2. Опис методів та засобів вирішення задачі

2.1. Запропоновані методи та засоби вирішення задачі

Проаналізувавши ринок було виявлено що на даний момент відсутні додатки які б задовольняли всі потреби, а саме:

* Навчити людину термінології на українській мові;
* Провести тестування користувача по цим термінам;
* Навчити людину правил української мови
* Провести тестування по даним правилам
* Ведення статистики по проходженню;
* Створення декількох профілів для проходження декільком користувачам на одному пристрої.
* Україномовний інтерфейс

На основі вище перерахованих потреб було вирішено створити даний програмний засіб.

Для вирішення поставленої задачі було запропоновано такі варіанти форматів додатків:

* Створення прикладного додатку для персонального комп’ютера;
* Створення веб-додатку;
* Створення мобільного додатку.

Суттю даного програмного продукту є те що користувач міг користуватися додатком в вільний час. Дана програма призначена для самоосвіти, а зазвичай у людини яка працює чи навчається не досить багато вільного часу і тому було вирішено зробити веб-додаток. Головною перевагою такого додатку є те що він «веб-додаток» тобто користувач може використати в будь-який момент на будь якому пристрої який підключений до мережі, а це саме те що було потрібно для продуктивного опрацювання матеріалу з додатку.

Враховуючи все вище сказане можна зробити висновок що хоч додаток для персонального комп’ютера працює найшвидше, додаток для мобільного пристрою зручніший у використанні на телефоні, проте найкращим вибором є веб-додаток так як доступ до нього можна здійснити з будь-якого пристрою (телефон, комп’ютер, планшет, телевізор…).

2.2 Архітектуру програмного забезпечення

Для даного програмного засобу було вирішено обрати архітектуру модені MVC.

**Model-View-Controller** (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

* Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан.
* Представлення (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.
* Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу підвищується можливість повторного використання коду. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах і / або з різних точок зору. Зокрема, виконуються наступні завдання:

До однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми;

Не торкаючись реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних) - для цього досить використовувати інший контролер;

Ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можливо добитися того, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть обізнані про те, яке уявлення буде використовуватися.

**Модель**

Модель надає дані і методи роботи з ними: запити до бази даних, перевірка на коректність. Модель не залежить від уявлення (не знає як дані візуалізувати) і контролера (не має точок взаємодії з користувачем) просто надаючи доступ до даних і управління ними.

Модель будується таким чином, щоб відповідати на запити, змінюючи свій стан, при цьому може бути вбудовано повідомлення «спостерігачів».

Модель, за рахунок незалежності від візуального представлення, може мати кілька різних уявлень для однієї «моделі».

**Представлення**

Представлення відповідає за отримання необхідних даних з моделі і відправляє їх користувачеві. Подання і не виконує жодних введені дані користувача.

Представлення може впливати на стан моделі, повідомляючи моделі про це.

**Контролер**

Контролер забезпечує «зв'язок» між користувачем і системою. Контролює і направляє дані від користувача до системи і навпаки. Використовує модель і уявлення для реалізації необхідного дії.

2.3 Обрані програмні засоби

2.3.1 Середовище розробки

Інтегроване середовище розробки, ІСP (англ. Integrated development environment - IDE), також єдине середовище розробки, ЕСР - комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення (ПО).

Середовище розробки включає в себе:

* текстовий редактор,
* компілятор і / або інтерпретатор,
* засоби автоматизації збирання,
* відладчик.

Іноді містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасні середовища розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів - для використання при об'єктно-орієнтованої розробки ПЗ. ІСР зазвичай призначені для декількох мов програмування - такі як IntelliJ IDEA, NetBeans, Eclipse, Qt Creator, Geany, Embarcadero RAD Studio, Code :: Blocks, Xcode або Microsoft Visual Studio, Visual Studio Code але є і IDE для однієї певної мови програмування - як, наприклад , Visual Basic, Delphi, Dev-C ++.

Для розробки веб-додатків в наш час актуальними є такі середовища розробки:

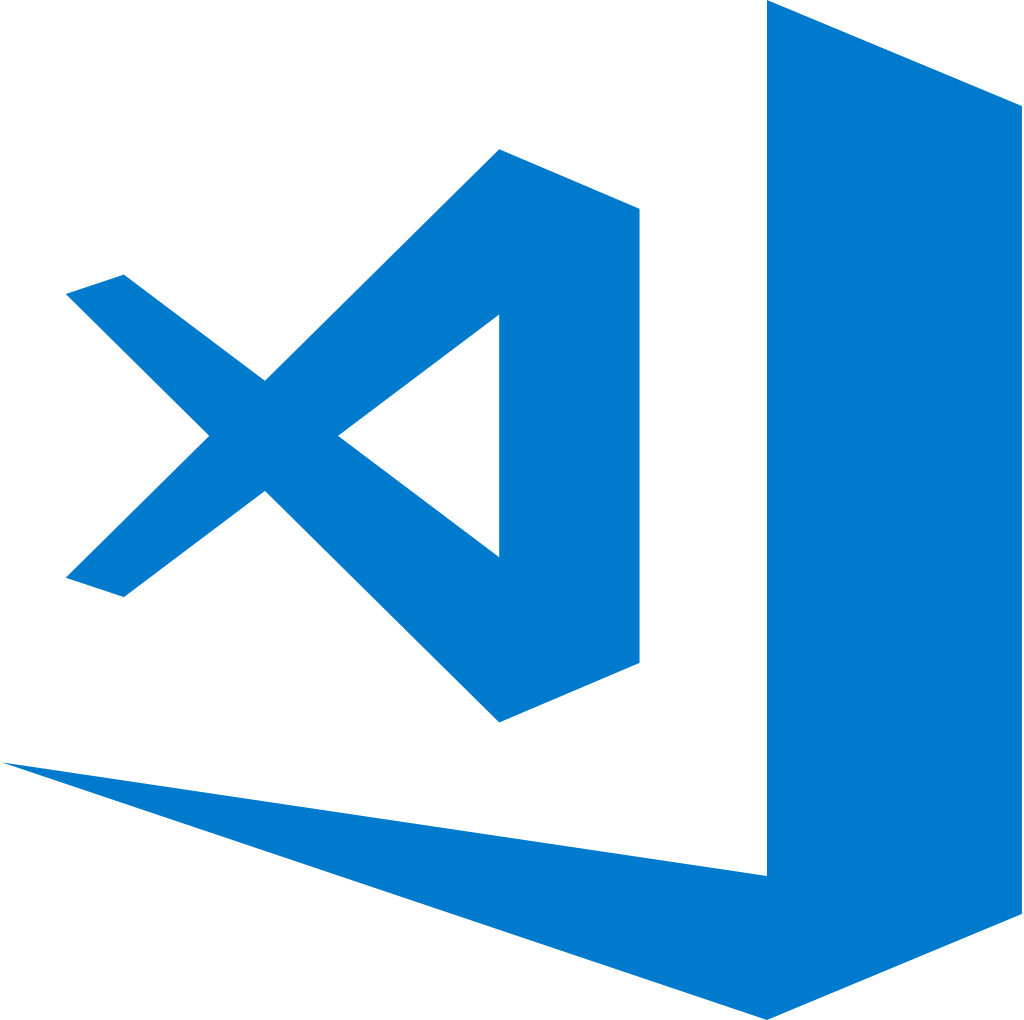


Рисунок 2.3.1.1 – логотип Visual Studio Code

1. **Visual Studio Code** — засіб для створення, редагування та завантаження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X. Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим крос-платформовим продуктом у лінійці Visual

Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js.

Примітно, що про використання напрацювань вільного проекту Atom на сайті Visual Studio Code і в прес-релізі і в офіційному блозі не згадується. Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковаге рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.

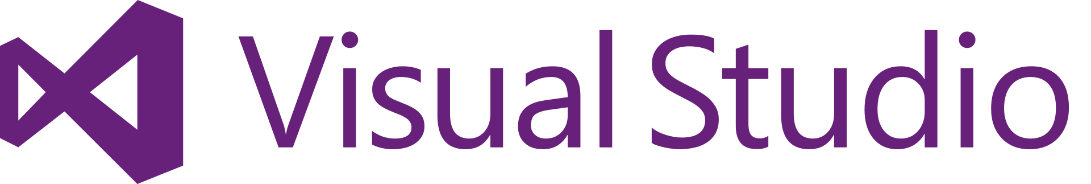


Рисунок 2.3.1.2 – Логотип Visual Studio

1. **Microsoft** **Visual Studio** - серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio 2017 - Перший нестабільний випуск наступної версії програми під умовною назвою "15" побачив світ 30 березня 2016 року. Основними змінами стали інтерфейс інсталятора та численні незначні покращення у різних компонентах середовища розробки. Очікується підтримка мови програмування Solidity.

Версія остаточно побачила світ під назвою Visual Studio 2017 7-го березня 2017-го року.



Рисунок 2.3.1.3 – Логотип MonoDevelop

1. **MonoDevelop** — відкрите інтегроване середовище розробки для платформ Linux, Mac OS X[4] та Microsoft Windows[5], передусім націлене на розробку програм, які використовують і Mono, і Microsoft .NET framework. На даний момент підтримуються мови C#, Java, Boo, Visual Basic.NET, CIL, Python, Vala, C та C++. Також MonoDevelop підтримує такі технології, як Gtk#, ASP.NET MVC, Silverlight, MonoMac и MonoTouch.

MonoDevelop включає можливості подібні до NetBeans та Microsoft Visual Studio, такі як автоматичне доповнення, інтеграція контролю коду, графічний користувацький інтерфейс і веб-дизайнер. В MonoDevelop інтегрований Gtk# GUI дизайнер під назвою Stetic.

2.3.2 Мови програмування

Мови програмування які використовуються підчас створення веб-додатку в середовищах розробки, які наведені вище:

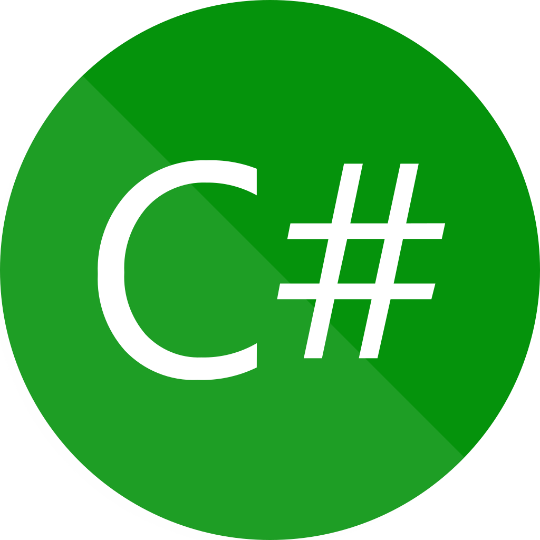


Рисунок 2.3.2.1 – Логотип C#

1. **C# (вимовляється Сі-шарп)** — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# стандартизований в ECMA[5] та ISO.

У серпні 2000 Microsoft Corporation, Hewlett-Packard та Intel Corporation виступили спонсорами стандартизації специфікації мови C#, а також Common Language Infrastructure (CLI) в організації зі стандартизації ECMA International. У грудні 2001 ECMA випустила ECMA-334 Специфікація мови C#. C# стала стандартом ISO у 2003 (ISO/IEC 23270:2006 — Information technology—Programming languages—C#). До того ECMA ще встигла адоптувати еквівалентну специфікацію як другу редакцію C# у грудні 2002.

У червні 2005 ECMA схвалила редакцію 3 специфікації C#, і відредагувала ECMA-334. Доповнення включали часткові класи, анонімні методи, тип null, і генерики (аналоги шаблонів C++).

У липні 2005 ECMA подала стандарти і відповідні технічні умови на ISO/IEC JTC 1 через пришвидшену процедуру (Fast-Track). Цей процес звичайно займає 6-9 місяців.



Рисунок 2.3.2.2 – Логотип JavaScript

1. **JavaScript (JS)** — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для: написання сценаріїв веб-сторінок для надання їм інтерактивності; створення односторінкових веб-застосунків (React, AngularJS, Vue.js); програмування на стороні сервера (Node.js); стаціонарних застосунків (Electron, NW.js); мобільних застосунків (React Native, Cordova);

сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);

всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови С, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.



Рисунок 2.3.2.3 – Логотип HTML

1. **HTML** — стандартна мова розмітки для створення веб-сторінок і веб- додатків .

З Cascading Style Sheets (CSS) і JavaScript, вона утворює тріаду основних технологій для World Wide Web.

Веб-браузери отримують HTML-документи з веб-сервера або з локальної пам'яті і передають документи в мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами , написаними з використанням кутових дужок . Теги, такі як і безпосередньо вводять вміст на сторінку. Інші теги, такі як <img /> <input /> <p>оточують і надають інформацію про текст документа і можуть включати інші теги як під-елементи. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати програми, написані на мові сценаріїв, наприклад JavaScript, що впливає на поведінку та вміст веб-сторінок. Включення CSS визначає вигляд і компонування вмісту. World Wide Web Consortium (W3C), які супроводжують як HTML і CSS стандартів, заохочує використання CSS над явним презентаційним HTML з 1997 року.

HTML впроваджує засоби для: створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше; отримання інформації із Всесвітньої мережі через гіперпосилання;

створення інтерактивних форм;

включення зображень, звуку, відео, та інших об'єктів до тексту.

2.3.3 База даних

База даних (БД) - упорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система управління БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).



Рисунок 2.3.3.1 – Логотип Microsoft SQL Server

**1. Microsoft SQL Server** - комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних. Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL (скорочено T-SQL), яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC)—інтерфейс взаємодії застосунків з СУБД. Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

SQL Server підтримує дзеркалювання та кластеризацію баз даних. Кластер серверу SQL—це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Усі сервери мають одне віртуальне ім'я, а дані розподіляються за IP-адресами машин кластеру протягом робочого циклу. Також у разі відмови або збою на одному з серверів кластеру доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями:

Знімок: Виконується «знімок» бази даних, який сервер відправляє одержувачам.

Історія змін: Всі зміни бази даних безперервно передаються користувачам.

Синхронізація з іншими серверами: Бази даних декількох серверів синхронізуються між собою. Зміни усіх баз даних відбуваються незалежно на кожному сервері, а під час синхронізації відбувається звірка даних. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних.

SQL Server 2005 має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server.

**2. SQL Server Management Studio (SSMS)** — утиліта із Microsoft SQL Server 2005 и більш пізніших версій для конфігурування, управління і адміністрування всіх компонентів Microsoft SQL Server. Утиліта включає скриптовий редактор і графічну программу, яка працює з об’єктами и налаштуваннями сервера.

Головним інструментом SQL Server Management Studio являється Object Explorer, який дозволяє користувачу переглядати, вилучати об’єкти сервера, а також повністю ними керувати.

Ще є SQL Server Management Studio Express для Express версії сервера, яка являється безкоштовною. Однак в ні немає підтримки ряду компонентів (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

Починаючи з версії 16.5.3 пакет SSMS виділений в окремий оновлюваний продукт. Версія доступна на даний момент SSMS 18.0.18118.0 (general availability) підтримує MS SQL server починаючи з версії 2008 по 2019 (preview)

Підсумовуючи все вище сказане можливо зробити висновки що найкращим вибором для мене було середовище розробки як Visual Studio Code в ньому самий широкий вибір інструментарію на відміну від MonoDevelop. Тому для мене вибір Visual Studio Code був єдиновірним. На даний момент для розробки на WEB платформі застосовується різні мови програмування. Проте я обрав такі мови як C#, JavaScript, HTML5, CSS3 так як це одні з найпопулярніших мов, до того ж я звик розробляти додатки саме на них. SQL Server на даний момент займає лідируючу позицію з баз даних і виразних конкурентів в неї нема тому і вибір впав на неї.

2.4 Структура даних

В процесі розробки даного програмного засобу я використав базу даних SQL Server в якій було створено такі таблиці:

1. Users — дана таблиця містить відомості про вірну відповідь на запитання;
2. Vocabluarys — таблиця в якій знаходяться категорії які є в програмі;

Таблиця Users містить такі поля як:

* Id — порядковий номер;
* FirstName — ім’я користувача ;
* LastName — прізвище користувача;
* Username — логін користувача;
* PasswordHash — пароль користувача який пройшов процедуру хешування;
* PasswordSalt — модифікатор який передається з хешем паролю хеш-функції для декодування паролю;

Таблиця Answer містить такі поля як:

* Id — порядковий номер;
* word — слово з знаками наголосу;
* wordKey — слово без форматування, необхідне для пошуку сортування і т.п.;
* designation — опис слова;
* table — назва таблиці в якій розміщено дане слово, якщо така є;

В процесі розробки було сформовано представлення майбутніх таблиць для другої ітерації проекту, а саме:

1. Answer — таблиця в якій містяться запитання;
2. Completion — таблиця для зберігання вірних відповідей;
3. Statistics — статистика по курсу;

Таблиця Answer містить такі поля як:

* Id — порядковий номер;
* text — додатковий текст до питання ;
* answerCorrect — номер вірної відповіді на питання;
* answerVariant — варіанти відповідей;
* answerType — тип питання;

Таблиця Completion містить такі поля як:

* Id — порядковий номер;
* stage — стадія проходження тесту;
* lastCompleted — останнє питання на яке дано відповідь;
* user\_Id — порядковий номер користувача;
* answer\_Id — номер питання

Таблиця Statistics містить такі поля як:

* Id — порядковий номер;
* succeeded — кількість виконаних запитань;
* user\_Id — порядковий номер користувача;
* answer\_Id — номер запитання

2.5 Особливості реалізації

Процес реалізації програмного продукту включає в себе такі етапи як

* Створення інтерфейсу додатку
* Написання коду який буде працювати з логікою проекту, і буде створювати при необхідності запити до бек-енд частини і буде опрацьовувати результати
* Написання бек-енд частини нашого додатку який напряму взаємодіє з базою даних, в той час як з фронт-енд частиною він буде взаємодіяти через API
* Створення та реалізація бази даних

Створення інтерфейсу додатку складається з того що ми використовуючи порівняно новий JavaScript фреймворк Vue.js та його модуль Vue CLI генеруємо новий проект, який і стане стартовою точкою в створенні інтерфейсу нашого додатку, після чого використовуючи мову розмітки HTML створюємо розмітку нашого додатку, а з допомогою мови каскадних стилів CSS створюємо візуальне оформлення наших компонентів.



Рисунок 2.4.1 – дизайн першої ітерації нашого додатку

Представлення у вигляді HTML коду

|  |
| --- |
| <div class="container-fluid p-0 app-root" v-if="loaded">  <div class="app-root-header">  <Header></Header>  </div>  <SubMenu v-if="status && status.loggedIn" :show="show" @changeState="show = !show"></SubMenu>  <!-- <PulseButton v-if="status && status.loggedIn" @changeState="show = !show"></PulseButton> -->  <div @click="clearAlert" v-if="alert && alert.message" :class="`alert ${alert.type}`"><span>{{alert.message}}</span></div>  <div class="app-root-container my-2">  <router-view></router-view>  </div>  <Footer></Footer>  </div>  <div class="row justify-content-md-center align-items-center h-100 m-0" v-else>  <div class="col-2">  <CustomLoader :LoaderWidth="220" :LoaderColor="'#FFDD00'"/>  </div>  </div> |

Рисунок 2.4.2 – HTML код головного компонента

Представлення у вигляді СSS коду

|  |
| --- |
| body {  height: 100vh;  background-color: $myBgColor !important;  font-family: $font-family-sans-serif;  input {  font-family: $font-family-sans-serif;  }  button.btn {  background-color: #007dcc;  font-weight: 500;  letter-spacing: 2px;  border-color: none;  }  button.btn:hover {  background-color: #006eb3;  border-color: none;  }  button.btn:active {  background-color: #006eb3 !important;  }  button.btn:focus {  box-shadow: none !important;  }  button.btn:active:focus {  box-shadow: none !important;  }  }  ::-webkit-scrollbar {  width: 9px;  }  /\* Track \*/  ::-webkit-scrollbar-track {  -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);  -webkit-border-radius: 10px;  border-radius: 10px;  }  /\* Handle \*/  ::-webkit-scrollbar-thumb {  -webkit-border-radius: 10px;  border-radius: 10px;  background: $scrollBarColor;  -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.5);  }  ::-webkit-scrollbar-thumb:window-inactive {  background: $scrollBarInactiveColor;  }  .app-root {  &-header {  height: 73px;  }  } |

Рисунок 2.4.3 – CSS код головного компонента

Компоненти фронт-енд частини:

* SubMenu – компонент бокового меню, в який передається атрибут “show” і в залежності від нього меню відображається, плюс в самому меню іде перевірка url і в залежності від активної сторінки відображається ті чи інші елементи меню.

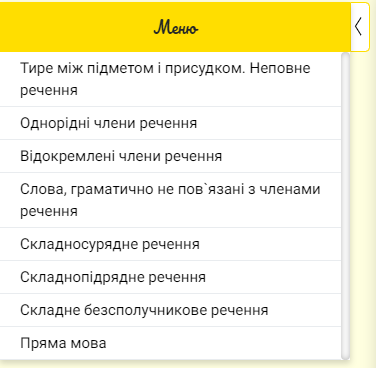


Рисунок 2.4.4 – Візуальний вигляд компонента SubMenu

* Header – компонент шапки веб-додатку, в якому міститься меню та перехід на форму логіна та реєстрації

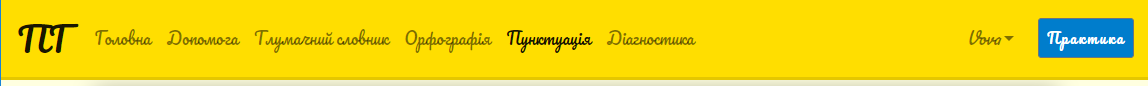


Рисунок 2.4.5 – Візуальний вигляд компонента Header

* Footer – нижній блок веб-додатку в якому містяться пости спонсорів та ссилки на ресурси



Рисунок 2.4.6 – Візуальний вигляд компонента Footer

* Table0 … Table28 – компоненти таблиць в них відображена головна інформація по правилам української мови
* CustomLoader – компонент який використовується під час завантаження системи, чи під час очікування отримання інформації із бек-енд частини

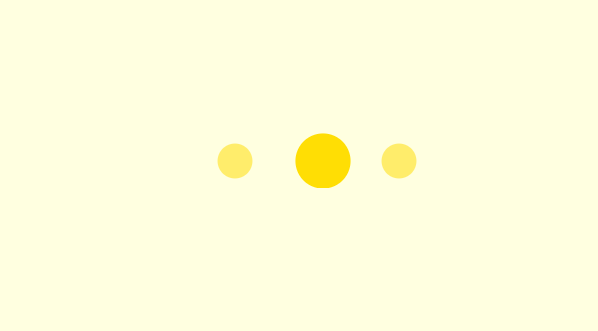


Рисунок 2.4.7 – Візуальний вигляд компонента CustomLoader

* ModalVocabluary – компонент модального вікна в якому відображається інформація про тлумачне значення слова, в нього передається слово та його значення.

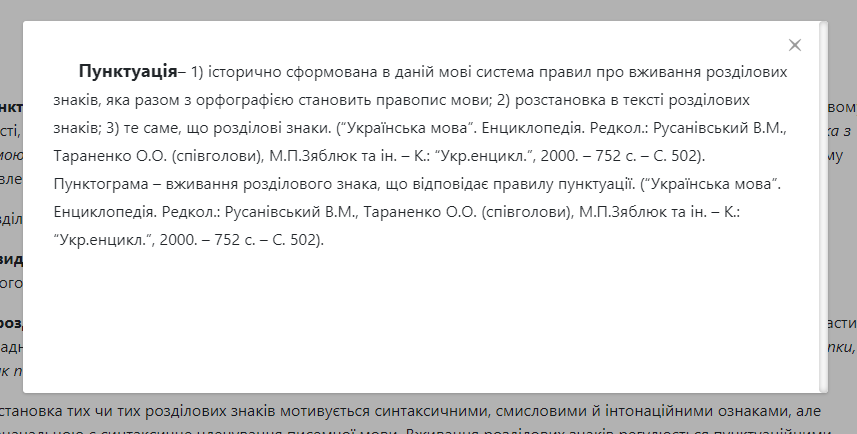


Рисунок 2.4.8 – Візуальний вигляд компонента ModalVocabluary

Сторінки вронт-енд частини:

* About – сторінка на якій відображатиметься інформація про проект
* ExplanatoryDictionary – сторінка на якій відображатимуться всі слова з їх тлумачними визначеннями

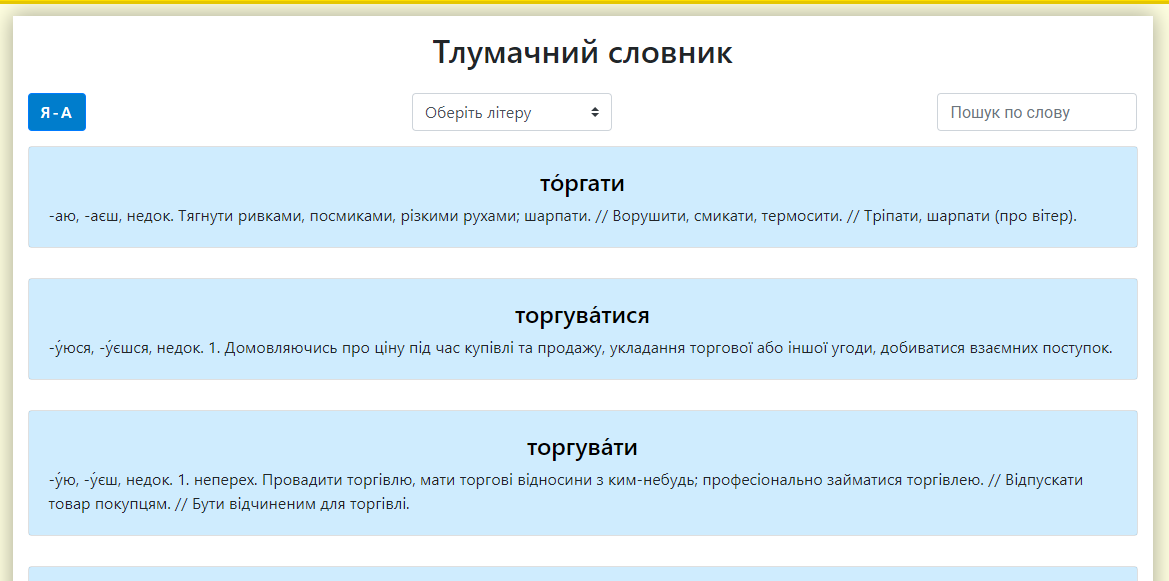


Рисунок 2.4.9 – Візуальний вигляд компонента ModalVocabluary

* Home – головна сторінка веб-додатку
* Login – сторінка входу в веб додаток
* NotFound – сторінка яка виводиться коли по заданому url не має результатів
* Orthography – сторінка із даними про орфографію
* Punctuation – сторінка із даними про пунктуацію
* Register – сторінка реєстрації нового користувача

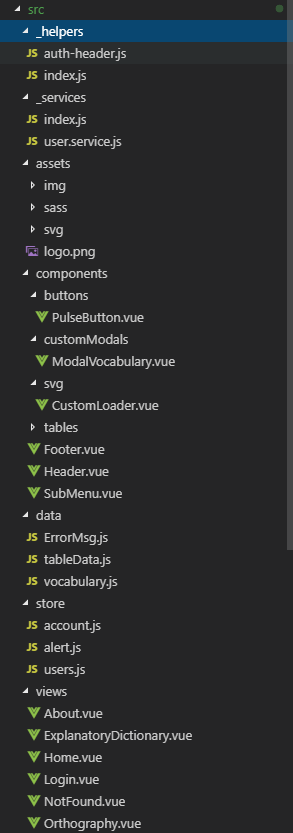
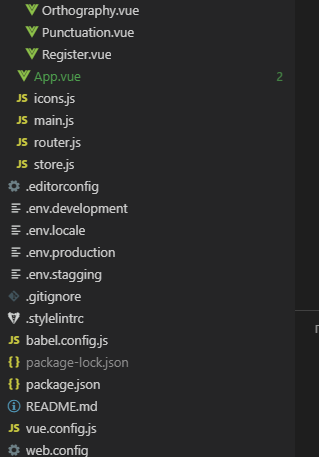
 

Рисунок 2.4.10 – Загальна структура фронт-енд частини проекту

Під час створення бек-енд частини було застосовано ASP .NET Core 2.1, на даній ітерації на бек-енд частині лежить реєстрація та авторизація користувачів, хешування паролів та запис результатів в базу даних.

|  |
| --- |
| private static void CreatePasswordHash(string password, out byte[] passwordHash, out byte[] passwordSalt)  {  if (password == null) throw new ArgumentNullException("password");  if (string.IsNullOrWhiteSpace(password)) throw new ArgumentException("Value cannot be empty or whitespace only string.", "password");  using (var hmac = new System.Security.Cryptography.HMACSHA512())  {  passwordSalt = hmac.Key;  passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));  }  } |

Рисунок 2.4.11 – Код хешування паролю

|  |
| --- |
| private static bool VerifyPasswordHash(string password, byte[] storedHash, byte[] storedSalt)  {  if (password == null) throw new ArgumentNullException("password");  if (string.IsNullOrWhiteSpace(password)) throw new ArgumentException("Value cannot be empty or whitespace only string.", "password");  if (storedHash.Length != 64) throw new ArgumentException("Invalid length of password hash (64 bytes expected).", "passwordHash");  if (storedSalt.Length != 128) throw new ArgumentException("Invalid length of password salt (128 bytes expected).", "passwordHash");  using (var hmac = new System.Security.Cryptography.HMACSHA512(storedSalt))  {  var computedHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));  for (int i = 0; i < computedHash.Length; i++)  {  if (computedHash[i] != storedHash[i]) return false;  }  }  return true;  } |

Рисунок 2.4.12 – Код перевірки введеного паролю

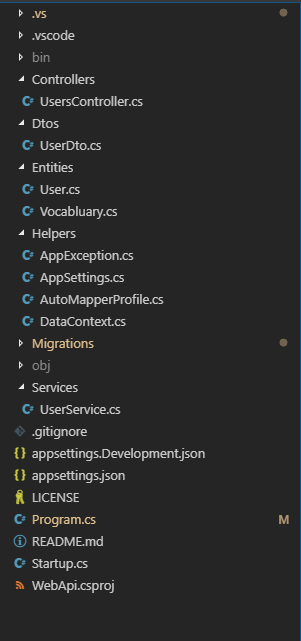


Рисунок 2.4.13 – загальна структура бек-енд частини

2.6 Особливості застосування

До особливостей даного програмного забезпечення можливо віднести:

* Використання декілька підходів для тестування користувача.
* Поступове навчання користувача, так як користувач отримує доступ до наступного тесту та курсу тільки після проходження попереднього етапу
* Можливість попрактикуватися в будь якій категорії курсу в режимі практики
* Статистика, яка відображає проходження курсу
* Налаштування профілю користувача
* Ще не менш важлива особливість це те що додаток оптимізовано до застосування в будь якому середовищі не залежно від операційної системи та типу пристрою, єдине що потрібно це браузер та інтернет.

Виходячи з особливостей, можна зробити висновок що даний додаток буде дуже зручний для використання всіх користувачів які ставлять собі мету вивчити досконало українську мову, знайти для себе щось нове в українській мові чи навіть просто попрактикуватися, плюс в другій ітерації проекту буде додано можливість написання диктанту, що дасть можливість ще краще готуватися учням до диктантів.