() класу Goal (код наведений в додатку В):

Цей метод також викликається на кожному кроці анімації гри. Він перевіряє, чи гравець наблизився до мети, якщо так, то встановлює змінну ThePlayer.finishedLevel в true, щоб позначити, що рівень було завершено. Якщо рівень вже завершено, то зменшується прозорість мети.

Метод render() класу Goal (код наведений в додатку В):

Цей метод відповідає за відображення мети та точок, які навколо неї рухаються. Кожна точка рухається навколо мети та має випадковий колір. Якщо рівень вже завершено, то точки зникають плавно, оскільки їх прозорість зменшується до нуля.

Цей клас відповідає за візуалізацію мети та рухомих точок навколо неї у грі, а також за перевірку, чи гравець досяг мети для завершення рівня.

3.2 Тестування веб-додатку

Система гейміфікації, запропонована для підтримки вивчення шкільних курсів англійської мови, розширює навчальну програму за допомогою гейміфікації та інтерактивних елементів. У розробленому додатку перед користувачем ставиться завдання успішно пройти певну кількість рівнів, де він має вирішити практичні завдання, пов'язані з конкретною темою курсу англійської мови. Також користувач може пограти в цікаву гру-платформер

Спочатку додаток вимагає входу за допомогою логіну та паролю (рис. 3.6).

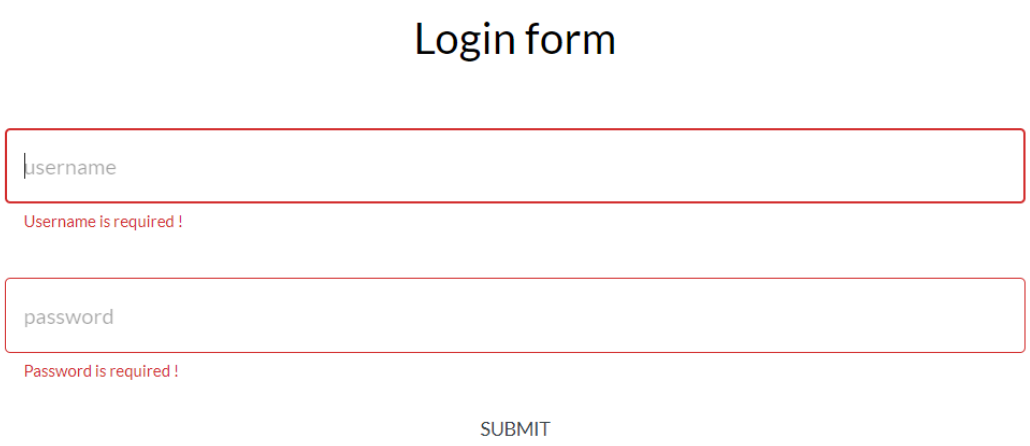


Рисунок 3.6 - Інтерфейс форми реєстрації

Після введення імені користувача та пароля з'являється дошка повідомлень (рис. 3.7). Після цього учні можуть перейти на вкладку "Тести" (Рис. 3.8) і почати розв'язувати завдання.

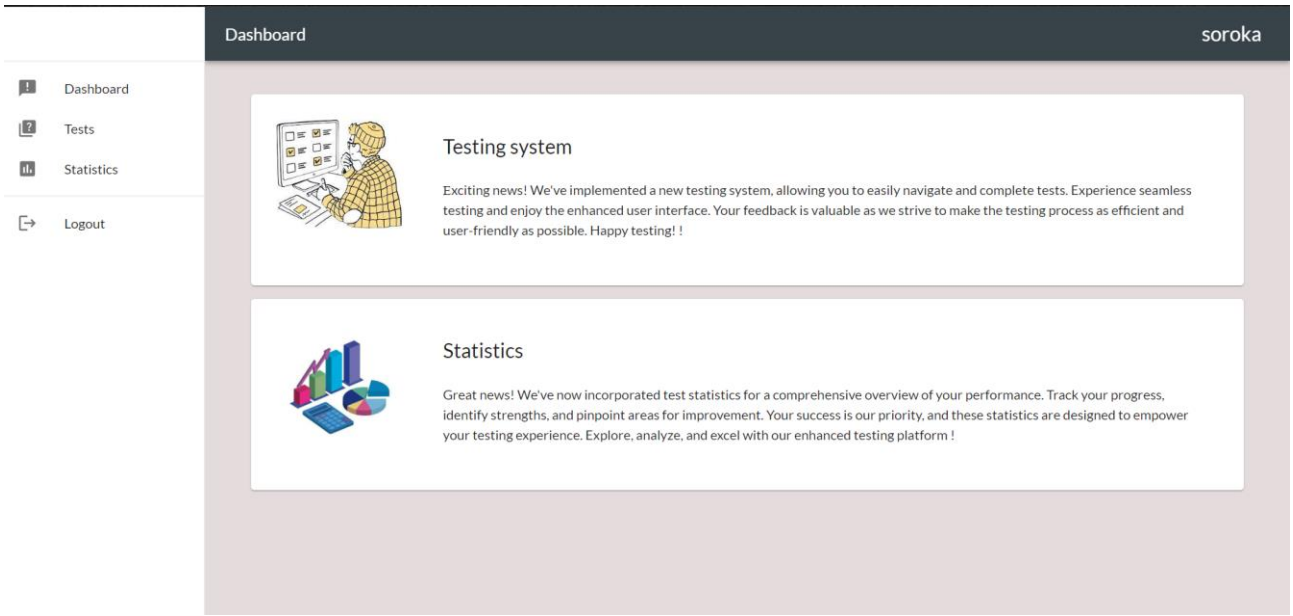


Рисунок 3.7 -Інтерфейс веб-додатку

Наведений у якості прикладу зображення показує загальні тези та новини

Суть завдання полягає в тому, що є кілька рівнів (це рівні англійської мови A1, A2, B1 і B2). З самого початку три з них закриті. Це означає, що рівні потрібно проходити по порядку, починаючи з A1. Сам алгоритм гри показує одне

слово англійською мовою і чотири варіанти відповіді (та їх переклад), і користувач повинен вибрати правильний.

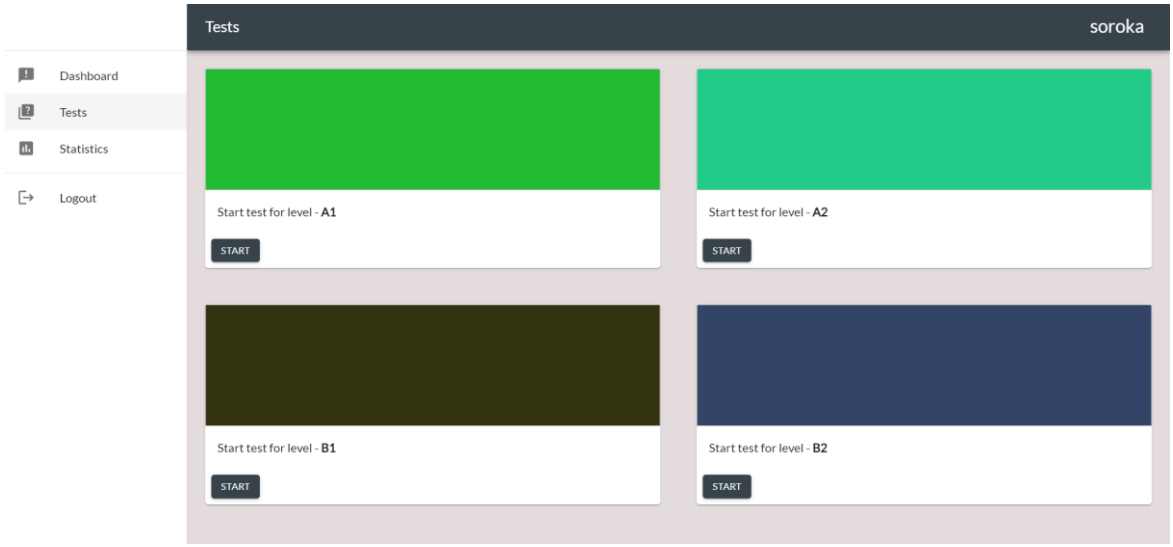


Рисунок 3.8 - Інтерфейс вкладки Tests

В іншому випадку кількість запитань і самі запитання були б різними кожного разу, коли користувач проходив би тест. Щоб пройти тест, користувач повинен отримати 80% правильних відповідей. Після завершення тесту, користувач одразу бачить свій результат.

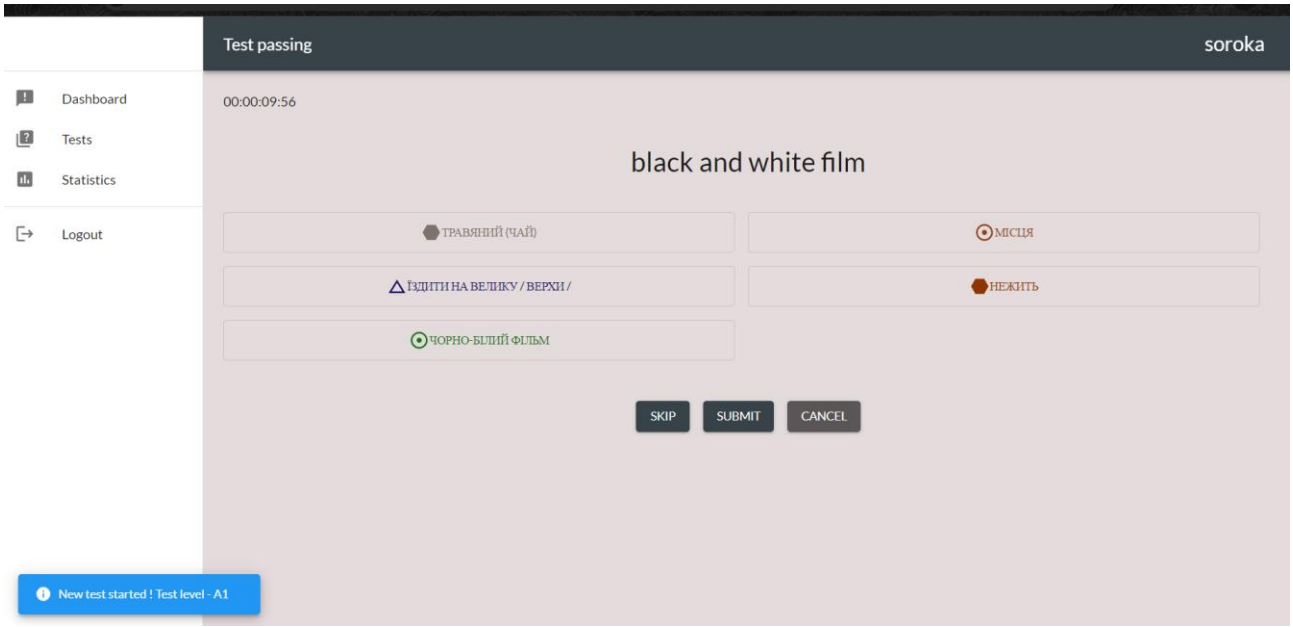


Рисунок 3.9 - Інтерфейс самого завдання

Ви також можете побачити додаткові функції у вигляді таймерів і поставити тест на паузу. Ви також можете пропустити самі запитання.

Крім того, є третя вкладка "Статистика" (рисунок - 3.10), яка разом зі збором та аналізом статистичних даних (рисунок - 3.11) дозволяє систематично відслідковувати академічний прогрес учнів.

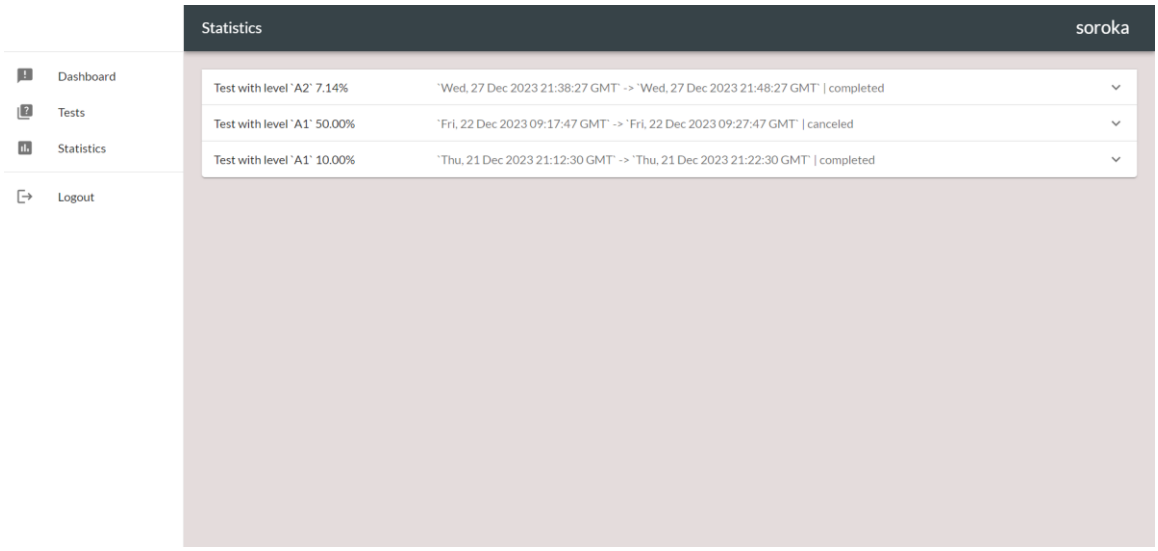


Рисунок 3.10 -Інтерфейс вкладки Tests

На ілюстрації можна побачити кількість тестів, процент правильних відповідей і коли вони були пройдені.

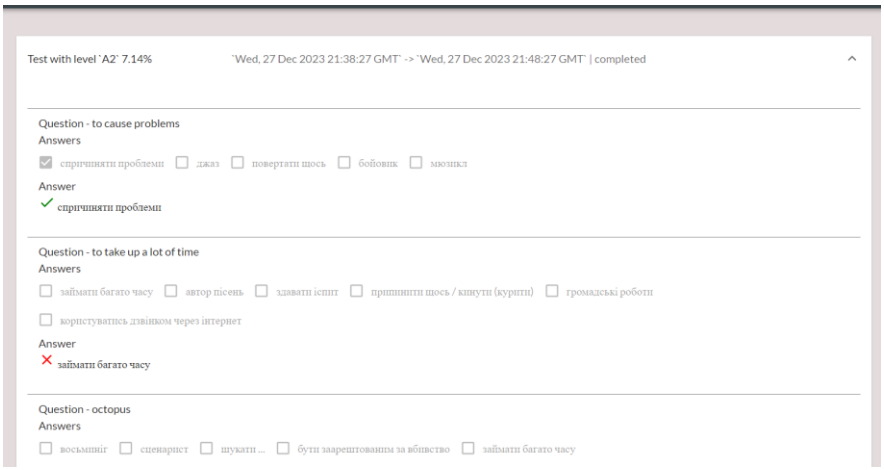


Рисунок 3.11 -Інтерфейс самої статистики

Для кожного питання можна побачити, які відповіді були обрані і які з них були правильними.

Наступним етапом, з яким може взаємодіяти користувач є гра-платформер,в якій гравцю потрібно проходити рівень за фентезійного героя, оминаючи перешкоди(рисунок – 3.12). Під час проходження рівня головному

герою потрібно збирати скрині,дотик до яких відкриває слова,а точніше слово на англійські мові та його переклад.Також в героя є цікава функція,яка дозволяє йому включати “крила”,за допомогою яких він здатен робити вищі стрибки.

Рисунок 3.12 – Інтерфейс гри

На ілюстрації зображено головного героя, перешкоду, скриню і саме слово з перекладом.

Ці слова потрібні для того щоб пройти рівень ,а саме в фінальній точці, при дотику, з’являється вікно в якому потрібно ввести переклади 3 слів, які були в скринях(рисунок 3.13). Самі слова обираються випадково, адже під час проходження рівня скринь є набагато більше.

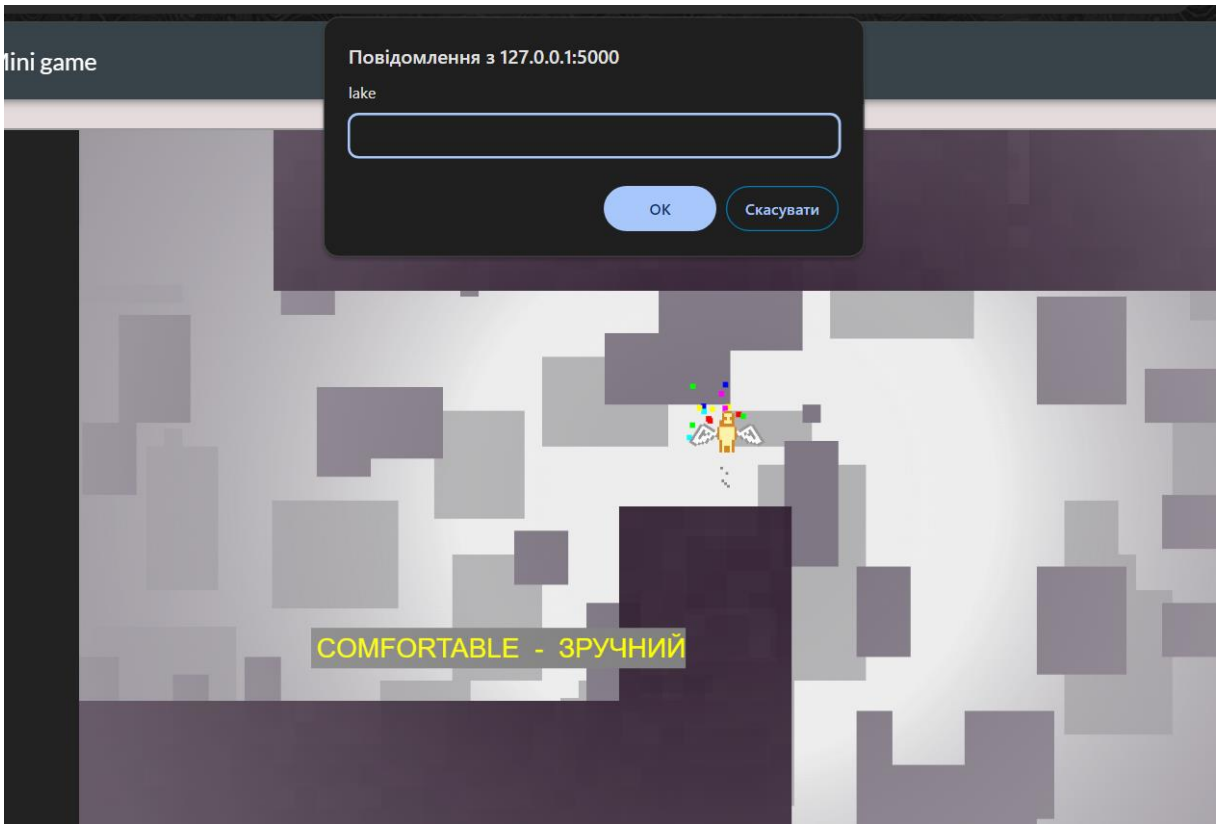


Рисунок 3.13 – Інтерфейс фіналу рівня

На ньому є вікно в якому потрібно ввести переклад слова, саме слово показане вище строки вводу.

Самі рівні створюються в ручному режимі, за допомогою спеціального вікна, після чого копіюється html-код сторінки і вноситься в спеціальний json файл.

Висновки до третього розділу:

Розділ 3 охоплює створення та розробку веб-додатку, включаючи етап тестування. При розробці використовувалися такі технології, як React для фронтенду і Flask для бекенду, а також база даних MySQL.

Етап тестування дозволив перевірити функціональність додатку, виявити можливі помилки та покращення. Результати цього етапу підтвердили, що розроблений веб-додаток відповідає початковим вимогам і готовий до подальшого використання в навчальному процесі.

ВИСНОВКИ

Мета дослідження, яка полягала в розробці гейміфікованої системи для вивчення англійської мови у формі веб-сервісу, спрямованого на покращення якості та ефективності навчання, була успішно досягнута. В ході роботи було створено інтерактивний веб-сервіс, який поєднує гейміфікацію з мовними матеріалами, що дозволяє підвищити інтерес учнів до навчання та сприяє ефективному засвоєнню знань. Система містить різноманітні завдання та ігрові елементи, що стимулюють взаємодію з навчальним контентом, покращують мотивацію та забезпечують більш глибоке розуміння мовних процесів. Дослідження показало, що використання гейміфікації в навчальному процесі є ефективним підходом для досягнення вищих результатів у вивченні англійської мови.

Рекомендації щодо застосування: Основною цільовою аудиторією для використання даної системи є діти віком від 7 до 15 років. Ця вікова категорія особливо сприятлива до ігрових елементів в навчанні, що допомагає підтримувати високу мотивацію та інтерес до вивчення англійської мови.

Подальший розвиток проекту: Рекомендується постійно оновлювати та розширювати мовні матеріали, додаючи нові завдання, ігри та інтерактивні елементи. Це допоможе підтримувати інтерес учнів та забезпечити різноманітність навчального процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методична розробка "Гейміфікація в сучасному навчально-освітньому просторі" [електронний ресурс] -https://naurok.com.ua/metodichna-rozrobka-geymifikaciya-v-suchasnomu-navchalno-osvitnomu-prostori-135509.html. [дата доступу 03.02.2024]
2. ВАЖЛИВІСТЬ ВИВЧЕННЯ ІНОЗЕМНИХ МОВ У СУЧАСНОМУ СВІТУ [електронний ресурс] - https://essuir.sumdu.edu.ua [дата доступу 08.02.2024]
3. Корисність інтерактивних методів навчання [електронний ресурс] - https://osvita.ua/school/method/technol/6564/ [дата доступу 08.02.2024]
4. Які переваги та недоліки гейміфікації в освіті [електронни ресурс] – https://life.obozrevatel.com/ [дата доступу 11.03.2024]
5. LearnEnglish Kids [електронний ресурс] - https://learnenglishkids.britishcouncil.org/ [дата доступу 07.02.2024]
6. FunBrain [електронний ресурс] - https://www.funbrain.com/ [дата доступу 07.03.2024]

7.WordClouds [електронний ресурс] - https://www.wordclouds.com [дата доступу 07.03.2024]

8. You don’t Know JS,Kyle Simpson,2015

9.Beginning React and Firebase 1st ED,Nabendu Biswas,2022

10. Офіційна документація React [електронний ресурс] - https://uk.legacy.reactjs.org/docs/getting-started.html [дата доступу 02.01.2024]

11.Fluent Python: Clear, Concise, and Effective Programming,Luciano Ramalho,2015.

12.Flask Web Development: Developing Web Applications with Python 2nd Edition, Miguel Grinberg,2018

13. Canvas API[електронний ресурс] - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API [дата доступу 06.03.2024]

14. Software Architecture in Practice (SEI Series in Software Engineering) 3rd Edition,Len Bass,Paul Clements,2012.

15. Lehinovych A., Soroka R., Nevmerzhytskyi V., Semianyk M.-I., Izmailov A. Gamified Web Assistant for Support of Learning Process in Secondary School – Zgamifikowany Asystent Webowy do Wspierania Procesu Uczenia w Szkole Średniej. Zeszyty Studenckiego Towarzystwa Naukowego AGH. Wydawnictwo Studenckiego Towarzystwa Naukowego AGH – ISSN 1732-0925; 2024. In print.

16.SQLAlchemy 2 In Practice: Learn to program relational databases in Python step by step,Miguel Grinberg,2023.

ДОДАТОК А.КОД РЕАЛІЗЦІЇ ОДНОГО З REACT КОМПОНЕНТІВ

```
import React from "react";
import {
    Box,
    Grid,
    Typography
} from '@mui/material'



export default function Footer() {
    return (
        <Box sx={{
            backgroundColor: '#201E28FF',
            mt: 8,
            p: 3,
            position: 'sticky',
        }} id={'footer'}>
            <Grid container spacing={2}>
                <Grid item xs={12} md={6} >
                    <Typography variant={'h6'} component={'p'} color={'white'}>
                        {'© 2023 Something Inc. All rights protected'}
                    </Typography>
```

```
            </Grid>
            <Grid item xs={12} md={6} sx={{textAlign: 'center'}}>
              <Typography variant={'h6'} component={'p'} color={'gray'}>
                Privacy policy
              </Typography>
              <Typography variant={'h6'} component={'p'} color={'gray'}>
                Terms of use
              </Typography>
              <Typography variant={'h6'} component={'p'} color={'gray'}>
                Our social
              </Typography>
            </Grid>
          </Grid>
        </Box>
    )
}
```

ДОДАТОК Б.РЕАЛІЗАЦІЯ КЛАСУ ГРИ,А САМЕ ГЕНЕРАЦІЯ “ЯЩИКА”

```
import { TILE_SIZE } from '../constants'

import { ThePlayer, deltaTime } from '../globals'

import { TheGraphics } from '../Graphics'

import { TheRenderer } from '../Renderer'

import { approach, distanceSquared } from '../utils'

import { GridEntity } from './GridEntity'

let index = 0

let colors = ['#f00', '#fff', '#0f0', '#00f', '#0ff', '#ff0', '#f0f']


class Point {

  constructor (goalX, goalY) {

    this.goalX = goalX

    this.goalY = goalY

    let r = 5 + Math.random() * 10

    let rSpeed = 10

    let a = Math.random() * 2 * Math.PI

    this.x = this.goalX + Math.cos(a) * r

    this.y = this.goalY + Math.sin(a) * r

    this.xSpeed = Math.sin(a) * rSpeed

    this.ySpeed = -Math.cos(a) * rSpeed

    this.gravity = 500
```

```
    this.color = colors[index++ % colors.length]
  }
  step () {
    this.x += this.xSpeed * deltaTime
    this.y += this.ySpeed * deltaTime
    this.gravitateTowards(this.gravity, this.goalX, this.goalY)
    if (ThePlayer.finishedLevel) {
      this.gravity = approach(this.gravity, 100, 100)
    }
  }
  gravitateTowards (gravity, x, y) {
    let dx = this.x - x
    let dy = this.y - y
    let rr = Math.max(1, dx * dx + dy * dy)
    let ax = -gravity * dx / rr
    let ay = -gravity * dy / rr

    this.xSpeed += ax * deltaTime
    this.ySpeed += ay * deltaTime
  }
}
const NUM_POINTS = 20
const GOAL_RADIUS = TILE_SIZE * 2
```

```
export class Goal extends GridEntity {
  constructor (x, y) {
    super(x, y)
    this.points = []
    for (let i = 0; i < NUM_POINTS; i++) {
      this.points.push(new Point(
        this.x + TILE_SIZE / 2,
        this.y + TILE_SIZE / 2
      ))
    }
    this.alpha = 1
  }
  step () {
    let bbox = ThePlayer.boundingBox
    let playerX = bbox.centerX
    let playerY = bbox.centerY
    let centerX = this.x + TILE_SIZE / 2
    let centerY = this.y + TILE_SIZE / 2
    if (distanceSquared(centerX, centerY, playerX, playerY) < GOAL_RADIUS * GOAL_RADIUS && (ThePlayer.isDashing || ThePlayer.ySpeed >= 260)) {
      ThePlayer.setFinished()
    }
```

```
  if (ThePlayer.finishedLevel) {
    this.alpha = approach(this.alpha, 0, deltaTime)
  }
}
render () {
  TheRenderer.setAlpha(this.alpha)
  for (let i = 0; i < NUM_POINTS; i++) {
    let point = this.points[i]
    let skip = Math.round(Math.random() * Math.random() * 8)
    for (let j = 0; j < skip; j++) {
      point.step()
    }
    TheRenderer.drawRectangle(
      point.color,
      point.x - 1,
      point.y - 1,
      2,
      2
    )
  }
  TheRenderer.resetAlpha()
}
```

ДОДАТОК В.РЕАЛІЗАЦІЯ АЛГОРИТМУ ПРОХОДЖЕННЯ РІВНЯ В ГРІ

```
import { TILE_SIZE } from '../constants'
import { ThePlayer, deltaTime } from '../globals'
import { TheGraphics } from '../Graphics'
import { TheRenderer } from '../Renderer'
import { approach, distanceSquared } from '../utils'
import { GridEntity } from './GridEntity'



let index = 0
let colors = ['#f00', '#fff', '#0f0', '#00f', '#0ff', '#ff0', '#f0f']


class Point {
  constructor (goalX, goalY) {
    this.goalX = goalX
    this.goalY = goalY


    let r = 5 + Math.random() * 10
    let rSpeed = 10
    let a = Math.random() * 2 * Math.PI
    this.x = this.goalX + Math.cos(a) * r
    this.y = this.goalY + Math.sin(a) * r
    this.xSpeed = Math.sin(a) * rSpeed
```

```
    this.ySpeed = -Math.cos(a) * rSpeed
    this.gravity = 500
    this.color = colors[index++ % colors.length]
  }
  step () {
    this.x += this.xSpeed * deltaTime
    this.y += this.ySpeed * deltaTime
    this.gravitateTowards(this.gravity, this.goalX, this.goalY)
    if (ThePlayer.finishedLevel) {
      this.gravity = approach(this.gravity, 100, 100)
    }
  }
  gravitateTowards (gravity, x, y) {
    let dx = this.x - x
    let dy = this.y - y
    let rr = Math.max(1, dx * dx + dy * dy)
    let ax = -gravity * dx / rr
    let ay = -gravity * dy / rr
    this.xSpeed += ax * deltaTime
    this.ySpeed += ay * deltaTime
  }
}
```

```
const NUM_POINTS = 20
const GOAL_RADIUS = TILE_SIZE * 2
export class Goal extends GridEntity {
  constructor (x, y) {
    super(x, y)
    this.points = []
    for (let i = 0; i < NUM_POINTS; i++) {
      this.points.push(new Point(
        this.x + TILE_SIZE / 2,
        this.y + TILE_SIZE / 2
      ))
    }
    this.alpha = 1
  }
  step () {
    let bbox = ThePlayer.boundingBox
    let playerX = bbox.centerX
    let playerY = bbox.centerY
    let centerX = this.x + TILE_SIZE / 2
    let centerY = this.y + TILE_SIZE / 2

    if (distanceSquared(centerX, centerY, playerX, playerY) < GOAL_RADIUS *
GOAL_RADIUS && (ThePlayer.isDashing || ThePlayer.ySpeed >= 260)) {
```

```
    ThePlayer.setFinished()
  }

  if (ThePlayer.finishedLevel) {
    this.alpha = approach(this.alpha, 0, deltaTime)
  }
}
render () {
  TheRenderer.setAlpha(this.alpha)
  for (let i = 0; i < NUM_POINTS; i++) {
    let point = this.points[i]
    let skip = Math.round(Math.random() * Math.random() * 8)
    for (let j = 0; j < skip; j++) {
      point.step()
    }
    TheRenderer.drawRectangle(
      point.color,
      point.x - 1,
      point.y - 1,
      2,
      2
    )
  }
```

```
    TheRenderer.resetAlpha()
  }
}
```