Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра мультимедійних систем

Розробка web - застосунку для навчання іноземних мов за допомогою карток

Текстова частина до курсової роботи за
спеціальністю «Комп’ютерні науки»

Керівник курсової роботи:
Калітовський Б. В.

Виконала студентка:
Петренко Є. Г.

Київ 2023

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

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Жежерун О.П.

_____

(підпис) “____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Петренко Єлизавета Григорівни факультету інформатики 3 курсу

Тема: Розробка web - застосунку для навчання іноземних мов за допомогою карток.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Аналіз предметної області. Постановка завдання курсової роботи

2. Теоретичні відомості

3. Опис реалізації додатку

4. Висновки

Перелік використаних джерел

Дата видачі “___” _____ 2023 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	14.09.2022	
2.	Огляд технічної літератури за темою роботи.	15.12.2022	
3.	Аналіз актуальності та дослідження аналогів.	10.01.2023	
4.	Написання практичної частини.	26.02.2023	
5.	Написання пояснювальної роботи.	05.05.2023	
5.	Попередня демонстрація роботи науковому керівнику	09.05.2023	
7.	Створення презентації	14.05.2023	
8.	Захист курсової роботи		

Студент _____

Керівник _____

“_____” _____

Зміст

Перелік використаних скорочень та термінів

Метод Flash – метод запам’ятовування слів, у якому при неправильній відповіді алгоритм методу повертає «неправильну» карточку до користувача ще раз через деякий час після проходження інших карток.

Метод Contextual Relearning – метод запам’ятовування слів, який полягає у тому, щоб при вивченні нового слова одразу дивитись на його значення та використання у реченні або іншому фразовому прикладі.

Метод асоціацій – метод запам’ятовування слів, який створює асоціацію з навчальним словом, за допомогою чому слово запам’ятовується набагато краще та швидше.

DOM (Document Object Model) – представлення структури HTML-документа у вигляді дерева об’єктів. DOM надає програмному забезпеченню доступ до цього дерева та можливість взаємодії з його елементами.

NoSQL (“Not only SQL”) – відноситься до нових типів баз даних, які відрізняються від традиційних SQL. NoSQL працюють з нереляційними даними, які зберігаються в різних форматах, таких як JSON, XML тощо.

SDK (Software Development Kit) – набір інструментів, бібліотек, документацій та прикладів, які допомагають розробникам створювати програмне забезпечення для певної платформи або сервісу. Він забезпечує набір інструментів, які спрощують розробку, тестування та випуск ПЗ, надаючи необхідні компоненти та інтерфейси.

Анотація

У даній курсовій роботі передбачається розробка веб-застосунку для вивчення іноземних мов за допомогою карток. Веб-застосунок використовує декілька методів для запам’ятовування нових слів. Основні з них – метод Flash, метод Contextual Relearning та метод асоціацій. Створений застосунок містить увесь реалізований функціонал, вказаний у вимогах. У текстовій частині роботи міститься опис предметної області, опис роботи методів навчання та детальний опис процесу розробки веб-застосунку.

ВСТУП

На сьогоднішній день вивчення мов вважається нормою для людини, яка хоче розвиватися, навчатися чомусь новому, розширювати свої можливості, збагачувати знання та отримувати досвід. Існують різні способи вивчення нових мов, але одне з найпоширеніших та менш ресурсозатратних вважаються веб-застосунки. Саме вони вважаються дуже актуальними та корисними у використанні, бо не займають багато часу, грошей та зусиль. До того ж, і людей, які хочуть швидко отримувати інформацію зараз все більше і більше.

Для того, щоб процес навчання тривав цікавіше застосунки робляться у різних форматах, використовуючи різний функціонал, дизайн та методи навчання.

Одним з таких прикладів можуть стати веб-застосунки, які використовують картки для запам’ятовування слів. Якщо розглядати саме такий формат навчання, то тут також можна використовувати різні підходи методи та форми.

Метою курсової роботи є створення сучасного, простого у застосуванні середовища для вивчення іноземних мов за допомогою карток.

Завданням – реалізувати веб-застосунок, який буде за допомогою карток та декількох методів надавати можливість студенту запам’ятовувати слова, визначені цим самим студентом.

Об’єктом дослідження стало створення застосунку, який буде надавати відповідний функціонал для того, щоб користувач міг вивчати нові слова за допомогою запам’ятовування перекладу слів.

Предмет дослідження роботи представляє собою методи якісного запам’ятовування слів використовуючи картки.

Текстова частина курсової роботи містить у собі 3 розділи.

Перший з них проводить аналіз сучасного стану питання та обґрунтування теми, також містить у собі огляд аналогів та визначає постановку завдання . Другий аналізує та наводить опис засобів та інструментів розробки веб-застосунків. Третій розділ вже описує реалізацію самого програмного продукту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.

1.1 Аналіз сучасного стану питання та обґрунтування теми

Взагалі, питання вивчення іноземних мов є досить актуальним за останні десятиріччя, особливо за останні декілька років, коли люди почали багато подорожувати. Також, якщо говорити про розвиток особистих навичок та вмінь людини, то суспільство призначає велику роль саме мовам.

Знання іноземних мов дає людям купу можливостей у багатьох напрямках. Основні з них – це подорожі, підвищення по кар'єрних сходах, заробіток на новому рівні, розвиток себе та своїх навичок і, звичайно, поліпшення самої комунікації між людьми. Навчальні школи та курси пропонують багато можливостей та методів по вивченню та запам'ятовуванню слів. Їх вважаються якісною платформою для стрибку у знання та навичках. Також, вони забезпечують багато плюсів. Головні – це:

- спілкування з людьми мовою, яка вивчається
- професійні викладачі
- певне знаходження у мовному просторі
- поглибленні знання у різних темах

Але, на жаль, не всі мають можливість та ресурси навчатися на курсах. Багато хто з людей працюють до самого вечора та просто не мають часу, проте є бажання вивчити хоча б базу іншої іноземної мови. Тому, існують деякі альтернативні шляхи вирішення питання щодо запам'ятовування нових слів. Веб-застосунки можуть бути чудовим прикладом цьому.

- не потрібен багатий фінансовий ресурс
- навчання стає доступнішим
- не залежить від місцезнаходження
- займає менше часу
- вивчення може відбуватися скоріше та за коротший термін часу, аніж на курсах.
- навчання у тому напрямку, який подобається саме тобі

Існує багато методів навчання нових слів, але значна кількість досліджень стверджують, що вивчення нових слів за допомогою карток – це дійсний метод для досягнення мети.

Отже, з'ясовується, що існує нагальна потреба у якісних веб-застосунках по вивченню іноземних мов за допомогою карток.

1.2 Огляд аналогів

Сфера розробки веб-застосунків для вивчення іноземних мов за допомогою карток наразі не є надпоширеною, тому їх буде розглянуто декілька .

Перша з таких освітніх платформ - це LingoLeo. Вона застосовує ігрову механіку у своїй методиці.

Також, LingoLeo надає змогу:

- Позначити кількість слів, яку хочеш вивчати за день
- Моніторити статистику за словами, що вивчаєш (Рис. 1.2.3)
- Вибирати самому теми наборів слів
- Дивитися на приклад вживання слова, що вивчаєш
- Почути звучання слова
- Вивчати граматику на різних рівнях (Рис. 1.2.1)
- Переглянути інформацію про зареєстрованого користувача (Рис. 1.2.2)
- Додати нове слово для вивчення самостійно

З недоліків:

LingoLeo – це російська освітня платформа, через що немає доступу до деяких функцій, як, наприклад, функція «написати нам».

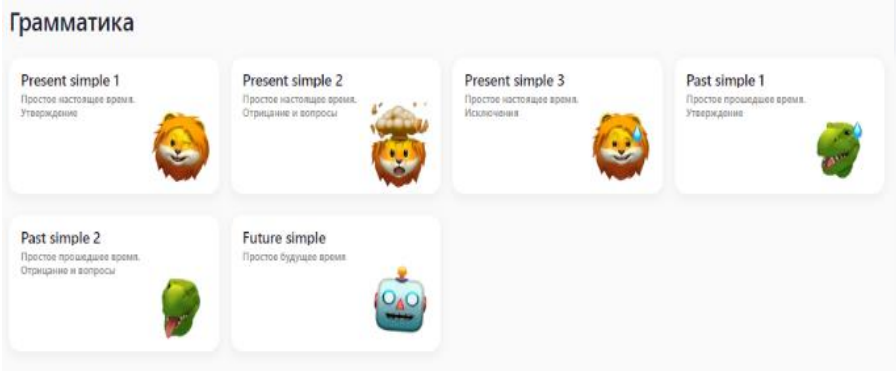

Рис. 1.2.1

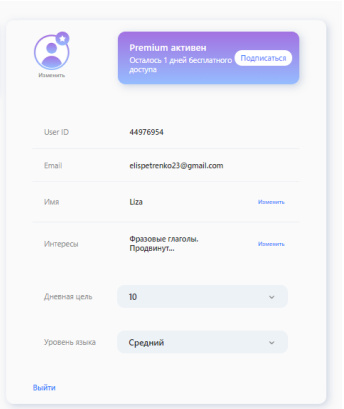

Рис. 1.2.2

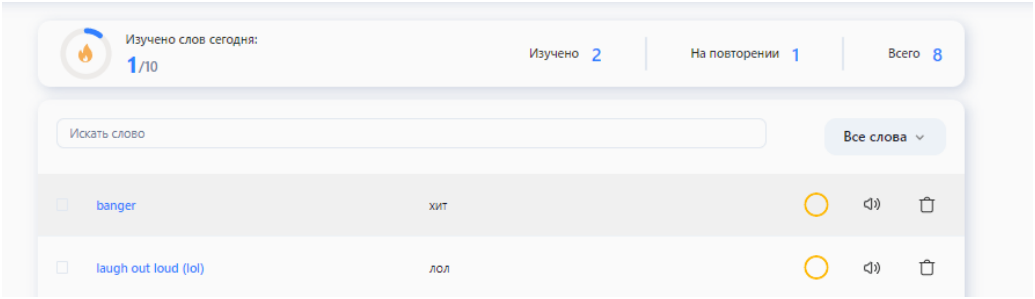

Рис. 1.2.3

Наступний аналог – це Chegg Prep. Він вважається безкоштовним цифровим мобільним та веб-застосунок, який дає багато можливостей у самостійному вивченні мови.

Chegg Prep забезпечує:

- Створення, редагування або видалення колод з новими словами
- Можливість оренди або покупки книг
- Онлайн репетиторство
- Доступ до вже створених колод за відповідною темою (Рис.1.2.6)
- Перегляду статистики по вивченим словам з колоди (Рис.1.2.4)
- Практика слів у двох форматах: флеш-картки та широкий вибір відповідей (Рис.1.2.5)

З недоліків:

Взагалі Chegg Prep показала себе як потужна платформа, але, на жаль, вона не надає можливість тестувати усі картки одразу, а лише окремо по колодах.

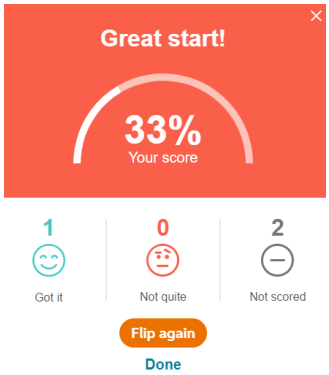

Рис. 1.2.4

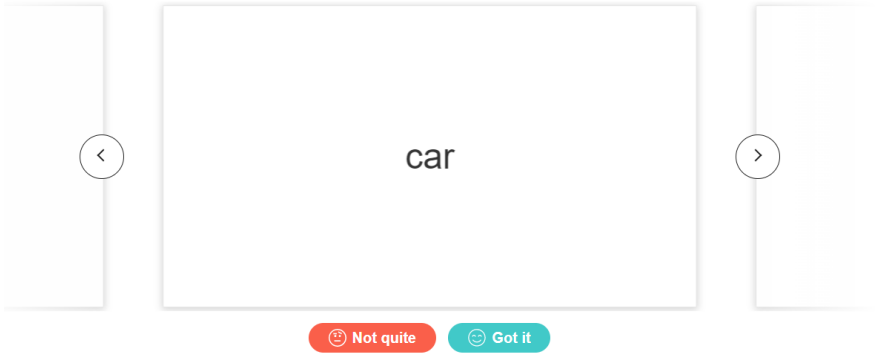

Рис. 1.2.5

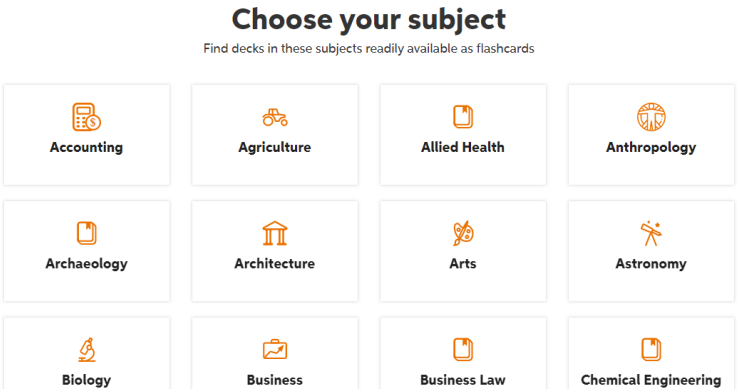

Рис. 1.2.6

Розглянемо наступного конкурента – Quizlet. У процесі тестування він проявив себе як якісна платформа для навчання з багатьма можливостями для вивчення слів.

Він надає змогу:

- Створювати картки з додаванням малюнку, запису вимови, пропонування похідних слів, від того, яке заповнюється
- Навчатися у різні формах застосування карток: флеш, завчання, формат тесту та підбір (Рис.1.2.8, Рис.1.2.9)
- Перегляду особистої сторінки з останніми діями
- Навчатися за уроками, створеними експертами (Рис.1.2.7)
- Шукати відповіді на запитання, які виникли у процесі навчання

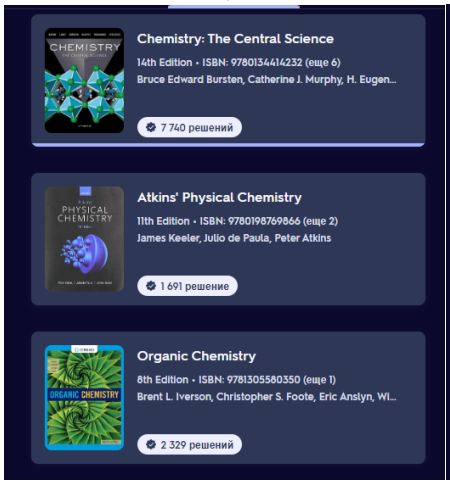

Рис. 1.2.7

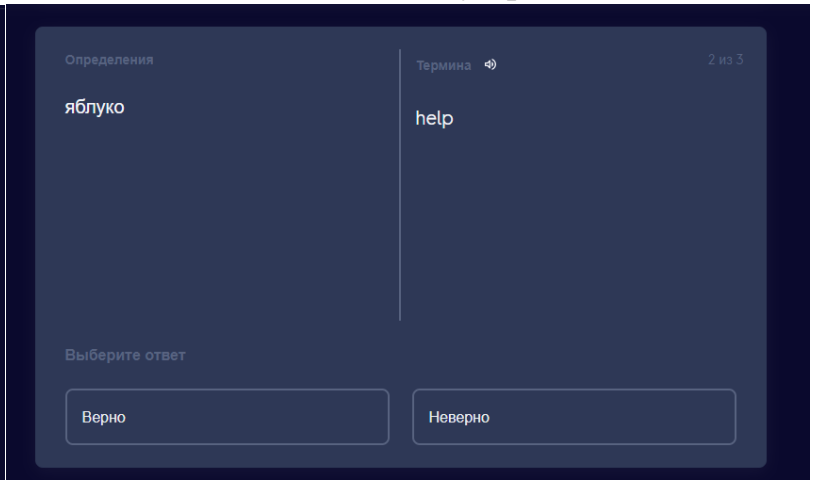

Рис. 1.2.8

Рис. 1.2.9

1.3Постановка завдання

У результаті ретельного аналізу предметної області, детального вивчення роботи та тестування аналогів застосунку, а також вивчення проблематики теми сформоване певне уявлення про завдання, вимоги та цілі, які потрібно реалізувати.

Потрібно створити зручний у користуванні застосунок з приємним, нескладним та ненагромадженим інтерфейсом, який буде сприяти запам’ятовуванню нових іноземних слів за допомогою декількох методів, основний з яких – це метод запам’ятовування інформації за допомогою карток. Веб-застосунком передбачається один тип користувача – студент, який буде вивчати слова. Також, веб-застосунок передбачає доступ і до незареєстрованих студентів. Отже, основні функціональні вимоги – це:

- Створення та зберігання карток зі словами у спеціально створених колодах.
- можливість зберігати у картці не тільки слово, яке вивчається та його переклад, а також і додаткова інформація, в якій може міститись приклад речення з певним словом або стійкий вираз, приказка тощо. Для більш якісного запам’ятовування у додатковій інформації також може міститись слово – асоціація, яке буде пов’язано з навчальним словом.
- реєстрація нових користувачів
- інтерфейс відповідно до сьогоденних вимог UI/UX
- перегляд збереженої інформації про користувача
- перегляд інформації про застосунок та зворотній зв’язок з розробником
- зручне вивчення нових слів

Зважаючи на вимоги до системи та певний проведений аналіз конкурентів, веб-застосунок повинен мати такі інформаційні сторінки, як:

- Для авторизованих користувачів сторінка «Профіль» для перегляду їх даних
- Сторінка «Про нас» для опису застосунку та методів, що використовуються для навчання

- Сторінка «Контакти» для зворотнього зв’язку

1.4Висновки

Був проведений аналіз предметної області, також було визначено актуальність та проблематику теми. Окрім цього, розглянуті конкурентні аналоги застосунку, їх переваги та недоліки. Спираючись на це, було визначено основні тезиси для реалізації веб-застосунку.

РОЗДІЛ 2. ТЕОРИТИЧНІ ВІДОМОСТІ.

2.1. Аналіз засобів розробки веб-застосунків

Для того, щоб розробити веб-застосунок відповідно до вже визначених функціональних вимог, потрібно правильно та раціонально підібрати засоби розробки програми.

Обираючи мову розробки застосунку, доцільним буде зупинитися на JavaScript, бо саме ця мова має купу переваг:

- підтримується усіма сучасними веб-браузерами,
- дозволяє робити веб-застосунки інтерактивними, робити анімації, реагувати на користувацькі дії,
- взаємодіяти з сторонніми сервісами,
- має велику кількість бібліотек та фреймворків, які спрощують розробку,
- використовується на серверному боці завдяки платформам, як Node.js.

Тож, на сьогоднішній день, JavaScript вважається незамінною мовою для розробки веб- застосунків, які мають працювати на різних платформах та пристроях.

Обираючи засоби збірки та налаштування проекту, варто звернути увагу на Vite.js.

Vite.js – це один з інструментів збірки, який налаштовує середовище розробки та покращує роботу фронтенду. За допомогою Vite.js можна працювати з такими фреймворками, як Vue.js, React, Svetle та JavaScript.

Головна перевага Vite.js полягає у його швидкості роботи. Використання модульних завантажувачів значно прискорює збирання і запуск веб-застосунку. Окрім того, це ще й підвищує продуктивність розробки та зменшує час, необхідний для збирання проекту.

Також, обираючи бібліотеки для створення веб-застосунків, гарним рішенням буде зупинитися на такій бібліотеці, як React.

React.js – одна з найпопулярніших бібліотек JavaScript для створення інтерфейсів користувача. Вона використовується для створення компонентів, які знаходяться у браузері і відповідають за відображення та управління даними на сторінці.

При розробці веб-застосунку код може дуже легко стати складним, навантаженим, мати багато різних рухомих частин тощо. Саме тому

розробники прийшли до створення бібліотеки, яка може значно полегшити життя.

React має багато переваг і от декілька з них:

- React використовує компоненти, що дозволяє створювати повторно використовувані компоненти. Це значно зменшує кількість дублювання коду і полегшує підтримку.
- React використовує віртуальний DOM, що дозволяє зменшити кількість звернень до реального DOM.
- Також, React має декларативний підхід, тобто дозволяє описувати, які елементи потрібно відобразити на сторінці.
- Широка спільнота розробників також вважається значущою перевагою React, бо вона надає підтримку та розвиває екосистему інструментів та бібліотек.

Хоча React має багато переваг, він також має деякі недоліки:

- Він вимагає більше коду для розробки, ніж деякі інші фреймворки.
- React не має вбудованої підтримки для маршрутизації та керування станом додатка, що може вимагати використання сторонніх бібліотек.

Для того, щоб керувати станом застосунку у React або інших JavaScript фреймворках, управляти станом додатку централізовано та прогнозовано, можна зупинитися саме на Redux.

Redux – невелика бібліотека JavaScript, яку можна використовувати із будь-яким зовнішнім фреймворком, таким як React, Angular, jQuery. Вона була створена для того, щоб зробити управління станом передбачуваним, надаючи єдиний контейнер стану та суворі правила щодо того, як можна змінити стан.

Redux використовує схему «simple source of truth». Це означає, що єдине джерело істини стосується лише переміщення стану програми та всієї пов’язаної логіки за межами програми, дозволяючи будь-якому компоненту отримувати доступ до необхідних даних. Таке рішення значно полегшує керування станом програми, оскільки можна отримувати доступ до даних і змінювати їх із будь-якого компонента , якому вони потрібні, без необхідності передавати дані.

Redux Toolkit – це набір інструментів, що надає зручний та простий спосіб для розробки додатків на базі бібліотек Redux. Він потрібен для того, щоб полегшити та прискорити процес розробки, забезпечуючи стандартний підхід до налаштування та використання бібліотеки. Також, він дозволяє

зберігати стан додатка в централізованому місці і оновлювати його за допомогою чистих функцій – “reducers”.

Основними перевагами Redux Toolkit вважається наступне:

- використання конфігурації за замовчуванням, що дозволяє швидко налаштувати Redux store та зменшити кількість додаткового коду.
- Redux Toolkit добре інтегрується з іншими пакетами, наприклад, Redux DevTools, React-Redux.
- Redux Toolkit дозволяє спростити саму розробку, зменшуючи кількість необхідного коду для налаштування Redux store, диспетчеризації дій та створення редукторів.

Для більш зручної розробки застосування та його стилізації також варто приділити увагу такому розширенню CSS, як SCSS.

SCSS - це препроцесор для CSS, який дозволяє розширити можливості звичайного CSS за допомогою нових функцій та конструкцій. SCSS використовує синтаксис, який є розширенням синтаксису звичайного CSS. Тобто, можна писати CSS-код, який виглядає як звичайний CSS, але з деякими додатковими можливостями такими, як змінні, цикли, оператори, міксіни, функції та інше. Такі конструкції можна використовувати для збереження значень кольорів, розмірів шрифтів, що дозволяє легко змінювати значення на всій сторінці замість того, щоб змінювати кожен елемент окремо.

- До того ж, SCSS підтримує вкладеність, що дозволяє вбудувати один CSS-блок в інший.
- Функції у SCSS дозволяють створювати власні функції для повторюваних обчислень. Прикладом може бути обчислення розмірів контейнера з врахуванням відступів та величини шрифту.
- Оператори надають можливість виконувати математичні операцій, на кшталт, додавання розмірів, віднімання, множення і таке інше.
- Також, у SCSS є міксіни. Вони дозволяють створювати групи властивостей, які можна застосувати до різних елементів на сторінці. Це надає можливість зменшити кількість коду та зробити його більш зрозумілим.

2.2. Опис інструментів розробки

При виборі інструментів розробки веб-застосувань, важливо звертати увагу на наступні пункти:

- Функціональність

- Розширюваність
- Документацію
- Ефективність
- Підтримку

Тож, при створенні застосунків значно можуть полегшити роботу інструменти, вказані нижче.

Github – дуже зручна онлайн-платформа керування версіями програмного забезпечення та спільної розробки проектів. Вона базується на системі керування версіями Git і надає зручний інтерфейс для спільної роботи розробників над кодом.

ESLint – це інструмент для статичного аналізу коду в проектах на JavaScript. Він використовується для виявлення потенційних проблем та неправильного стилю коду, що допомагає забезпечити високу якість програмного коду.

Uuid – бібліотека JavaScipt, що надає зручний спосіб генерації UUID. Основна перевага використання UUID полягає у тому, що вони є гарантовано унікальними.

VS Code – редактор та інтегроване середовище розробки. Підтримує ряд мов програмування, включаючи JavaScript. VS Code є одним з найпопулярніших виборів при розробці веб-застосувань, тому що він забезпечує багатофункціональність, розширюваність, підтримку JavaScript та Node.js, кросплатформеність. Також, є легким у використанні та має активну спільноту розробників, що активно розвивається.

Redux-persist – бібліотека для зберігання стану застосунку Redux у JavaScript. Вона забезпечує зручний спосіб конфігурації збереження стану Redux і автоматично синхронізує зміни стану з обраним сховищем. Також ця бібліотека надає можливість налаштувати які саме частини стану потрібно зберігати.

Hooks – підходи або функції бібліотеки React, які дозволяють компонентам зберігати і змінювати стан, підписуватися на певні події та інше.

2.3. Вибір СКБД

Після визначення засобів розробки проекту переходимо до вибору СКБД. Тут є декілька важливих пунктів, на які потрібно звертати увагу, а саме:

- Підтримка JavaScript

- Швидкодія та продуктивність
- Сумісність
- Масщтабованість
- Надійність та безпека

Тому мене зацікавив саме Firebase.

Firebase – це платформа для розробки веб-застосувань та мобільних додатків, яка надає ряд інструментів для ефективної роботи. Вона включає у себе безліч сервісів, які допомагають не відволікатися на backend і більше зосередитись на UI/UX.

Я певна, що саме ця платформа є гарним вибором, тому що

- Firebase надає відмінну підтримку JavaScript. За допомогою SDK Firebase для JavaScript можна легко з'єднуватися з базою даних, виконувати запити та отримувати дані.
- Firebase базується на NoSQL, що забезпечує хорошу швидкодію при роботі з даними. Також, Firebase надає кешування даних на клієнтському пристрої.
- Firebase інтегрується з багатьма технологіями розробки, такими як JavaScript, React, Angular, Vue.js.
- За допомогою Firebase можна легко масштабувати свій застосунок використовуючи інструменти Realtime Database, Cloud Firestore та Cloud Functions.

Однією з ключових складових Firebase є сервіс авторизації Firebase Authentication. Він надає сервісні служби, прості у використані SDK і готові бібліотеки інтерфейсу для автентифікації користувачів у застосунку. Він, також підтримує автентифікацію з використанням паролів, телефонних номерів та популярних постачальників федеративних посвідчень, а саме Google, Firebase, Twitter тощо.

Можна використовувати або FirebaseUI, як повноцінне рішення для автентифікації, або Firebase Authentication SDK, для інтегрування одного або кілька методів входу до програми.

Ще одним важливим сервісом є Firebase Cloud Firestore.

Cloud Firestore – це повнофункціональна база даних, яка забезпечує зберігання та синхронізацію даних між клієнтами веб-застосування Firebase використовує модель документів-колекцій, що дозволяє легко організовувати та опрацьовувати дані. Окрім того, Cloud Firestore підтримує оновлення даних в реальному часі, що дозволяє створювати реактивні застосування без

необхідності написання додаткового коду сервера. Також, Firestore надає масштабовану структуру бази даних, яка дозволяє легко масштабувати застосунок, навіть якщо він отримує великий обсяг запитів.

2.4. Висновок до розділу

Отже, було проаналізовано одні з найпопулярніших засобів розробки веб-застосунків таких, як Vite.js, React, Redux, SCSS та описано інструменти їх розробки (VS Code, ESLint, uuid, Github, redux-persist, hooks). Також, було визначено та обґрунтовано вибір мови JavaScript та СКБД Firebase.

РОЗДІЛ 3. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ.

3.1. Аналіз технічного завдання

Основна мета даного веб-застосунку – створити сучасне середовище для вивчення слів за допомогою карток. Сам процес вивчення містить декілька підходів до запам’ятовування, а саме - метод Flash, Contextual Relearning та метод асоціацій. Перевагою також може бути візуальне запам’ятовування слів.

У застосунку наявна лише одна група користувачів – студенти, що вивчають слова. Вони мають доступ до авторизації, створення нових колод, загальну інформацію про застосунок та доступ до зворотнього зв’язку. Окрім того, вже зареєстровані користувачі мають доступ до вивчення вже створених колод, перегляду персональної інформації та виходу.

Інтерфейс веб-застосунку складається зі сталого верхнього меню, у якому є такі розділи, як:

- Головна сторінка – основна сторінка, де розміщена кнопка “Get Started”.

При натиску на цю кнопку можна створити нову колоду карток для запам’ятовування. Якщо користувач не зареєструвався, після виходу інформація не зберігається. У разі, якщо користувач – зареєстрований, усі дані, що він увів – зберігаються у базі даних.

- Сторінка з колодами – сторінка, де містяться усі створені зареєстрованим користувачем колоди. В іншому випадку є можливість зробити авторизація або аутентифікацію за допомогою відповідних форм Login та Sign up.
- Інформація про застосунок – на сторінці цього розділу розміщена інформація про методи навчання, дослідження, які підтверджують якість запам’ятовування тощо.
- Контактна сторінка – цей розділ призначений для зворотнього зв’язку, де можна ввести таку інформацію про себе, як своє ім’я, текст повідомлення та e-mail.
- Персональна інформація – при натиску на цей розділ є можливість або переглянути інформацію про себе, якщо користувач зареєстрований, або зареєструватись, або вийти зовсім.

3.2. Опис розробки застосунку

Після аналізу технічного завдання переходимо до розглядання архітектури проекту.

Взагалі, при розробці проектів різних розмірів, існують умовні базові структури для для них.

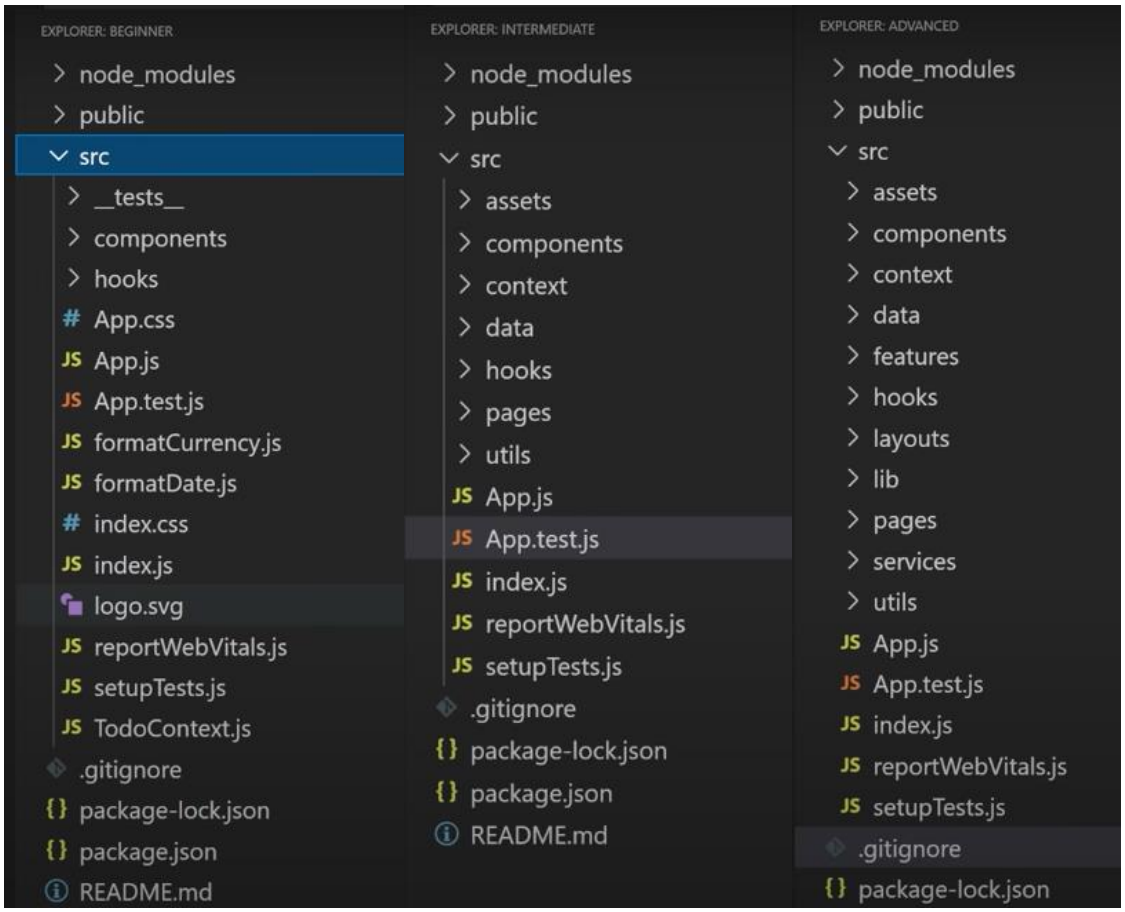

Рис. 3.2.1 *Рис. 3.2.2* *Рис. 3.2.3*

На прикладах, зображених на рис.3.2.1, рис.3.2.2, рис.3.2.3, можна побачити, щоб чим більший проект, тим складніше зв’язність між файлами, та, одночасно, і простіша структура з точки зору масштабованості.

Рис. 3.2.4

У своєму прикладі (Рис.3.2.4) я дотримуюсь легкості, але й водночас використовую Page-oriented Structure.

Assets – ця папка містить такі дані, як зображення, svg-файли, global css, будь-який тип ресурсу, який імпортується. Досить зручно зберігати всі ці файли в одному місці, щоб мати легкий доступ, коли вони знадобляться.

Common – ця папка зберігає загальні ресурси, модулі, бібліотека та ресурси, які надалі можуть використовуватися у проекті.

Components – у цій папці розміщується компоненти та модулі для програмного проекту.

Config – тут зберігаються конфігураційні файли, які містять параметри налаштувань проекту.

Helpers – містить допоміжні модулі та класи, які надають підтримку основним частинам проекту.

Hooks – у цій папці розміщуються файли хуки для керування різними подіями і функціями у процесі розробки проекту.

Pages – тут знаходяться сторінки , що використовуються для створення застосунку.

Services – у цій папці розміщуються файли та модулі, що відповідають за логіку та функціональність сервісів у проекті. Файли містять код, який обробляє дані та зебезпечує взаємодію з іншими частинами проекту.

Store – папка містить файли, які пов’язані зі зберіганням і керуванням станом веб-застосунку.

Стилі CSS:

Коли браузер відтворює HTML-сторінку, він застосовує базові стилі ще до написання нами стилізації. Хоча всі браузери застосовують свої основні стилі, кожен браузер має особливості, відмінні від інших, і це спричиняє проблему невідповідності. Для вирішення цієї проблеми існують два шляхи – Normalize CSS та Reset CSS.

Normalize CSS – забезпечує узгодженність між браузерами в стилях HTML за замовчуванням, коли Reset CSS – зовсім відмовляється від стилів браузерів за дефолтом. Але проблема з Reset CSS полягає у тому, що при скиданні стилів вони мають жахливий вигляд та роблять купу непотрібних визначень. До того ж, вони не читаються під час налагодження.

Для того, щоб уникати таких недоліків можна використовувати Normalize CSS разом із невеликою частиною Reset CSS.

Далі розглянемо роботу функції App()(Рис. 3.2.5):

```
function App() {
    return (
        <>
            <Header />
            <main className="main">
                <Routes>
                    <Route path={AppRoutes.ROOT} element={<Home />} />
                    <Route path={AppRoutes.PROFILE} element={<Profile />} />
                    <Route path={AppRoutes.CONTACTS} element={<Contacts />} />
                    <Route path={AppRoutes.DECKS} element={<Decks />} />
                    <Route path={AppRoutes.ABOUT} element={<About />} />
                    <Route path={AppRoutes.ADD_DECK} element={<AddDeck />} />
                    <Route path={AppRoutes.ADD_CARD_$DECK_ID} element={<AddCard />} />
                    <Route path={AppRoutes.LOGIN} element={<Login />} />
                    <Route path={AppRoutes.SIGNUP} element={<Login isLogin={false} />} />
                    <Route path={AppRoutes.ANY} element={<NotFound />} />
                </Routes>
            </main>
        </>
    );
}

export default App;
```

Рис. 3.2.5

Ця функція використовує компонент <Routes> для навігації між різними сторінками застосунку. (Рис. 3.2.5)

```
<Route path={AppRoutes.ROOT} element={<Home />} />
```

Рис. 3.2.7

Рядок, зображений на рис. 3.2.7 вказує для AppRoutes.ROOT відображення компонента Home.

Так само і для інших:

```
<Route path={AppRoutes.PROFILE} element={<Profile />}
```

Рис. 3.2.8

Для AppRoutes.PROFILE – відображення компонента Profile. (Рис. 3.2.8)

```
<Route path={AppRoutes.ANY} element={<NotFound />} />
```

Рис. 3.2.9

Цей Route (Рис. 3.2.9) вказує на те, що для будь-якого шляху, який не відповідає жодному попередньому , повинен бути відображений компонент <Not found>.

Далі розглянемо використання бібліотеки Persist.(Рис. 3.2.10)

```
import { configureStore } from '@reduxjs/toolkit';
import storage from 'redux-persist/lib/storage';
import { persistReducer, persistStore } from 'redux-persist';
import { rootReducer } from './root-reducer';

const extraArgument = {};

const persistConfig = {
    key: 'root',
    storage,
    whitelist: [],
};

const persistedReducer = persistReducer(persistConfig, rootReducer);

const store = configureStore({
    reducer: persistedReducer,
    middleware: (getDefaultMiddleware) => {
        return getDefaultMiddleware({
            thunk: { extraArgument },
            serializableCheck: false,
        });
    },
});

const persistor = persistStore(store);

export { extraArgument, store, persistor };
```

Рис. 3.2.10

У цьому коді (Рис. 3.2.10) визначається конфігурація для постійного збереження стану за допомогою “redux-persist”. Цей об’єкт містить ключ “key”, модуль “storage” та опцію “whitelist”, яка вказує які частини стану потрібно зберігати.

Далі створюється новий persistReducer, який об’єднує конфігурацію збереження та корневий редуктор. Це забезпечить зберігання та відновлюваність стану за допомогою “redux-persist”.

Далі створюється store за допомогою configStore(). Функція middleware приймає параметр “getDefaultMiddleware” , що повертає стандартні middleware Redux. Далі передається thunk з аргументом “extraArgument”, окрім цього параметр “serializableCheck” встановлюється на false.

Далі викликом функції “persistStore(store) ” створюється об’єкт “persistor”, який використовується для збереження стану додатку.

3.3. Тестування програми і результати її виконання

При запуску застосунку перше, що бачить користувач – сторінка “Home”(Рис 3.3.1). На ній розміщена назву застосунку та кнопка “Get Started”. Варто зазначити, що збереження наступної роботи залежить від реєстрації користувача:

1) Якщо користувач вже зареєстрований усі данні будуть зберігатися у Firebase. Також, при наступних входах він зможе переглядати те, що було збережено за час минулого використання.
2) У випадку, якщо користувач незареєстрований – всі данні, які він введе та уся робота буде зберігатися локально доти, доки він не вийде із застосунку.

Отже, після натискання на кнопку з’являється повідомлення про те, що користувач ще немає жодної колоди, та з’являється пропозиція створити нову у вигляді кнопки «+». (Рис. 3.3.2)

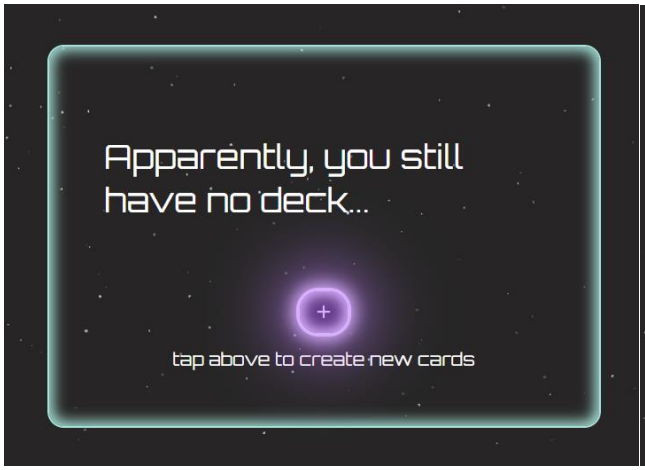

Рис. 3.3.2

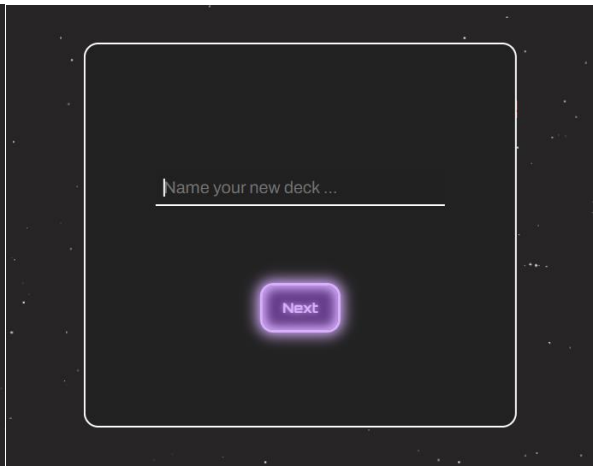

Рис. 3.3.3

При натиску на цю кнопку з’являється вікно з полем, куди треба ввести назву колоди, та кнопка «Next». (Рис. 3.3.3)

Після заповнення поля назви колоди та натисканні на кнопку, з’являються нові поля для введення слова, що потрібно вивчити, його переклад, та додаткової інформації. Поле для додаткової інформації вважається вибірковим, натомість у випадку пустих два інших поля кнопки «Create card» та «Done», які знаходяться нижче, не будуть працювати. (Рис. 3.3.4)

При натисканні на кнопку «Create card» - інформація, яку користувач увів до цього буде збережена, та відбудеться візуальна імітація знаходження тільки що створеної картки під наступною.

При натисканні на кнопку «Done» - колода буде створена, вікно створення карток зникне і користувач зможе побачити свою нову колоду на сторінці. Кнопка «Done» спрацьовує лише тоді, коли створена хоча б одна картка.

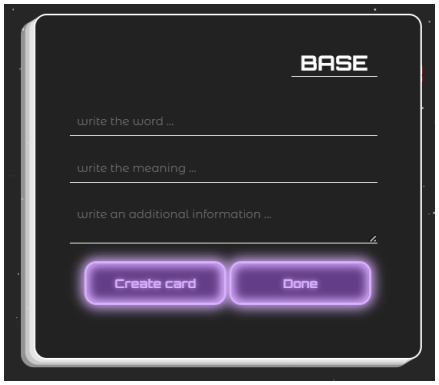

Рис. 3.3.4

Відображення колод:

На сторінці зі створеними колодами можна побачити групи карток з назвою колоди, кнопкою редагування, відсотком вивчених слів та кнопкою “Start”. (Рис. 3.3.5)

При натисканні на цю кнопку відбувається запуск колоди. Користувачу показується слово, переклад якого треба вивчити та додаткова інформація, якщо вона є.

При натисканні на картку – вона перевертається та видно її переклад.

Також користувачу доступні дві кнопки “Don`t know” та “Know”. В залежності від відповіді користувача натискається відповідна кнопка, та зберігається результат. Далі з’являється наступна картка. (Рис. 3.3.9)

Рис. 3.3.9

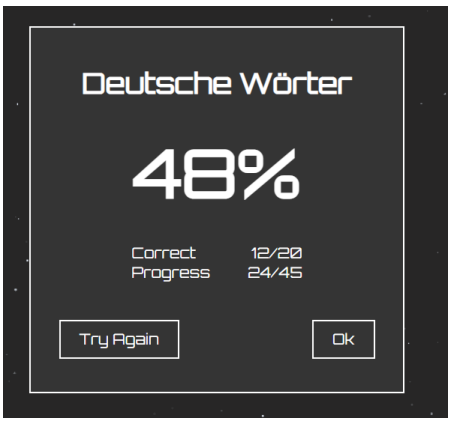

Рис. 3.3.10

Коли користувач пройшов навчання до останньої картки, з’являється вікно з відсотком вгаданих слів та кнопка “OK”. (Рис. 3.3.10)

Окрім цього, користувачу доступне верхнє меню з такими розділами, як ”My Decks”, ”About”, ”Contacts” та ”Login”.

При натисканні на розділ ”My Decks” відбувається перевірка чи зареєстрований користувач. Якщо ні, то одразу пропонується зареєструватись. Якщо так, то на сторінці видно усі створені колоди користувачем.

У розділі ”About” можна переглянути інформацію про застосунок, методі навчання тощо.

У розділі ”Contacts” користувач має можливість зв’язатися з розробником.(Рис.3.3.6)

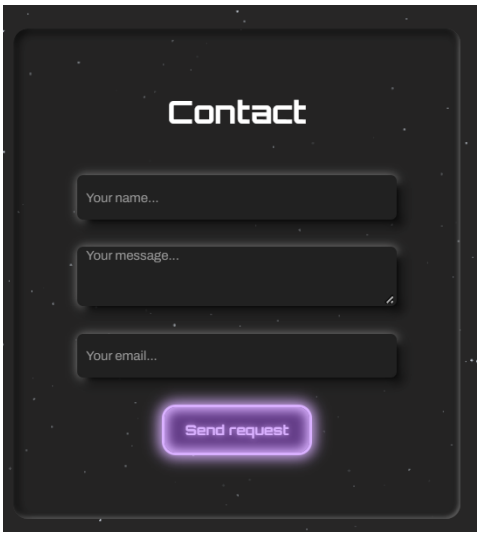

Рис. 3.3.6

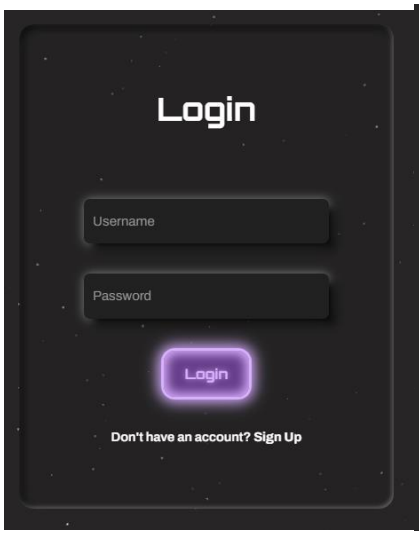

Рис. 3.3.7

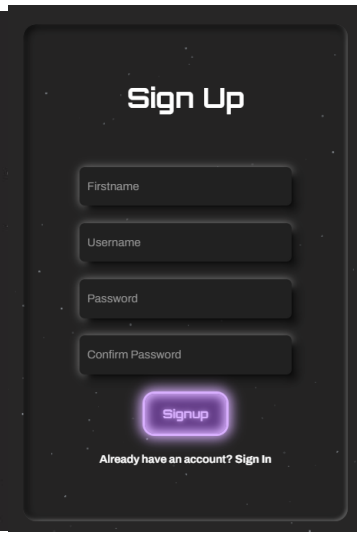

Рис. 3.3.8

У розділі ”Login” користувач може авторизуватися, пройти автентифікацію або вийти. (Рис. 3.3.7 та Рис. 3.3.8)

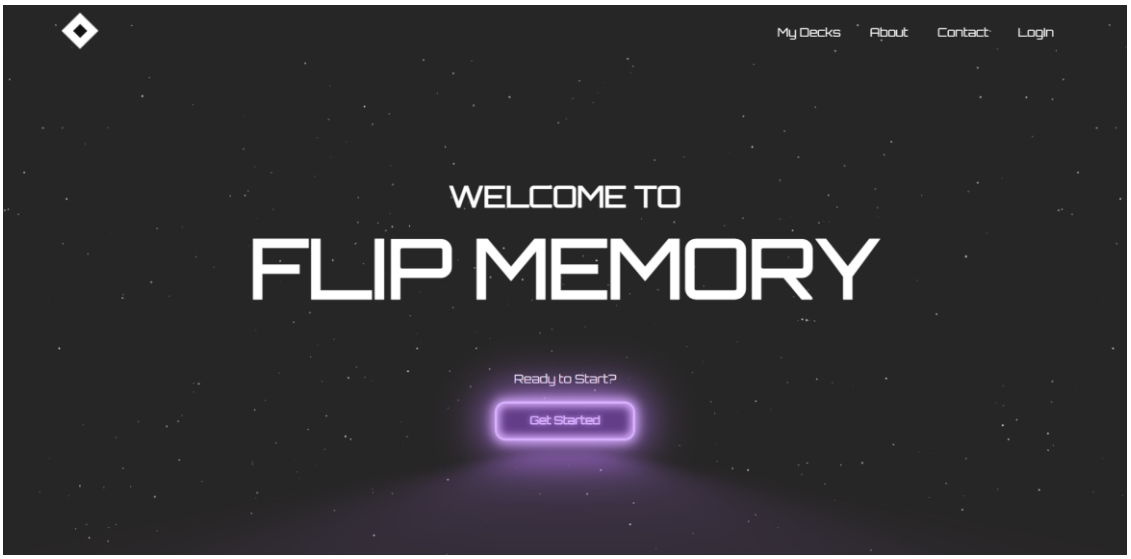

Рис. 3.3.1

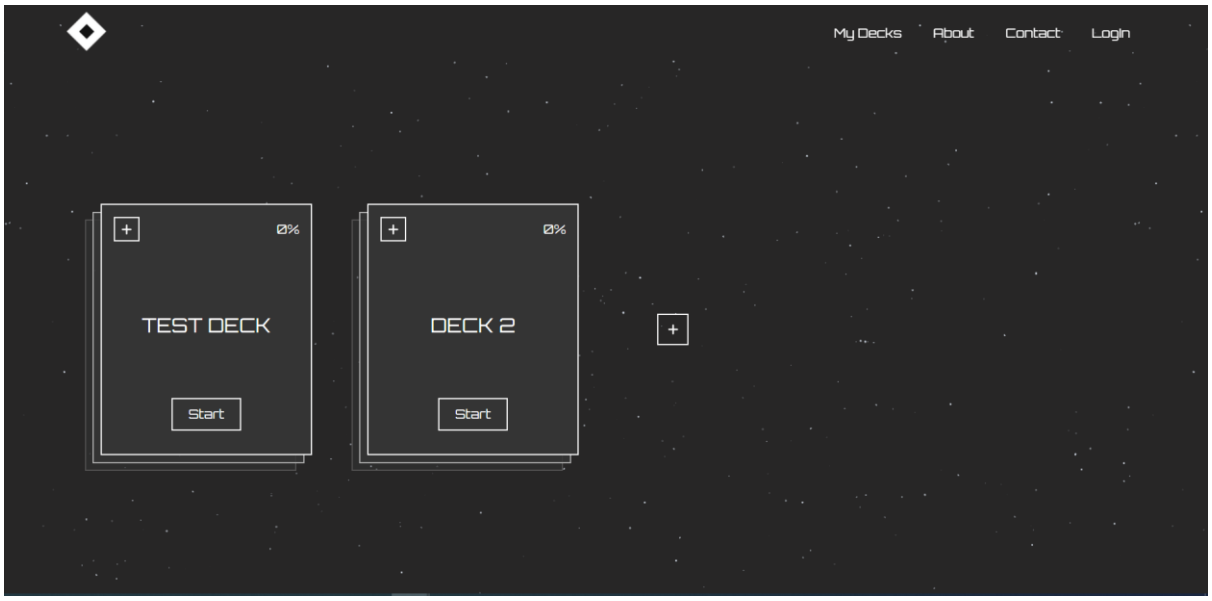

Рис. 3.3.5

3.4. Висновки до розділу

Отже, у цьому розділі було проаналізовано стилі CSS, було продемонстровано роботу бібліотеки Persist та компоненту <Routes> . Окрім цього, були продемонстровані результати виконання програми.

ВИСНОВКИ.

Тож, у процесі роботи було створено веб-застосунок для вивчення іноземних мов за допомогою карток. Перед початком розробки проекту було ретельно проаналізовано предметну область, детально вивчено роботу та тестування аналогів. Були визначені функціональні вимоги.

Також, було проведено аналіз засобів розробки, опис інструментів та вибрано СКБД. Веб-застосунок був реалізований за допомогою таких засобів, як : Vite.js, React, Redux та SCSS. Окрім цього у проекті були використані такі інструменти розробки, як VS Code, ESLint, uuid, Github, redux-persist, hooks. Для роботи також була визначена СКБД Firebase.

Перспективи розвитку:

- Імпорт / експорт карток
- Обмін картками / колодами
- Створення додаткових форматів використання карток

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] https://firebase.google.com/docs/auth

[2] https://github.com/

[3] https://uiverse.io/

[4] https://firebase.google.com/docs/firestore/rtdb-vs-firestore

[5] https://avada-media.ua/ua/services/firebase/

[6] https://telegram.com

[7] https://lingualeo.com/ru/leo-new/trainings

[8] https://quizlet.com/latest

[9] https://www.chegg.com/flashcards

[10]https://www.youtube.com/watch?v=UUga4-z7b6s&t=5s&pp=ygUPcmVhY3Qgc3RydWN0dXJl

[11] https://elad.medium.com/normalize-css-or-css-reset-9d75175c5d1e

[12] https://elad.medium.com/css-architecture-folders-files-structure-f92b40c78d0b

[13] https://www.cssportal.com/css-generators.php

[14] https://courses.webdevsimplified.com/react-hooks-simplified

[15] https://canva.com

[16] https://miro.com

[17] https://figma.com

[18] https://www.sololearn.com/learning/

[19] https://itproger.com/ua/course/sass

[20]https://maxstizhko.com/uk/blog/vite-react-tutorial-getting-started-with-lightning-fast-development