Київський національний торговельно-економічний університет
Кафедра програмної інженерії та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення тестування знань англійської мови»

Студента 2м курсу, 5 групи, спеціальності 121«Інженерія програмного забезпечення», спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Деремешка Максима Михайловича

Науковий керівник кандидат технічних наук, професор _____ підпис керівника — Пашорін Валерій Іванович

Гарант освітньої програми доктор технічних наук, професор _____ підпис гаранта — Криворучко Олена Володимирівна

КИЇВ – 2019

Київський національний торговельно-економічний університет

Факультет ФОАІС Кафедра «Програмної інженерії та кібербезпеки»

Спеціальність 121 «Інженерія програмного забезпечення»

Спеціалізація «Інженерія програмного забезпечення»

Затверджую

Зав. Кафедри

_____ Криворучко О.В.

«___» _____2019р.

Завдання

на випускну кваліфікаційну роботу студентові

Деремешку Максиму Михайловичу

1. Тема випускної кваліфікаційної роботи «Розробка програмного забезпечення тестування знань англійської мови»

   Затверджена наказом ректора від *«07» листопада 2018 р. №4183*

2. Строк здачі студентом закінченої роботи _____

3. Цільова установка та вихідні дані до роботи (проекту)

Мета роботи – розробка програмного забезпечення тестування знань англійської мови.

Об'єкт дослідження - тестування знань англійської мови.

Предмет дослідження - методи та алгоритми створення програмного забезпечення тестування знань англійської мови.

4. Перелик графічного матеріалу *10 рисунків*.

5. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ТЕСТУВАННЯ ЗНАНЬ АНГЛІЙСЬКОЇ МОВИ

1.1. Доцільність знання англійської мови

1.2. Дослідження процесу тестування та його форми

1.3. Висновки до розділу 1

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТЕСТУВАННЯ ЗНАНЬ АНГЛІЙСЬКОЇ МОВИ

2.1. Постановка задачі

2.2. Проектування модуля тестування

2.3. Висновки до розділу 2

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТЕСТУВАННЯ ЗНАНЬ АНГЛІЙСЬКОЇ МОВИ

3.1. Основні відомості про HTML, CSS, JavaScript, JQuery та AJAX

3.2. Процес розробки програмного забезпечення

3.3. Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1.	*Вибір теми випускної кваліфікаційної роботи*	*21.09.2018*	
2.	*Розробка та затвердження завдання на проект магістра (Наказ №4183 від 07.11.18)*	*07.11.2018*	
3.	*Вступ та перелік літературних джерел*	*28.02.2019*	
4.	*Розробка технічного завдання*	*22.03.2019*	
5.	*Розділ 1.* (*Назва розділу 1*)	*18.04.2019*	
6.	*Розділ 2.* (*Назва розділу 2*)	*24.05.2019*	
7.	*Розділ 3.* (*Назва розділу 3*)	*20.09.2019*	
8.	*Розробка програми та методики тестування*	*18.10.2019*	
9.	*Написання наукової статті*	*27.09.2019*	
10.	*Керівництво користувача*	*23.10.2019*	
11.	*Висновки та пропозиції*	*01.11.2019*	
12.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	*13.11.2019*	
13.	*Підготовка автореферату та презентації доповіді*	*14.11.2019*	
14.	*Попередній захист випускної кваліфікаційної роботи*	*14.11.2019 – 15.11.2019*	
15.	*Здача зброшурованої випускної кваліфікаційної роботи*	*25.11.2019*	
16.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*26.11.2019*	
17.	*Підготовка до публічного захисту випускної кваліфікаційної роботи*	*02.12.2019 – 04.12.2019*	

7. Дата видачі завдання «**26**» **вересня 2018 р**.

8. Науковий керівник випускної кваліфікаційної роботи

_____

(прізвище, ініціали, підпис)

9. Гарант освітньої програми _____

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент_____

(прізвище, ініціали, підпис)

11. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Науковий керівник випускної кваліфікаційної роботи____________________

(підпис, дата)

Відмітка про попередній захист ____

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента ________________________________

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми ____

(прізвище, ініціали, підпис)

Завідувач кафедри ____

(підпис, прізвище, ініціали)

«_____»_________________20 ___ р.

Аннотація

Актуалність даної роботи полягає у тому, то англійська - це одна з найбільш використовуваних мов світу. Якщо враховувати усіх, для кого англійська стала другою мовою, виходить близько 1 мільярда англомовних жителів світу. Крім того, англійська є офіційною мовою в 67 країнах світу, і ще є 27 країн, де англійська є другою офіційною мовою. Щоб сам процес тестування був зручним для користувачів та максимально об’єктивним, доцільно використовувати саме методи та засоби web-розробки, спираючись на рекомендації правильного формату тестування. Сучасне суспільство має доступ до мережі Інтернет майже завжди та майже в буд-якій точці світу, тому мережева сторінка не матиме проблем з доступом.

Метою роботи є створення інтерфейсу користувача та модуля тестування, які дадуть змогу користувачеві, використовуючи функціонал отримувати можливість тестувати свої знання англійської мови.

Annotation

The relevance of this work is that English is one of the most used languages in the world. If you consider everyone for whom English has become a second language, there are about 1 billion English-speaking inhabitants of the world. In addition, English is the official language in 67 countries, and there are still 27 countries where English is the second official language. In order for the testing process to be user-friendly and as objective as possible, it is advisable to use web development methods and tools based on the recommendations of the correct testing format. Modern society has access to the Internet almost always and almost anywhere in the world, so the web page will not have access problems.

The purpose of the work is to create a user interface and a testing module that will allow the user to use the functionality to test their English skills.

Київський національний торговельно-економічний університет

Програмне забезпечення тестування знань англійської мови

ТЕХНІЧНЕ ЗАВДАННЯ

на 2 аркушах

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Розробка програмного забезпечення тестування знань англійської мови.

Галузь застосування: Інформаційні технології, заклади освіти, підприємства, повсякденне життя.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на Випускну кваліфікаційну роботу, затверджене кафедрою програмної інженерії та кібербезпеки.

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмне забезпечення тестування знать англійської мови – це елемент для перевірки рівня знань англійської мови будь якого користувача. За допомогою даного засобу вчити та засвоювати мову можна значно швидше, а це в свою чергу тягне за собою інтелектуальний розвиток суспільства.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

1. Інформаційна система повинна забезпечувати доступ до тестів для усіх авторизованих користувачів.
2. Інтерфейс програмного продукту має бути інтуїтивно зрозумілий.
3. Інформаційна система повинна безпомилково передавати дані стосовно результатів тесту, та повідомляти користувачів про вірні або невірні відповіді.
4. У додатку повинно бути вірно складені тести, з правильно вказаними питаннями та відповідями.

5. ВИМОГИ ДО ПРОГРАНМОГО ЗАБЕЗПЕЧЕННЯ

Технології що будуть застосовуватися при побудові:

- WebStorm IDE;
- HTML;
- CSS;
- JavaScript;
- JQuery;
- Ajax;

- Мережевий протокол TCP/IP;

6. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання роботи повинна бути розроблена наступна документація:

1) Технічне завдання;
2) Керівництво користувача;
3) Програма та методика тестування.

7. ЕТАПИ ПРОЕКТУВАННЯ:

1) Аналаз моделі за тематикою роботи;
2) Розробка та узгодження технічного завдання;
3) Розробка моделі функціонування;
4) Розробка структури програмного забезпечення;
5) Програмна реалізація інформаційної системи;
6) Тестування інформаційної системи;
7) Підготовка матеріалів архітектурної частини роботи;
8) Підготовка матеріалів текстової частини роботи;
9) Підготовка матеріалів графічної частини роботи.

ЗМІСТ

ВСТУП

Розробка програмного забезпечення тестування знань англійської мови.- Деремешко М.М. 2 к 5м гр.

Актуальність: англійська - це одна з найбільш використовуваних мов світу. Якщо враховувати усіх, для кого англійська стала другою мовою, виходить близько 1 мільярда англомовних жителів світу. Крім того, англійська є офіційною мовою в 67 країнах світу, і ще є 27 країн, де англійська є другою офіційною мовою.

Щоб сам процес тестування був зручним для користувачів та максимально об’єктивним, доцільно використовувати саме методи та засоби web-розробки, спираючись на рекомендації правильного формату тестування. Сучасне суспільство має доступ до мережі Інтернет майже завжди та майже в буд-якій точці світу, тому мережева сторінка не матиме проблем з доступом. Крім того, далеко не кожен користувач має бажання завантажувати та встановлювати на власному комп’ютері невідоме програмне забезпечення.

Об'єкт дослідження: тестування знань англійської мови.

Предмет дослідження: розробка можливої моделі програмного забезпечення тестування.

Мета роботи: розробити модуль тестування знань англійської мови та зручний інтерфейс користувача.

Задачі дослідження: розробити інтерфейс користувача та модуль тестування, які дають змогу користувачеві використовувати функціонал програмного забезпечення та отримувати результати тестування в зручному вигляді.

Методи і технології розробки: аналіз алгоритмів пошуку, веб-програмування, техногологія front-end розробки, візуальне і об'єктно-орієнтоване програмування,

порівняння, спостереження, побудова програмного продукту на основі проаналізованих данних.

Результат роботи: Створення програмного забезпечення тестування знань англійської мови.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ ТЕСТУВАННЯ ЗНАНЬ АНГЛІЙСЬКОЇ МОВИ

1.1. Доцільність знання англійської мови

В першу чергу, знання англійської мови відкриває нові кар'єрні можливості. В даний час ринок праці став глобальним - багатьом компаніям потрібні співробітники, здатні спілкуватися з партнерами і клієнтами з різних країн світу. І, як правило, це означає, що працівникам потрібно знати англійську мову. На глобальному ринку праці є і такі вакансії, які створені спеціально для білінгвів. Знаючи англійську мову, можна стати перекладачем, учителем іноземної мови або навіть фахівцем з маркетингу великої компанії.

Сучасне суспільство дуже часто використовує електронну пошту, щоб спілкуватися один з одним. Крім того, електронна пошта є ще й основним каналом для спілкування ділових партнерів. Уміння писати електронні листи англійською мовою буде дуже цінним навиком для співробітника будь-якої компанії.

Для багатьох інших людей постає питання у тому, що потрібно мати конкретне підтвердження знань певного рівня іноземної мови, для подальшого навчання. Тому зараз існує багато тестів на зразок TOEFL (Test of English as a Foreign Language), який є одним з найпоширеніших тестів на знання англійської мови, або IELTS (International English Language Testing System) і кембріджські іспити. Багатьом людям у сучасних реаліях необхідним є саме такий підхід, щоб мати змогу вчитися у вищих навчальних закладах, де викладання ведеться англійською мовою, чи переїхати в англомовних країну, щоб працювати там за студентською візою. Крім того, навіть якщо потрібно здавати тест заради конкретної мети підготовка до здачі тесту на знання англійської мови допоможе поліпшити мовні навички.

Наступною причиною необхідності знання англійської мови являється власне мережа Інтернет тому, що саме англійська мова найчастіше використовується в Інтернеті: її використовують більше 1 мільярда користувачів. Вміння читати англійською і розуміти прочитане відкриває безліч онлайн-ресурсів та можливостей: читання новин в Інтернеті, коментування відео на англійській мові, брати участь в обговореннях на форумах, використовувати будь-які види соціальних мереж та ін. Багато людей і організацій проводять дослідження, займаються просуванням і пошуком нових ділових можливостей в мережі і для їхнього успіху знання англійської мови стане вирішальним фактором.

1.2. Дослідження процесу тестування та його форми

Для визначення рівня сформованості знань і вмінь з навчальної дисципліни користуються методом тестів. Виокремлюють тести відкритої форми (із вільно конструйованими відповідями) і тести закритої форми (із запропонованими відповідями).

Тести відкритої форми передбачають короткі однозначні відповіді, які ґрунтуються переважно на відтворенні вивченого матеріалу, або складні (комплексні) відповіді, які потребують розвинутого логічного мислення, вміння аналізувати.

Тести закритої форми передбачають вибір відповіді з певної кількості варіантів. Серед таких тестів виокремлюють тест-альтернативу, тест-відповідність:

Тест-альтернатива вимагає вибору однієї з двох запропонованих відповідей. Застосовують його під час контролю таких показників засвоєння, як уміння визначати використання фактів, законів, підводити під поняття, встановлювати причину якогось явища. Недолік цього виду тесту полягає в тому, що він не дає змоги вільно конструювати, формулювати відповідь. Його перевагою є те, що він допомагає швидше орієнтуватися в матеріалі, знаходити спільне та відмінне у явищах, легше класифікувати конкретні явища за певними видами.

Тест-відповідність, як правило, складається з двох частин, між якими слід встановити відповідність. Застосовують його для виявлення таких результатів засвоєння, як уміння визначати використання речовин, апаратів, процесів, встановлювати зв'язок між абстрактним і конкретним поняттями, класифікувати їх тощо. Перевага тестів-відповідностей полягає у компактній формі завдання, завдяки якій протягом короткого часу вдається перевірити засвоєння великого обсягу навчального матеріалу. Недоліком є обмеженість безпосередньої мети контролю і ускладнення при доборі матеріалу.

Такий вид контролю дає змогу ефективніше використовувати час, ставить перед усіма студентами однакові вимоги, допомагає уникати надмірних хвилювань. Тестова перевірка унеможливлює випадковість в оцінюванні знань, стимулює студентів до самоконтролю. Однак тест може виявити лише знання фактів, він заохочує до механічного запам'ятовування, а не до роботи думки.

Серед головних переваг використання тестового контролю знань можна виділити такі :

- можливість застосування як засобу усіх видів контролю, а саме базового та початкового, поточного та тематичного, рубіжного та залікового, підсумкового та екзаменаційного, а також самоконтролю;
- можливість детальної перевірки рівня засвоєння кожного змістового модуля;
- наявність чіткої однозначної відповіді;
- економія навчального часу при здійсненні поточного контролю знань та об’єктивність оцінювання результатів навчання;
- мінімізація емоційного впливу.

Крім цього, традиційно до переваг тестування відносяться :

- індивідуальний характер контролю, можливість здійснення контролю за роботою кожного, його особистою навчальною діяльністю;
- можливість регулярного систематичного проведення тестового контролю на всіх етапах процесу навчання;

− можливість поєднання його з іншими традиційними формами педагогічного контролю;

− можливість проведення комп'ютеризованого у локальній мережі та паперового варіанта тестування;

− урахування індивідуальних особливостей, що потребує застосування відповідно до цих особливостей різної методики розробки тесту й тестових завдань.

Розробкою та застосуванням мовних і мовленнєвих тестів займається лінгводидактичне тестування, під яким розуміють „підготовлений згідно з певними вимогами комплекс завдань, які пройшли попереднє випробування з метою визначення показників його якості і дозволяють виявити в учасників тесту ступінь їх мовної (лінгвістичної) та/чи мовленнєвої (комунікативної) компетенції, і результати якого піддаються певному оцінюванню за заздалегідь встановленими критеріями".

Основними показниками якості лінгводидактичного тесту є:

- Валідність;
- Надійність;
- Диференційна здатність;
- Практичність та економічність.

Валідність - характеристика тесту, яка показує, що саме вимірює тест і наскільки ефективно він це вимірює. Валідність тесту означає його придатність для визначення рівня володіння певними іншомовними мовленнєвими навичками і вміннями.

Надійність - це необхідна умова валідності тесту. Надійність тесту визначається стабільністю його функції як інструмента вимірювання. Надійний тест дає приблизно однакові результати при повторному застосуванні.

Диференційна здатність - характеристика тесту, яка вказує на здатність даного тесту виявляти встигаючих і невстигаючих тестованих, тобто з достатнім і недостатнім рівнем володіння іншомовними навичками і вміннями.

Практичність - характеристика тесту, яка визначає:

а) доступність інструкцій тесту і змісту тестових завдань для розуміння тих, хто виконує тест;

б) простота організації проведення тестування в різних умовах;

в) простота перевірки відповідей і визначення результатів та оцінки.

Економічність - характеристика тесту, яка передбачає мінімальні витрати часу, зусиль і коштів на підготовку тесту від планування до видання.

1.3. Висновки до розділу 1

Проаналізовано доцільність знання англійської мови в сучасному світі, в тому числі для амбіційної та енергічної частки населення, а саме молоді. Визначено, що знання англійської мови на необхідному рівні дає можливість вести комунікацію зі світом, покращувати свої професійні навички, можливість вирушити у подорож, або навіть просто потренувати свій мозок.

Досліджено процес та форми тестування. Завдяки можливості використання процесу тестування як апарату перевірки знань можна отримати такі переваги:

- можливість застосування як засобу усіх видів контролю, а саме базового та початкового, поточного та тематичного, рубіжного та залікового, підсумкового та екзаменаційного, а також самоконтролю;
- можливість детальної перевірки рівня засвоєння кожного змістового модуля;
- наявність чіткої однозначної відповіді;
- економія навчального часу при здійсненні поточного контролю знань та об'єктивність оцінювання результатів навчання;
- мінімізація емоційного впливу;
- індивідуальний характер контролю, можливість здійснення контролю за роботою кожного, його особистою навчальною діяльністю;
- можливість регулярного систематичного проведення тестового контролю на всіх етапах процесу навчання;
- можливість поєднання його з іншими традиційними формами педагогічного контролю;

− можливість проведення комп'ютеризованого у локальній мережі та паперового варіанта тестування;

− урахування індивідуальних особливостей, що потребує застосування відповідно до цих особливостей різної методики розробки тесту й тестових завдань.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТЕСТУВАННЯ ЗНАНЬ АНГЛІЙСЬКОЇ МОВИ

2.1. Постановка задачі

Для проведення процесу тестування знань англійської мови необхідно створити форму, яка міститиме саме питання та варіанти відповіді. Після вибору користувачем необхідного варіанту(ів) для переходу до наступного питання повинна бути кнопка, яка перенаправлятиме користувача до наступного питання. Після проходження усіх питань тесту форма повинна містити кнопку перенаправлення на сторінку з даними про кількість правильних відповідей. Дані умови є необхідним мінімумом даного програмного продукту. Також можна розробити за необхідності потрібні додаткові модулі, наприклад, таймер для обмеження часу на проходження тесту, або форму авторизації для індивідуалізації користувачів.

2.2. Проектування модуля тестування

Щоб сам процес тестування був зручним для користувачів та максимально об’єктивним, доцільно використовувати саме методи та засоби web-розробки, спираючись на рекомендації правильного формату тестування. Сучасне суспільство має доступ до мережі Інтернет майже завжди та майже в буд-якій точці світу, тому мережева сторінка не матиме проблем з доступом. Крім того, далеко не кожен користувач має бажання завантажувати та встановлювати на власному комп’ютері невідоме програмне забезпечення.

Для навчальних організацій також не повинно бути проблемою використання web-ресурсу з метою проведення тестування знать англійської мови, а для запобігання появи необ’єктивності тестування є можливість створити ієрархію користувачів з різними правами доступу. Так, наприклад, викладач зможе сам стежити за наповненням тестів та збором результатів, при цьому

студенти матимуть лише можливість відповідати на тестові питання.

Також сучасні можливості розробки дозволяють створити продукт, який можна використовувати на будь-яких сучасних пристроях та з використанням будь-яких операційних систем та програмних продуктів, що спрощує весь процес та покращує зручність та легкість при використанні.

Для розробки доцільно використовувати мову програмування JavaScript доповнюючи прості елементи дизайну та інтерфейсу мовою розмітки HTML\CSS.

Мова розмітки гіпертекстових сторінок (HTML - Hypertext Markup Language) представляє собою мову, розроблену спеціально для створення Web-документів. Він визначає синтаксис і розміщення спеціальних інструкцій (тегів), які не виводяться на екран, але вказують браузеру, як відображати вміст документа. Він також використовується для створення посилань на інші документи, локальні або мережні, наприклад, що знаходяться в мережі Інтернет.

На практиці на стандарт HTML великий вплив надає наявність тегів, запропонованих і підтримуваних найвідомішими браузерами. Ці теги в даний момент можуть, як входити, так і не входити до складу діючої специфікації HTML.

Інформацію про теги HTML Compendium (короткий посібник по HTML) створено Ron Woodall. Компендіум містить список тегів та їх атрибутів в алфавітному порядку, а також оновлену інформацію про підтримку кожного з них з боку браузерів.

Документи HTML є звичайними текстовими ASCII-файлами. Це означає, що для їх створення можна використовувати будь-який текстовий редактор, навіть з мінімальними можливостями. Існують засоби редагування, розроблені спеціально для написання HTML. Вони дозволяють економити час, оскільки містять клавіші швидкого доступу для виконання повторюваних операцій, наприклад, завдання початкових установок документів, таблиць або просто застосування стилів до тексту.

Документ HTML містить текст (вміст сторінки) і вбудовані теги - інструкціями про структуру, зовнішній вигляд і функції вмісту. Документ HTML розділяється на дві основні частини: заголовок - head і тіло - body. Заголовок містить такі відомості про документ, як його назва і методична інформація, що описує вміст. У тілі знаходиться опис того, що має виводитися у вікні браузера.

Кожен тег складається з імені, за яким може слідувати список необов'язкових атрибутів, всі вони знаходяться всередині кутових дужок <>. Вміст дужок ніколи не виводиться у вікні браузера. Ім'я тега, як правило, представляє собою абревіатуру його функції, що полегшує його запам'ятовування. Атрибути є властивостями, які розширюють або уточнюють функцію тега.

Більшість тегів є контейнерами. Це означає, що у них є початковий (що відкриває або стартовий) і кінцевий (що закриває) теги. Текст, що знаходиться між тегами, виконуватиме усі присутні там інструкції.

Кінцевий тег має те ж ім'я, що і початковий, але перед ним стоїть слеш (/). Його можна розглядати як "вимикач" тега. Кінцевий тег ніколи не містить атрибутів. У деяких випадках кінцевий тег не обов'язковий, і браузер визначає кінець тега з контексту. Найчастіше опускають кінцевий тег <p> (абзац).

JavaScript же спочатку створювався для того, щоб надати динамічну інтерактивність web-сторінкам. Програми на цій мові називаються скриптами. У браузері вони підключаються безпосередньо до HTML і, як тільки завантажується сторінка - тут же виконуються.

Сучасний JavaScript - це «безпечна» мова програмування загального призначення, що не надає низькорівневих засобів роботи з пам'яттю, процесором, так як спочатку був орієнтований на браузери, в яких це не потрібно. Має такі особливості: повна інтеграція з HTML / CSS, прості речі робляться просто, підтримується всіма поширеними браузерами і включений за замовчуванням.

Структурно JavaScript можна представити у вигляді об'єднання трьох частин, що чітко різняться одна від одної : ядро (ECMAScript), об'єктна модель браузера (Browser Object Model або BOM) та об'єктна модель документа (Document Object Model або DOM). Об'єктну модель документа іноді розглядають

як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежного від мови інтерфейсу документа.

ECMAScript не є браузерною мовою і насправді в ній не визначаються методи введення і виведення інформації. Це скоріше основа для побудови скриптових мов. Специфікація ECMAScript описує типи даних, інструкції, ключові і зарезервовані слова, оператори, об'єкти, регулярні вирази, не обмежуючи авторів похідних мов від розширення їх новими складовими.

Об'єктна модель браузера - браузероспеціфічна частина мови, яка являється прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - керування вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом BOM. Об'єктна модель браузера на даний момент не стандартизована, проте специфікація знаходиться в розробці WHATWG та W3C.

Крім управління вікнами, в рамках об'єктної моделі браузера, браузерами зазвичай забезпечується підтримка наступних сутностей: керування фреймами, підтримка затримки у виконанні коду і зациклювання з затримкою, системні діалоги, управління адресою відкритої сторінки, управління інформацією про браузер, управління інформацією про параметри монітора, обмежене керування історією перегляду сторінок, підтримка роботи з HTTP cookie.

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів. Згідно DOM документом можна поставити у відповідність дерево об'єктів, які мають ряд властивостей, які дозволяють робити з ним різні маніпуляції: отримання вузлів, зміна вузлів, зміна зв'язків між вузлами, видалення вузлів.

Мета веб-сайту - створити зручну веб-платформу, що дозволить швидко, зручно та надійно тестувати знання англійської мови користувачів.

Веб-сайт призначений для:

1) Додавання тестових завдань;

2) Створення облікових записів користувачів;

3) Створення можливості проходження тестових завдань з декількома вірними відповідями;

4) Створення можливості проходження тестових завдань з однією вірною відповіддю;

5) інформування про результати проходження тесту.

Цільова аудиторія веб-сайту включає наступні групи людей:

1) адміністратор;

2) користувач;

Більшість сторінок в загальній структурі веб-сайту складається зі сторінок, які не мають жодного динамічного функціоналу і містять лише текстовий контент із деякими картинками чи можливістю завантажити файли в форматі .pdf, .jpg, .png, .xls, які редагуються із адміністративної частини відповідальними авторизованими користувачами.

Авторизація користувача містить поля «Логін», «Пароль». Подібний доступ суворо модерується і надається лише визначеним користувачам.

Доступ є індивідуальним та надається лише окремим експертам з адміністративних послуг, а публікація матеріалів відбувається тільки після того, як адміністратор веб-сайту підтверджує вихід публікації. Система управління контентом (адміністративна частина веб-сайту) повинна надавати можливість додавання, редагування та видалення вмісту статичних та динамічних сторінок.

Система управління контентом повинна мати стандартний для Windows, Unix або MacOS систем інтерфейс, який відповідає таким вимогам:

- реалізація в графічному віконному режимі;
- єдиний стиль оформлення;
- інтуїтивно зрозуміле, зручне призначення елементів інтерфейсу;
- відображення на екрані тільки тих можливостей, які доступні конкретному користувачу.

Внутрішня частина веб-сайту забезпечуватиме персоналізований вхід визначених осіб на веб-сайт та дозволятиме їм публікувати власні тексти в обраних розділах з відповідною перевіркою.

Архітектура веб-сайту повинна забезпечувати масштабованість і розширення системи. Додавання додаткового функціоналу повинне відбуватися за рахунок додавання додаткових модулів без суттєвої модернізації вже існуючих модулів. Архітектура веб-сайту має передбачати незалежність модулю системи відображення інформації від модулю зберігання та керування інформацією. Архітектура веб-сайту має передбачати незалежність реалізації системи від апаратної платформи і серверної операційної системи.

2.3. Висновки до розділу 2

Проведено постановку задачі, визначено усі необхідні вимоги до моделі програмного забезпечення тестування знань англійської мови. Було створено необхідну модель. Проаналізовано шляхи реалізації даної моделі, на основі даного аналізу було обрано найбільш слушні методи подальшої розробки. Було вирішено розробити модель веб-сторінки, що має за мету швидко, зручно та надійно тестувати знання англійської мови користувачів. Цільова аудиторія веб-сайту включає наступні групи людей:

1) адміністратор;

2) користувач.

Система управління контентом повинна мати стандартний для Windows, Unix або MacOS систем інтерфейс, який відповідає таким вимогам:

- реалізація в графічному віконному режимі;
- єдиний стиль оформлення;
- інтуїтивно зрозуміле, зручне призначення елементів інтерфейсу;
- відображення на екрані тільки тих можливостей, які доступні конкретному користувачу.

Архітектура веб-сайту має передбачати незалежність реалізації системи від апаратної платформи і серверної операційної системи.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТЕСТУВАННЯ ЗНАНЬ АНГЛІЙСЬКОЇ МОВИ

3.1. Основні відомості про HTML, CSS, JavaScript, JQuery та AJAX

HTML (Hypertext Markup Language — Мова гіпертекстової розмітки) — це мова опису структури сторінок документів, яка дозволяє звичайний текст форматувати в абзаци, заголовки, списки та інші структури, створювати посилання на інші сторінки. Це текстова мова, в якій інструкції з форматування, що називаються тегами, вбудовані в розділи документа, які містять конкретну інформацію. Теги повідомляють браузерам, як форматувати і представляти інформацію на екрані.

Мова гіпертекстової розмітки HTML була запропонована Тімом Бернерсом-Лі у 1989 як один з компонентів технології розробки розподіленої гіпертекстової системи World Wide Web. Ідея гіпертекстової інформаційної системи полягає у тому, що користувач має можливість переглядати документи (сторінки тексту) у найбільш зручному для себе порядку, а не послідовно, як це прийнято при читанні книг. Досягається це шляхом створення спеціального механізму пов`язуваня різних сторінок тексту за допомогою гіпертекстових посилань.

Мова HTML дозволяє визначити структуру електронного документа з поліграфічним рівнем оформлення. Результуючий документ може містити різноманітні елементи: ілюстрації, аудіо і відео фрагменти. Мова HTML включає розвинені засоби для визначення кількох рівнів заголовків, шрифтових виділень, різних груп об’єктів та багато інших можливостей.

Важливим чинником, який вплинув на розвиток мови HTML, став її вибір за основу для гіпертекстової бази даних звичайного текстового файла, який можна створювати у будь-якому текстовому редакторі на будь-якій апаратній платформі у середовищі будь-якої операційної системи.

Таким чином, гіпертекстова база даних у концепції WWW – це набір текстових файлів, розмічених мовою HTML, яка визначає форму представлення інформації (розмітка) і структуру зв'язків цих файлів (гіпертекстові посилання).

За основу моделі розмітки документів у HTML прийнята тегова модель. Тегова модель описує документ як сукупність контейнерів, кожен з яких починається і закінчується тегами. Тобто документ HTML є не чим іншим, як звичайним ASCII-файлом з доданими до нього керуючими HTML-кодами (тегами).

Теги HTML-документів в основному є простими і зрозумілими для використання, оскільки вони створені за допомогою загальновживаних слів англійської мови, зрозумілих скорочень і позначень.

HTML-тег складається з імені, за яким може слідувати необов'язковий список атрибутів тегу. Текст тегу вміщується у кутові дужки (<I>). Найпростіший варіант тегу – ім'я, вміщене у кутові дужки, наприклад, <HEAD>. Для більш складних тегів характерна наявність різних атрибутів, які можуть мати конкретні значення, визначені для видозмінення функцій тегу.

Атрибути тегу слідують за ім'ям і відділяються один від одного одним або кількома пропусками. Порядок запису атрибутів у тегу не має значення. Значення атрибута слідує за знаком рівняння, який стоїть після імені атрибута. Якщо значення атрибута – одне слово або число, його можна вказати безпосередньо після знаку рівняння, не виділяючи додатково. Решту значень необхідно вміщувати у одинарні або подвійні лапки, особливо якщо вони містять декілька розділених пропусками слів.

Найчастіше HTML-теги складаються з початкового і кінцевого компонентів, між якими розміщуються текст та інші елементи документа. Ім'я кінцевого тега ідентичне імені початкового тегу, але перед ім'ям ставиться коса риска (/) (наприклад, для тегу заголовка <TITLE> закриваючою парою буде </TITLE>). Кінцеві теги не містять атрибутів. При використанні вкладених тегів їх слід закривати, починаючи з останнього і рухаючись до першого.

Деякі HTML-теги не мають кінцевого компонента, оскільки є автономними елементами. Наприклад, тег зображення <IMG>, призначений для вставки зображення у документ, не має кінцевого компонента.

Для створення HTML-документа можна застосувати редактор ASCII (зокрема, Блокнот системи Windows). Такі редактори дозволяють вводити HTML-теги, не додаючи до створеного нічого додатково. Створення документа у такому редакторі дозволяє паралельно переглядати результат у програмі- браузері. Інший тип редакторів – візуальні HTML-редактори, наприклад, Microsoft FrontPage. Їх інтерфейс побудований за тим же принципом, що і інтерфейс текстового процесора, такого, як, наприклад, Word. Для роботи з візуальним редактором можна взагалі не володіти мовою HTML. Недоліком візуальних редакторів є те, що розмір створюваного ними HTML-документа у декілька разів більший, ніж документа, створеного звичайним Блокнотом системи Windows. В умовах низької пропускної здатності вітчизняних мереж цей недолік, який стосується швидкості завантаження сторінки (і, відповідно, вартості часу, який на це витрачається), є досить суттєвим недоліком (файл .htm, створений у WORD, в 4 - 9 разів більший, ніж файл аналогічного змісту, створений програмою Блокнот).

Структурні теги HTML-документа визначають початок і закінчення різних частин документа. Хоча багато браузерів правильно інтерпретують документ і без них, правила вимагають, щоб структурні теги були включені до документа. На початковому етапі вивчення мови HTML ці теги обов’язково повинні включатись до кожного із створюваних документів.

Позначення HTML-документа

<HTML>… < /HTML> - включають в себе всі інші теги HTML і весь інформаційний зміст документа.

< HTML> розташовується у першому рядку документа;

< /HTML> - у останньому рядку.

Заголовок документа

<HEAD>… </HEAD> - містить інформацію про документ.

<TITLE>… </TITLE> - назва документа, яка відображається в рядку заголовка Internet Explorer.

<BODY>…</BODY> - тіло документа містить весь текст з інформацією і всі теги HTML, які використовуються для форматування тексту.

Файл у форматі HTML:

```
<HTML>
<HEAD>
<TITLE>… </TITLE>
</HEAD>
<BODY>
…
</BODY>
</HTML>
```

JavaScript — це особлива мова програмування, який базується на об’єктному представленні браузера. Він потрібен для того, щоб надати сайту більше інтерактивності в порівнянні зі звичайним статичним HTML-документом. Наприклад, в інтерфейсі можна буде реалізувати мінливі малюнки, рухомий рядок з тексту та багато іншого! Відмінність JavaScript полягає в тому, що текст програми вбудовується в документ HTML і аналізується самим браузером. JavaScript — це мова програмування сценаріїв на веб-сторінках.

Знаючи, що таке JavaScript, багато користувачів все одно плутають цей термін з іншим поняттям — Java. Хоч мови і схожі за назвою, але мають вони різні значення. Основні відмінності полягають в складності і в кількості можливостей.

Реалізація JavaScript більш вільна у порівнянні з Java. Перетворення типів даних, наприклад, відбувається набагато простіше. Також програмісту не потрібно буде компілювати вихідний код програми на мові JavaScript, тобто JavaScript є інтерпретується мовою. Як це відбувається на JavaScript і на Java. У JavaScript програма обробляється рядок за рядком, та інформація про помилки видається після кожної прочитаної рядки, якщо вони є. В в Java компілятор видає

ці відомості після прочитання всього тексту програми. Зверніть увагу, що JavaScript не розглядається як заміна мови програмування Java. Краще за все перший використовувати як доповнення до другого.

За допомогою JavaScript створюються динамічні документи HTML. JavaScript пов’язує воєдино всі будівельні блоки програми, це як би засіб побудови фундаменту. JavaScript здійснює перевірку полів форм HTML до того, як вони передалися на сервер. Управління програмою на даній мові програмування йде через локальний введення інформації. Користувач має можливість бачити в окремих вікнах повідомлення-застереження, які виводяться за допомогою JavaScript. На JavaScript вплинули багато мов, при розробці була мета зробити мову схожою на Java, але при цьому легкою для використання непрограмістами. JavaScript має низку властивостей об'єктно-орієнтованої мови, але реалізоване в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними об'єктно-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, - функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання - що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript у порівнянні з мовою Сі має корінні відмінності:

- об'єкти, з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

У мові відсутні такі корисні речі, як:

- модульна система - JavaScript не надає можливості управляти залежностями та ізоляцією областей видимості;
- стандартна бібліотека - зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управління потоками вводу/виводу, базових типів для бінарних даних;

- стандартні інтерфейси до веб-серверів та баз даних;
- система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

Структурно JavaScript можна представити у вигляді об'єднання трьох частин, що чітко різняться одна від одної :

- ядро (ECMAScript);
- об'єктна модель браузера (Browser Object Model або BOM);
- об'єктна модель документа (Document Object Model або DOM).

Об'єктну модель документа іноді розглядають як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежного від мови інтерфейсу документа.

ECMAScript не є браузерною мовою і насправді в ній не визначаються методи введення і виведення інформації. Це скоріше основа для побудови скриптових мов. Специфікація ECMAScript описує типи даних, інструкції, ключові і зарезервовані слова, оператори, об'єкти, регулярні вирази, не обмежуючи авторів похідних мов від розширення їх новими складовими.

Об'єктна модель браузера - браузероспеціфічна частина мови, яка являється прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - керування вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом BOM. Об'єктна модель браузера на даний момент не стандартизована, проте специфікація знаходиться в розробці WHATWG та W3C. Крім управління вікнами, в рамках об'єктної моделі браузера, браузерами зазвичай забезпечується підтримка наступних сутностей:

- керування фреймами;
- підтримка затримки у виконанні коду і зациклювання з затримкою;
- системні діалоги;
- управління адресою відкритої сторінки;
- управління інформацією про браузер;
- управління інформацією про параметри монітора;

- обмежене керування історією перегляду сторінок;
- підтримка роботи з HTTP cookie.

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів. Згідно DOM документом можна поставити у відповідність дерево об'єктів, які мають ряд властивостей, які дозволяють робити з ним різні маніпуляції:

- отримання вузлів;
- зміна вузлів;
- зміна зв'язків між вузлами;
- видалення вузлів.

Щоб додати JavaScript-код на сторінку, можна використовувати теги <script></script>.

Фреймворк являє собою програмну платформу, яка служить засобом визначення структури програми. Використовується в розробці ПЗ, де необхідно звертатися до значних і малим фрагментів коду і компонентів, які і були об’єднані платформою, в даному випадку jQuery.

Бібліотека дозволяє звертатися абсолютно до будь-якого елементу DOM для зміни його вмісту, структури, параметрів і навіть оформлення.

Ясність у розумінні попереднього речення внесе визначення DOM – це універсальний багатоплатформовий (не залежить від операційної системи і використовуваного мови програмування) користувальницький інтерфейс, за допомогою якого програми і міні-програми (скрипти) здатні відкривати HTML і XML-файли для їх коригування.

При цьому правила, умови чи обмеження фактично відсутні, що відкриває перед юзером необмежені можливості по обробці і зміни вмісту HTML. Будь відомий документ представляється у вигляді ієрархічного дерева, в якому кожна гілка і її дочірні елементи – це атрибути, графічні або текстові об’єкти.

JQuery - швидка, невелика і багатофункціональна бібліотека JavaScript. Вона робить такі речі, як обхід і маніпулювання документами HTML, обробку подій, анімацію і Ajax набагато простіше з простим у використанні API, які

працюють у великій кількості браузерів. Завдяки поєднанню універсальності і розширюваності, jQuery змінив спосіб, яким мільйони людей пишуть JavaScript.

jQuery хороший не тільки тим, що забезпечує вибірку будь-якого елементу на сторінці, при цьому вибірку гнучку. Використовуючи jQuery ви отримуєте ще ряд плюсів в порівнянні з нативним JS. Ось деякі з них:

- проста робота з подіями;
- кроссбраузерность;
- зручна робота з AJAX (асинхронні запити до сервера);
- зручні методи для роботи з ефектами (приховування / поява елементів з додаванням візуальних ефектів).

На сьогоднішній день знання і робота з бібліотекою jQuery - це фактично стандарт для будь-якого веб-розробника. Без неї не обходиться практично жоден проект в мережі, оскільки jQuery реально спрощує написання коду на JavaScript.

AJAX - Asynchronous Javascript and XML. Насправді, AJAX не є новою технологією, так як і Javascript, і XML існують вже досить тривалий час, а AJAX - це синтез позначених технологій.

При використанні AJAX немає необхідності оновлювати кожен раз всю сторінку, так як оновлюється тільки її конкретна частина. Це набагато зручніше, так як не доводиться довго чекати, і економічніше, так як не всі мають безлімітним інтернетом. Правда в цьому випадку, розробнику необхідно стежити, щоб користувач був в курсі того, що відбувається на сторінці. Це можна реалізувати з використанням індикаторів завантаження, текстових повідомлень про те, що йде обмін даними з сервером. Необхідно також розуміти, що не всі браузери підтримують AJAX (старі версії браузерів і текстові браузери). Плюс Javascript може бути відключений користувачем. Тому, не слід зловживати використанням технології і вдаватися до альтернативних методів подання інформації.

Переваги AJAX:

- можливість створення зручного Web-інтерфейсу;
- активна взаємодія з користувачем;

- часткова перезавантаження сторінки, замість повної;

- зручність використання;

AJAX використовує два методи роботи з веб-сторінкою: зміна Web-сторінки без перезавантаження її, і динамічне звернення до сервера. Друге може здійснюватися кількома способами, зокрема, XMLHttpRequest, про що ми і будемо говорити, і використання техніки прихованого фрейма.

3.2. Процес розробки програмного забезпечення

Для розробки програмного продукту було використано інтегроване середовище розробки веб-додатків WebStorm (Рис. 3.1.). Програмне забезпечення JetBrains WebStorm являє собою інструмент для розробки web-сайтів і редагування HTML, CSS і JavaScript коду. Рішення забезпечує швидку навігацію по файлах і генерує повідомлення про виникаючі проблеми в коді в режимі реального часу. JetBrains WebStorm дозволяє додавати розмітку HTML-документів або елементів SQL безпосередньо в JavaScript. JetBrains WebStorm здійснює розгортання і синхронізацію проектів через протокол FTP.

Використовуючи можливості коду HTML / XHTML і XML, WebStorm забезпечує автоматичне завершення стилів, посилань, атрибутів і інших елементів коду. При роботі з CSS здійснюється завершення коду класів, HTML-номерів, ключових слів і т. д. WebStorm пропонує автоматичне рішення таких проблем, як вибір формату, властивостей, класів, посилань на файли і інших атрибутів CSS. Рішення дозволяє використовувати потужність інструменту Zen coding для верстки HTML, відображає дії тега на web-сторінці. Продукт WebStorm здійснює завершення коду JavaScript для ключових слів, лейблів, змінних, параметрів і функцій DOM і підтримує специфічні особливості популярних браузерів. Реалізовані в рішенні функції рефакторінга JavaScript дозволяють перетворювати структуру коду та файлів .js.

WebStorm забезпечує налагодження коду JavaScript і надає широкий діапазон можливостей: знаходження точки зупину в HTML і JavaScript, настройка параметрів точки зупину, тестування синтаксису коду в режимі реального часу і т. Д. Продукт підтримує платформи JQuery, YUI, Prototype, DoJo, MooTools,

Qooxdoo і Bindows. WebStorm передбачає інтегровану перевірку тексту на теги, послідовність коду, помилки в написанні і т. Д. WebStorm дозволяє редагувати файли і автоматично синхронізувати їх на вимогу при віддаленій роботі або зберіганні.

Продукт підтримує функцію контролю версій і попередніх варіантів коду і фіксує всі вироблені дії та зміни. Завдяки створенню історії в WebStorm можна відновлювати кодові вирази, блоки і навіть цілі файли.

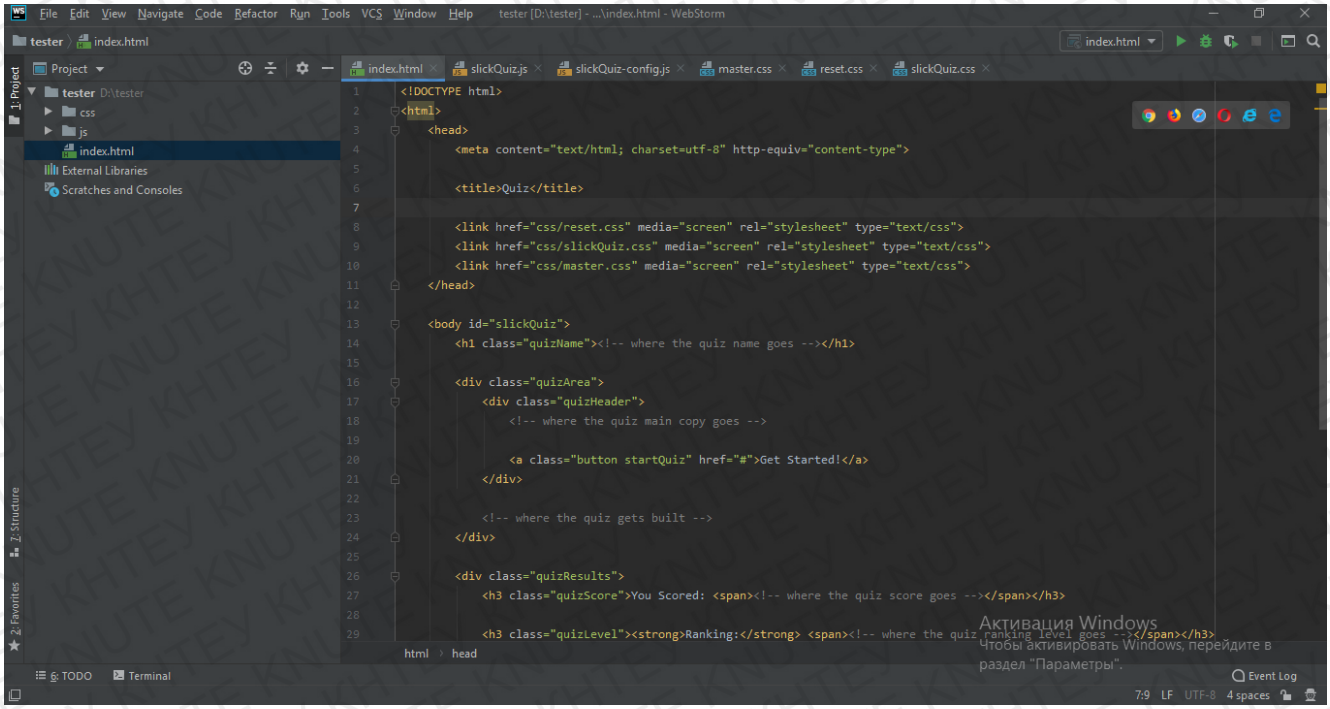

Рис. 3.1. Інтерфейс середовища розробки WebStorm

Початком розробки є створення файлу розмітки HTML. Зазвичай це файл index.html. Нижче приведено структуру файлу даного проекту.

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=utf-8" http-equiv="content-type">

    <title>Quiz</title>

    <link href="css/reset.css" media="screen" rel="stylesheet" type="text/css">
    <link href="css/slickQuiz.css" media="screen" rel="stylesheet" type="text/css">
```

```
    <link href="css/master.css" media="screen" rel="stylesheet" type="text/css">
  </head>

  <body id="slickQuiz">
    <h1 class="quizName"><!-- where the quiz name goes --></h1>

    <div class="quizArea">
      <div class="quizHeader">
        <!-- where the quiz main copy goes -->

        <a class="button startQuiz" href="#">Get Started!</a>
      </div>

      <!-- where the quiz gets built -->
    </div>

    <div class="quizResults">
      <h3 class="quizScore">You Scored: <span><!-- where the quiz score goes --></span></h3>

      <h3 class="quizLevel"><strong>Ranking:</strong> <span><!-- where the quiz ranking level goes --></span></h3>

      <div class="quizResultsCopy">
        <!-- where the quiz result copy goes -->
      </div>
    </div>

    <script src="js/jquery.js"></script>
    <script src="js/slickQuiz-config.js"></script>
```

```
    <script src="js/slickQuiz.js"></script>
    <script src="js/master.js"></script>
  </body>
</html>
```

Далі необхідно розробити модуль авторизації для користувачів (Рис 3.2.). Для цього було використано бібліотеку AJAX, описану вище.

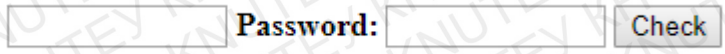

Рис. 3.2. Приклад полей авторизації на сторінці

Приклад опису модуля авторизації:

```
<script language="javascript">
var logins=new Array('1', '2', '3', '4', '5');
var passwords=new Array('5', '4', '3', '2', '1');
function auth()
{
var log=document.form.log.value;
var pass=document.form.pass.value;
 for (i=0; i<logins.length; i++)
 {
    if (logins[i]==log && passwords[i]==pass) {document.location.href =
'http://localhost:63342/Testing/choise.html'; break}
    else if (i==logins.length-1) {alert('Incorrect data'); document.form.log.value='';
document.form.pass.value=''}
 }
}
</script>
</head>
<body>
<form action="" name="form">
  <b> Login:  </b>
```

```
  <input type="text" maxlength="10" size="11" name="log" value="">
  <b> Password: </b>
  <input type="password" maxlength="10" size="11" name="pass" value="">
  <input type="button" value="Check" onClick='auth()'>
</form>
```

Після створення модуля авторизації необхідно розробити модуль тестування. Для цього було використано такі функції:

checkAnswerText(Рядок) За замовчуванням: 'Перевір мою відповідь!'; - текст, який потрібно використати на кнопці відповіді

nextQuestionText(Рядок) За замовчуванням: 'Далі »'; - текст для використання на наступній кнопці питання

completeQuizText(Рядок) За замовчуванням: ''; - текст, який буде використаний для останньої кнопки, яку користувач натисне, перш ніж отримати результати; якщо залишити нулевим / порожнім (за замовчуванням) - nextQuestionTextбуде використано. Приклад: "Отримайте результати!"

backButtonText(Рядок) За замовчуванням: ''; - текст, який потрібно використовувати на кнопці "Назад"; якщо залишити нулевим / порожнім (за замовчуванням) - кнопка повернення назад не відображатиметься

tryAgainText(Рядок) За замовчуванням: ''; - текст, який потрібно використати на кнопці "Повторити спробу"; якщо залишити нульовим / порожнім - кнопка повторної спроби не відображатиметься

preventUnansweredText(Рядок) Дефолт: 'Ви повинні вибрати принаймні одну відповідь.'; - текст, який відображається, якщо користувач подає порожню відповідь під час preventUnansweredувімкнення

questionCountText(Рядок) Дефолт: 'Питання% поточний% загальної кількості'; - якщо displayQuestionCountце ввімкнено, це буде форматувати цей текст, використовуючи наданий рядок. %currenti %totalє заповнювачами, які видадуть відповідні значення. Примітка. displayQuestionCountЗрештою, це може бути застарілим на користь цієї опції

questionTemplateText(Рядок) Дефолт: '% кол. % тексту '; - якщо displayQuestionNumberввімкнено, це буде форматувати номер запитання та запитання, використовуючи наданий рядок. %counti %textє заповнювачами, які видадуть відповідні значення. Примітка. displayQuestionNumberЗрештою, це може бути застарілим на користь цієї опції

scoreTemplateText(Рядок) Дефолт: '% бал /% усього'; - формат тексту підсумкової оцінки. %scorei %totalє заповнювачами, які видадуть відповідні значення

nameTemplateText(Рядок) Defaut: ' Тест: % name'; - формат назви тесту; %name- це заповнювач, який виведе ім'я тесту. Примітка: проміжок "Тест" у значенні за замовчуванням використовується для підвищення доступності, він не відображатиметься на екрані.

Параметри функціональності

skipStartButton(Булева) За замовчуванням: false; - пропустити кнопку "старт" тесту чи ні

numberOfQuestions(Ціле число) За замовчуванням: null; - кількість запитань для завантаження з питання, заданого в об'єкті JSON, за замовчуванням до нуля (усі питання); Примітка. Якщо ви встановите це значення на ціле число, ви, ймовірно, також захочете встановити randomSortQuestionsзначення true, щоб забезпечити отримання змішаного набору питань, що завантажуються на кожну сторінку.

randomSortQuestions(Булева) За замовчуванням: false; - незалежно від того, чи слід випадково сортувати питання

randomSortAnswers(Булева) За замовчуванням: false; - чи слід випадково сортувати відповіді, чи ні

preventUnanswered(Булева) За замовчуванням: false; - заважає надсилати запитання з нульовими відповідями

perQuestionResponseMessaging(Булева) За замовчуванням: вірно; - Відображає правильні / неправильні повідомлення відповіді після подання кожного запитання.

perQuestionResponseAnswers(Булева) За замовчуванням: false; - Зберігає параметри відповіді на дисплеї після надсилання запитання. Примітка: це слід використовувати в тандемі зperQuestionResponseMessaging

completionResponseMessaging(Булева) За замовчуванням: false; - Виводить усі запитання та відповіді з правильними чи неправильними повідомленнями відповідей, коли тест завершено.
displayQuestionCount(Булева) За замовчуванням: вірно; - відображати чи ні кількість запитань і на яке запитання користувач, наприклад, "Питання 3 з 10". Примітка: з часом це може бути застарілим на користьquestionCountText
displayQuestionNumber(Булева) За замовчуванням: вірно; - чи відображати номер питання поряд із самим питанням, наприклад, "1." у "1. Яка перша буква алфавіту?" Примітка: з часом це може бути застарілим на користьquestionTemplateText
disableScore(Булева) За замовчуванням: false; - Видаляє результат з відображення кінцевих результатів. Виключає потребу в елементі з класом quizScorey розмітці.
disableRanking(Булева) За замовчуванням: false; - Видаляє показник рейтингу з відображення кінцевих результатів. Виключає потребу в елементі з класом quizLevely розмітці, а також потреба в значеннях JSON для level1наскрізних level5.
scoreAsPercentage(Булева) За замовчуванням: false; - Повертає бал у відсотках, а не на кількість правильних відповідей. Якщо ввімкнено, ви також хочете налаштувати scoreTemplateTextщось на зразок "% балів"
Параметри питання
Див. "Параметри конфігурації бази" нижче для прикладів
select_any(Булева) Необов’язково - використовувати, якщо є більше однієї правдивої відповіді, і при поданні будь-якої єдиної правдивої відповіді слід вважати правильною. (Оберіть БУДЬ, що застосовується, та оберіть ВСІ, що застосовуються)
force_checkbox(Булева) Необов’язково - встановіть це, trueякщо ви хочете відображати прапорці замість радіостанцій, навіть якщо на питання є лише одна справжня відповідь.
Параметри подій

events.onStartQuiz(функція) За замовчуванням: порожній; - функція, яку слід виконати після запуску тесту.

events.onCompleteQuiz(функція) За замовчуванням: порожній; - функція, яку слід виконати, тест виконано; функція буде передаватися два аргументи в об'єкті: options.questionCount,options.score

Параметри зворотного виклику анімації

animationCallbacks.setupQuiz(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в setupQuizметоді

animationCallbacks.startQuiz(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в startQuizметоді; зауважте, що events.onStartQuiz()виконується до цього методу зворотного виклику через тривалість анімації jQuery

animationCallbacks.resetQuiz(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в resetQuizметоді

animationCallbacks.checkAnswer(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в checkAnswerметоді

animationCallbacks.nextQuestion(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в nextQuestionметоді

animationCallbacks.backToQuestion(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в backToQuestionметоді

animationCallbacks.completeQuiz(функція) За замовчуванням: порожній; - функція, яка виконується після завершення всіх анімацій jQuery в completeQuizметоді; зауважте, що events.onCompleteQuiz()виконується до цього методу зворотного виклику через тривалість анімації jQuery

Застарені параметри

disableNext- Не дозволяє надсилати запитання з нульовими відповідями. Тепер ви повинні використовувати preventUnansweredзамість цього.

disableResponseMessaging- Приховує всі правильні / неправильні повідомлення відповідей. Тепер слід використовувати perQuestionResponseMessagingi completionResponseMessagingзамість цього.

randomSort- Випадково сортуйте всі питання та їх відповіді. Тепер слід використовувати randomSortQuestionsi randomSortAnswerszамість цього.

Розширене використання

Хочете керувати своїми тестами в системі управління вмістом?

Просто переведіть ваші дані тесту CMS у об’єкт JSON, відформатований як quizJSONy js/slickQuiz-config.js. Потім призначте його як quizJSON змінну замість завантаження js/slickQuiz-config.js.

Крім того, ви можете передати JSON прямо у плагін за допомогою jsonпараметра (корисно, якщо ви розміщуєте кілька тестів на сторінці):

```
$(function () {
  $('#slickQuiz').slickQuiz({json: {YOUR_JSON_HERE}});
});
```

Структура бази HTML

Тут важливі ідентифікатор slickQuiz та назви класів:

```
<body id="slickQuiz">
  <h1 class="quizName"></h1>
  <div class="quizArea">
    <div class="quizHeader">
      <a class="startQuiz" href="">Get Started!</a>
    </div>
  </div>
  <div class="quizResults">
    <h3 class="quizScore">You Scored: <span></span></h3>
    <h3 class="quizLevel"><strong>Ranking:</strong> <span></span></h3>
    <div class="quizResultsCopy"></div>
  </div>
</body>
```

Параметри конфігурації бази

js/slickQuiz-config.js

```
var quizJSON = {
  "info": {
    "name":    "The Quiz Header",
    "main":    "The Quiz Description Text",
    "results": "The Quiz Results Copy",
    "level1":  "The highest ranking",
    "level2":  "The almost highest ranking",
    "level3":  "The middle ranking",
    "level4":  "The almost lowest ranking",
    "level5":  "The lowest ranking"
  },
  "questions": [
    {
      "q": "The Question?",
      "a": [
        {"option": "an incorrect answer",      "correct": false},
        {"option": "a correct answer",         "correct": true},
        {"option": "another correct answer",   "correct": true}
      ],
      "correct": "The Correct Response Message",
      "incorrect": "The Incorrect Response Message",
      "select_any": false, // optional, see "Question Options" above
      "force_checkbox": false // optional, see "Question Options" above
    }
  ]
}
```

Додавання HTML до питань та відповідей

Стандартні елементи HTML, такі як зображення, вкладиші до відео, заголовки, абзаци тощо, можуть використовуватися як текстові значення, як q і a параметри.

```
"q": "The Question? <img src='path/to/image.png' />",
"a": [
   {"option": "an <b>incorrect</b> answer", "correct": false},
   {"option": "a <b>correct</b> answer",    "correct": true},
]
```

Приклад тестового питання на створеній сторінці приведено на Рис. 3.3.

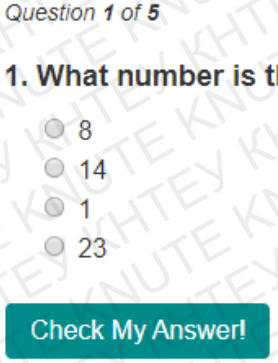

Рис. 3.3. Приклад тестового питання

Також додатково можна реалізувати деякі модулі, які покращать функціонал програмного продукту, як наприклад таймер (Рис. 3.4.).

Рис. 3.4. Таймер для відліку часу

Результат проходження тесту є останнім необхідним етапом створення програмного продукту. Після відповіді на останнє питання додаток покаже користувачеві кількість правильних відповідей (Рис. 3.5.).

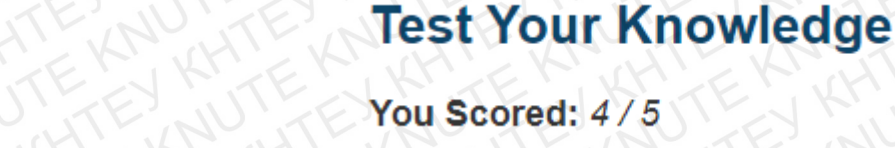

Рис. 3.5. Приклад виведення результату тесту

На цьому етапі основні функціональні потреби програмного продукту є реалізованими, проте можливі модифікації у подальшому користуванні.

3.3. Висновки до розділу 3

У процесі розробки програмного забезпечення тестування знань англійської мови було використано інтегроване середовище розробки веб-сторінок WebStorm. Було досліджено та прийнято до уваги усі переваги використання мови розмітки сторінок HTML, мови веб-програмування JavaScript та її бібліотек JQuery та Ajax, а також мову для оформлення структурованих документів CSS. За допомогою використання даних технологій було створено програмний продукт, що повністю виконує необхідний функціонал поєднаний зі зручним та інтуїтивно зрозумілим користувальницьким інтерфейсом.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У процесі розробки програмного забезпечення тестування знань англійської мови було проаналізовано доцільність знання англійської мови в сучасному світі. Визначено, що знання англійської мови на необхідному рівні дає можливість вести комунікацію зі світом, покращувати свої професійні навички, можливість вирушити у подорож, або навіть просто потренувати свій мозок.

Досліджено процес та форми тестування та визначено найголовніші переваги тестування як форми контролю знань.

Спроектовано модель процесу тестування знань англійської мови, при цьому було визначено необхідні функціональні потреби програмного забезпечення. Завдяки цьому було створено технічне завдання, на основі якого було розроблено програмний продукт. У процесі розробки було розглянуто доцільність використання технологій веб-програмування, таких як мова розмітки сторінок HTML, мова оформлення структурних документів CSS, мова веб-програмування JavaScript та її бібліотеки JQuery та Ajax, та визначено, що дані технології повністю підходять для поєднання інтуїтивно зрозумілого користувальницького інтерфейсу з необхідним функціоналом.

Розроблене програмне забезпечення повністю відповідає заявленому функціоналу, проте є відкритим до будь-яких модифікацій, тому проект є досить гнучким та може бути задіяний для різноманітних цілей. Також за будь-яких умов завдяки гнучкості використаних технологій є можливість налаштувати інтерфейс, так як того потребуватиме користувач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аванесов В. С. Научные основы тестового контроля знаний / В. С. Аванесов. – М. : Исследовательский центр, 1994. – 135 с
2. Гершунский Б.С. Компьютеризация в сфере педагогики. – М.: Педагогика, 1987. – 264 с.
3. Петлюшкин А.В., HTML в Web-дизайне. – СПб.: БВХ-Петербург, 2004. – 400 с.: ил.
4. Габова О.В. Тестування - одна з форм діагностики та перевірки успішності навчання / О.В. Габова, А.А. Русаков. - Київ : САММІТ-КНИГА, 2005. - 340 с.
5. Майоров А.Н. Теорія і практика створення тестів для системи навчання (як вибирать, створювать і використовувати тести для цілей в навчанні): підручник / А.Н. Майоров. - Москва - «Интеллект- центр», 2001 - 296 с.
6. Булах І. Є. Створюємо якісний тест : навч. посіб. / І. Є. Булах, М. Р. Мруга. – К. : Майстер-клас, 2006. – 160 с.
7. Гуревич Р.С., Кадемія М.Ю. Інформаційно-телекомунікаційні технології в навчальному процесі та наукових дослідженнях: навчальний посібник для студентів педагогічних ВНЗ. – Вінниця: ООО “Планер”, 2005. – 366 с.
8. Федорук П.І. Адаптивні тести: статистичні методи обробки результатів тестового контролю знань / // Математичні машини і системи: навч. посібник / П.І. Федорчук; за ред. О.В. Руслановой - Редакційно-видавничий відділ (РВВ) НТУ «ХПІ» , 2007. -138 с.
9. Тестування як засіб перевірки й оцінки знань, умінь і навичок учнів - Режим доступу: http://osvita.ua/school/lessons_summary/proftech/39187/
10. Порядок створення web-сайта - Режим доступу: http://www.usif.org.ua/uk/nashitenderi/652-2017-08-28-14-38-28.html
11. Інформаційно-новинний портал про програмування – Режим доступу: http://www.e-helper.com.ua/ .
12. Веб-документація - Режим доступу: https://developer.mozilla.org/uk/ .

13. The ad-blocking usage report 2019 British Edition. Undertaken by YouGov, commissioned by eyeo. – Режим доступу: https://eyeo.com/2019-uk-ad-blocking-usage-report/.
14. Справочник по HTML - Режим доступу: http://htmlbook.ru/html.
15. DevDocs API Documentation - Режим доступу: https://devdocs.io/.
16. Testing knowledge base - Режим доступу: https://strongqa.com/qa-portal/knowledge-base.
17. 50+ кращих доповнень до Bootstrap – Режим доступу: https://habr.com/ru/company/dataart/blog/258301//.
18. GUI Testing Tutorial: User Interface (UI) TestCases with Examples – Режим доступу: https://www.guru99.com/gui-testing.html.

ДОДАТКИ

Додаток А

Програма та методики тестування

UI тестування – процес перевірки графічного інтерфейсу користувача на предмет його відповідності специфікаціям, загальним принципам і вимогам конкретного проекту. Тестування користувацького інтерфейсу проводилося вручну, оскільки цей вид тестування має такі переваги:

а. UI тестування покриває більшу частину дій користувача і дозволяє зрозуміти якість взаємодії потенційної аудиторії з веб-додатком;

б. Це дає можливість на практиці перевірити взаємодію компонентів і сервісів між собою;

в. Збільшується надійність програми, а знайдені недоліки можна перевірити ще до релізу.

Цей процес передбачає імітацію дій користувача – кліки на кнопки, переходи за посиланнями, та інші дії подібного плану. Таким чином перевіряється коректність роботи, взаємодія компонентів один з одним та зручність інтерфейсу в цілому. До мінусів такого способу тестування можна віднести лише великі затрати часу безпосередньо на тестування. Тестування інтерфейсу програмного забезпечення тестування знань англійської мови проводилося за наступним планом:

1. Відповідність інтерфейсу прототипам. На цьому етапі перевірялась візуальна відповідність положення елементів та відстаней між ними відповідно до макету проекту.

2. Перевірка типографіки. Перевірено текст на читабельність на основному фоні та інших елементах на різних браузерах.

3. Відповідність стилю. Перевірено чи всі елементи відповідають одній кольоровій гаммі, шрифти та інші стандартизовані елементи.

4. Адаптивність. Перевірено як елементи інтерфейсу відображаються на екранах різних розмірів та на різних орієнтаціях.

5. Відповідність стандартам. Перевірено інтерфейс на відповідність вимогам Human Interface Guidelines. Також перевірено чи інтерфейс правильно відображається в середовищах різних операційних систем.

6. Використання функціоналу. Перевірено, що інтерактивні елементи інтерфейсу поводяться належним чином: кнопки реагують на натискання. Також перевірено чи всі елементи достатнього розміру, щоб користувач міг без проблем їх використовувати.

7. Перевірка полів та стандартних елементів. Проведено тестування на те, як поля будуть поводитися при введенні некоретних даних, якщо вводиться довга назва, при виділенні даних. Також перевірено вигляд, положення і реакція чек-боксів, кнопок, полів.

8. Перевірка орфографії. Перевірено текст на орфографічні помилки, оскільки через помилки в тексі чи назвах елементів користувачі можуть неправильно зрозуміти призначення того чи іншого елементу.

9. Елементи інформування. Перевірено вигляд і розміщення всіх повідомлень про помилки, сповіщення тощо. При проектуванні інтерфейсу веб-додатку головними цілями повинні бути користь та зручність для кінцевого користувача. Адже від того, наскільки хорошим він є, залежить чи те, буде людина користуватися додатком далі чи ні.

Додаток Б

Керівництво користувача

Коли користувач потрапляє на головну сторінку, він має пройти авторизацію. Користувачеві необхідно ввести свої дані в поля логіну та паролю та натиснути кнопку «Check» (Рис. Б.1.).

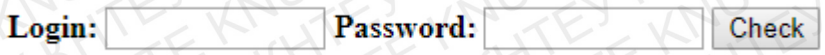

Рис. Б.1. Поля авторизації

У випадку невірного введення логіну або паролю система сповістить про помилку. Після проходження авторизації користувача перенаправить на сторінку вибору тесту (Рис. Б.2.). Також на даній сторінці є кнопка, що дозволяє змінити користувача («Reauthorize»).

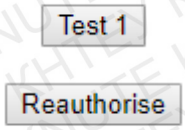

Рис. Б.2. Кнопки вибору тесту та зміни користувача

Після вибору тесту користувач має почати тест натиснувши кнопку «Get Started!» (Рис. Б.3.).

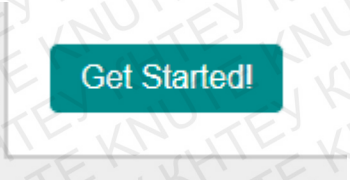

Рис. Б.3. Кнопка початку тестування

Далі користувачеві потрібно відповідати на питання для проходження тесту. Для вибору відповіді потрібно обрати перемикач або необхідну кількість прапорців, та натиснути на кнопку перевірки відповіді (Рис Б.4.). Так повторювати до закінчення тестування.

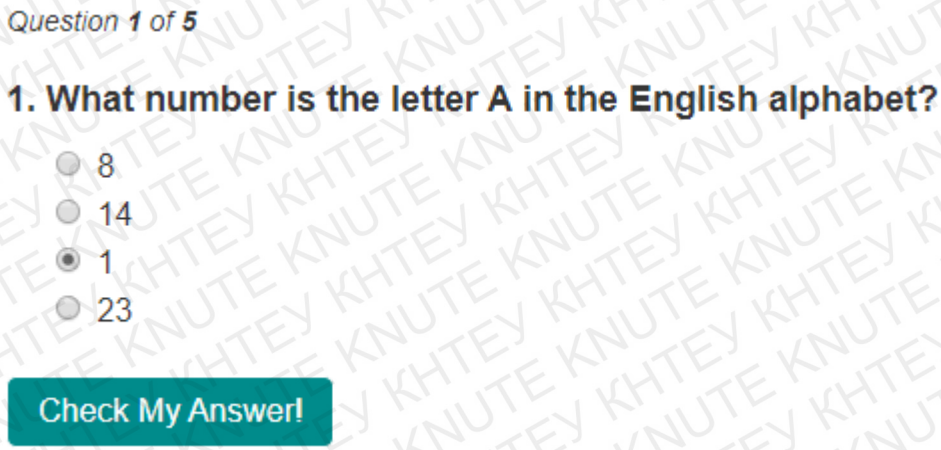

Рис. Б.4. Інтерфейс тестів

Після останнього питання користувача перенаправить на сторінку з результатом його тесту (Рис. Б.5.).

Рис. Б.5. Приклад результату тесту

За бажанням користувач може спробувати знову пройти тестування.

Додаток В

Файл index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=utf-8" http-equiv="content-type">

    <title>Quiz</title>

    <link href="css/reset.css" media="screen" rel="stylesheet" type="text/css">
    <link href="css/slickQuiz.css" media="screen" rel="stylesheet" type="text/css">
    <link href="css/master.css" media="screen" rel="stylesheet" type="text/css">
  </head>

  <body id="slickQuiz">
    <h1 class="quizName"><!-- where the quiz name goes --></h1>

    <div class="quizArea">
      <div class="quizHeader">
        <!-- where the quiz main copy goes -->

        <a class="button startQuiz" href="#">Get Started!</a>
      </div>

      <!-- where the quiz gets built -->
    </div>

    <div class="quizResults">
      <h3 class="quizScore">You Scored: <span><!-- where the quiz score goes --
></span></h3>
```

```
        <h3 class="quizLevel"><strong>Ranking:</strong> <span><!-- where the quiz ranking level goes --></span></h3>

        <div class="quizResultsCopy">
          <!-- where the quiz result copy goes -->
        </div>
      </div>

      <script src="js/jquery.js"></script>
      <script src="js/slickQuiz-config.js"></script>
      <script src="js/slickQuiz.js"></script>
      <script src="js/master.js"></script>
    </body>
</html>
```

Файл slickQuiz.js

```
/*!
 * SlickQuiz jQuery Plugin
 * http://github.com/jewlofthelotus/SlickQuiz
 *
 * @updated October 25, 2014
 * @version 1.5.20
 *
 * @author Julie Cameron - http://www.juliecameron.com
 * @copyright (c) 2013 Quicken Loans - http://www.quickenloans.com
 * @license MIT
 */

(function($){
  $.slickQuiz = function(element, options) {
    var plugin   = this,
      $element = $(element),
      _element = '#' + $element.attr('id'),

      defaults = {
        checkAnswerText:  'Check My Answer!',
        nextQuestionText: 'Next &raquo;',
```

```
            backButtonText: '',
            completeQuizText: '',
            tryAgainText: '',
            questionCountText: 'Question %current of %total',
            preventUnansweredText: 'You must select at least one answer.',
            questionTemplateText:  '%count. %text',
            scoreTemplateText: '%score / %total',
            nameTemplateText:  '<span>Quiz: </span>%name',
            skipStartButton: false,
            numberOfQuestions: null,
            randomSortQuestions: false,
            randomSortAnswers: false,
            preventUnanswered: false,
            disableScore: false,
            disableRanking: false,
            scoreAsPercentage: false,
            perQuestionResponseMessaging: true,
            perQuestionResponseAnswers: false,
            completionResponseMessaging: false,
            displayQuestionCount: true,   // Deprecate?
            displayQuestionNumber: true,  // Deprecate?
            animationCallbacks: { // only for the methods that have jQuery animations offering callback
                setupQuiz: function () {},
                startQuiz: function () {},
                resetQuiz: function () {},
                checkAnswer: function () {},
                nextQuestion: function () {},
                backToQuestion: function () {},
                completeQuiz: function () {}
            },
            events: {
                onStartQuiz: function (options) {},
                onCompleteQuiz: function (options) {}  // reserved: options.questionCount, options.score
            }
        },

        // Class Name Strings (Used for building quiz and for selectors)
        questionCountClass     = 'questionCount',
        questionGroupClass     = 'questions',
        questionClass          = 'question',
        answersClass           = 'answers',
        responsesClass         = 'responses',
        completeClass          = 'complete',
```

```
correctClass          = 'correctResponse',
incorrectClass        = 'incorrectResponse',
correctResponseClass  = 'correct',
incorrectResponseClass = 'incorrect',
checkAnswerClass      = 'checkAnswer',
nextQuestionClass     = 'nextQuestion',
lastQuestionClass     = 'lastQuestion',
backToQuestionClass   = 'backToQuestion',
tryAgainClass         = 'tryAgain',

// Sub-Quiz / Sub-Question Class Selectors
_questionCount        = '.' + questionCountClass,
_questions            = '.' + questionGroupClass,
_question             = '.' + questionClass,
_answers              = '.' + answersClass,
_answer               = '.' + answersClass + ' li',
_responses            = '.' + responsesClass,
_response             = '.' + responsesClass + ' li',
_correct              = '.' + correctClass,
_correctResponse      = '.' + correctResponseClass,
_incorrectResponse    = '.' + incorrectResponseClass,
_checkAnswerBtn       = '.' + checkAnswerClass,
_nextQuestionBtn      = '.' + nextQuestionClass,
_prevQuestionBtn      = '.' + backToQuestionClass,
_tryAgainBtn          = '.' + tryAgainClass,

// Top Level Quiz Element Class Selectors
_quizStarter          = _element + ' .startQuiz',
_quizName             = _element + ' .quizName',
_quizArea             = _element + ' .quizArea',
_quizResults          = _element + ' .quizResults',
_quizResultsCopy      = _element + ' .quizResultsCopy',
_quizHeader           = _element + ' .quizHeader',
_quizScore            = _element + ' .quizScore',
_quizLevel            = _element + ' .quizLevel',

// Top Level Quiz Element Objects
$quizStarter          = $(_quizStarter),
$quizName             = $(_quizName),
$quizArea             = $(_quizArea),
$quizResults          = $(_quizResults),
$quizResultsCopy      = $(_quizResultsCopy),
$quizHeader           = $(_quizHeader),
$quizScore            = $(_quizScore),
$quizLevel            = $(_quizLevel)
```

```
    ;


    // Reassign user-submitted deprecated options
    var depMsg = '';

    if (options && typeof options.disableNext != 'undefined') {
      if (typeof options.preventUnanswered == 'undefined') {
        options.preventUnanswered = options.disableNext;
      }
      depMsg += 'The \'disableNext\' option has been deprecated, please use
\'preventUnanswered\' in it\'s place.\n\n';
    }

    if (options && typeof options.disableResponseMessaging != 'undefined') {
      if (typeof options.preventUnanswered == 'undefined') {
        options.perQuestionResponseMessaging =
options.disableResponseMessaging;
      }
      depMsg += 'The \'disableResponseMessaging\' option has been deprecated,
please use' +
          ' \'perQuestionResponseMessaging\' and \'completionResponseMessaging\'
in it\'s place.\n\n';
    }

    if (options && typeof options.randomSort != 'undefined') {
      if (typeof options.randomSortQuestions == 'undefined') {
        options.randomSortQuestions = options.randomSort;
      }
      if (typeof options.randomSortAnswers == 'undefined') {
        options.randomSortAnswers = options.randomSort;
      }
      depMsg += 'The \'randomSort\' option has been deprecated, please use' +
          ' \'randomSortQuestions\' and \'randomSortAnswers\' in it\'s place.\n\n';
    }

    if (depMsg !== '') {
      if (typeof console != 'undefined') {
        console.warn(depMsg);
      } else {
        alert(depMsg);
      }
    }
    // End of deprecation reassignment
```

```
        plugin.config = $.extend(defaults, options);

        // Set via json option or quizJSON variable (see slickQuiz-config.js)
        var quizValues = (plugin.config.json ? plugin.config.json : typeof quizJSON !=
'undefined' ? quizJSON : null);

        // Get questions, possibly sorted randomly
        var questions = plugin.config.randomSortQuestions ?
                    quizValues.questions.sort(function() { return
(Math.round(Math.random())-0.5); }) :
                    quizValues.questions;

        // Count the number of questions
        var questionCount = questions.length;

        // Select X number of questions to load if options is set
        if (plugin.config.numberOfQuestions && questionCount >=
plugin.config.numberOfQuestions) {
          questions = questions.slice(0, plugin.config.numberOfQuestions);
          questionCount = questions.length;
        }

        // some special private/internal methods
        var internal = {method: {
          // get a key whose notches are "resolved jQ deferred" objects; one per notch on
the key
          // think of the key as a house key with notches on it
          getKey: function (notches) { // returns [], notches >= 1
            var key = [];
            for (i=0; i<notches; i++) key[i] = $.Deferred ();
            return key;
          },

          // put the key in the door, if all the notches pass then you can turn the key and
"go"
          turnKeyAndGo: function (key, go) { // key = [], go = function ()
            // when all the notches of the key are accepted (resolved) then the key turns
and the engine (callback/go) starts
            $.when.apply (null, key). then (function () {
              go ();
            });
          },

          // get one jQ
```

```
        getKeyNotch: function (key, notch) { // notch >= 1, key = []
            // key has several notches, numbered as 1, 2, 3, ... (no zero notch)
            // we resolve and return the "jQ deferred" object at specified notch
            return function () {
                key[notch-1].resolve (); // it is ASSUMED that you initiated the key with
enough notches
            };
        }
    }};

    plugin.method = {
        // Sets up the questions and answers based on above array
        setupQuiz: function(options) { // use 'options' object to pass args
            var key, keyNotch, kN;
            key = internal.method.getKey (3); // how many notches == how many jQ
animations you will run
            keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
            kN = keyNotch; // you specify the notch, you get a callback function for your
animation

            $quizName.hide().html(plugin.config.nameTemplateText
                .replace('%name', quizValues.info.name) ).fadeIn(1000, kN(key,1));
            $quizHeader.hide().prepend($('<div class="quizDescription">' +
quizValues.info.main + '</div>')).fadeIn(1000, kN(key,2));
            $quizResultsCopy.append(quizValues.info.results);

            // add retry button to results view, if enabled
            if (plugin.config.tryAgainText && plugin.config.tryAgainText !== '') {
                $quizResultsCopy.append('<p><a class="button ' + tryAgainClass + '"
href="#">' + plugin.config.tryAgainText + '</a></p>');
            }

            // Setup questions
            var quiz  = $('<ol class="' + questionGroupClass + '"></ol>'),
                count = 1;

            // Loop through questions object
            for (i in questions) {
                if (questions.hasOwnProperty(i)) {
                    var question = questions[i];

                    var questionHTML = $('<li class="' + questionClass +'" id="question' +
(count - 1) + '"></li>');
```

```
        if (plugin.config.displayQuestionCount) {
            questionHTML.append('<div class="' + questionCountClass + '">' +
                plugin.config.questionCountText
                    .replace('%current', '<span class="current">' + count + '</span>')
                    .replace('%total', '<span class="total">' +
                        questionCount + '</span>') + '</div>');
        }

        var formatQuestion = '';
        if (plugin.config.displayQuestionNumber) {
            formatQuestion = plugin.config.questionTemplateText
                .replace('%count', count).replace('%text', question.q);
        } else {
            formatQuestion = question.q;
        }
        questionHTML.append('<h3>' + formatQuestion + '</h3>');

        // Count the number of true values
        var truths = 0;
        for (i in question.a) {
            if (question.a.hasOwnProperty(i)) {
                answer = question.a[i];
                if (answer.correct) {
                    truths++;
                }
            }
        }

        // Now let's append the answers with checkboxes or radios depending on
truth count
        var answerHTML = $('<ul class="' + answersClass + '"></ul>');

        // Get the answers
        var answers = plugin.config.randomSortAnswers ?
            question.a.sort(function() { return (Math.round(Math.random())-0.5);
}) :
            question.a;

        // prepare a name for the answer inputs based on the question
        var selectAny     = question.select_any ? question.select_any : false,
            forceCheckbox = question.force_checkbox ? question.force_checkbox
: false,
            checkbox      = (truths > 1 && !selectAny) || forceCheckbox,
            inputName     = $element.attr('id') + '_question' + (count - 1),
            inputType     = checkbox ? 'checkbox' : 'radio';
```

```
                if( count == quizValues.questions.length ) {
                    nextQuestionClass = nextQuestionClass + ' ' + lastQuestionClass;
                }

                for (i in answers) {
                    if (answers.hasOwnProperty(i)) {
                        answer   = answers[i],
                        optionId = inputName + '_' + i.toString();

                        // If question has >1 true answers and is not a select any, use
checkboxes; otherwise, radios
                        var input = '<input id="' + optionId + '" name="' + inputName +
                                '" type="' + inputType + '" /> ';

                        var optionLabel = '<label for="' + optionId + '">' + answer.option +
'</label>';

                        var answerContent = $('<li></li>')
                            .append(input)
                            .append(optionLabel);
                        answerHTML.append(answerContent);
                    }
                }

                // Append answers to question
                questionHTML.append(answerHTML);

                // If response messaging is NOT disabled, add it
                if (plugin.config.perQuestionResponseMessaging ||
plugin.config.completionResponseMessaging) {
                    // Now let's append the correct / incorrect response messages
                    var responseHTML = $('<ul class="' + responsesClass + '"></ul>');
                    responseHTML.append('<li class="' + correctResponseClass + '">' +
question.correct + '</li>');
                    responseHTML.append('<li class="' + incorrectResponseClass + '">' +
question.incorrect + '</li>');

                    // Append responses to question
                    questionHTML.append(responseHTML);
                }

                // Appends check answer / back / next question buttons
                if (plugin.config.backButtonText && plugin.config.backButtonText !==
'') {
```

```
                questionHTML.append('<a href="#" class="button ' +
backToQuestionClass + '">' + plugin.config.backButtonText + '</a>');
            }

            var nextText = plugin.config.nextQuestionText;
            if (plugin.config.completeQuizText && count == questionCount) {
              nextText = plugin.config.completeQuizText;
            }

            // If we're not showing responses per question, show next question button
and make it check the answer too
            if (!plugin.config.perQuestionResponseMessaging) {
              questionHTML.append('<a href="#" class="button ' +
nextQuestionClass + ' ' + checkAnswerClass + '">' + nextText + '</a>');
            } else {
              questionHTML.append('<a href="#" class="button ' +
nextQuestionClass + '">' + nextText + '</a>');
              questionHTML.append('<a href="#" class="button ' +
checkAnswerClass + '">' + plugin.config.checkAnswerText + '</a>');
            }

            // Append question & answers to quiz
            quiz.append(questionHTML);

            count++;
          }
        }

        // Add the quiz content to the page
        $quizArea.append(quiz);

        // Toggle the start button OR start the quiz if start button is disabled
        if (plugin.config.skipStartButton || $quizStarter.length == 0) {
          $quizStarter.hide();
          plugin.method.startQuiz.apply (this, [{callback:
plugin.config.animationCallbacks.startQuiz}]); // TODO: determine why 'this' is being
passed as arg to startQuiz method
          kN(key,3).apply (null, []);
        } else {
          $quizStarter.fadeIn(500, kN(key,3)); // 3d notch on key must be on both
sides of if/else, otherwise key won't turn
        }

        internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});
```

```
    },

    // Starts the quiz (hides start button and displays first question)
    startQuiz: function(options) {
      var key, keyNotch, kN;
      key = internal.method.getKey (1); // how many notches == how many jQ
animations you will run
      keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
      kN = keyNotch; // you specify the notch, you get a callback function for your
animation

      function start(options) {
        var firstQuestion = $(_element + ' ' + _questions + ' li').first();
        if (firstQuestion.length) {
          firstQuestion.fadeIn(500, function () {
            if (options && options.callback) options.callback ();
          });
        }
      }

      if (plugin.config.skipStartButton || $quizStarter.length == 0) {
        start({callback: kN(key,1)});
      } else {
        $quizStarter.fadeOut(300, function(){
          start({callback: kN(key,1)}); // 1st notch on key must be on both sides of
if/else, otherwise key won't turn
        });
      }

      internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});

      if (plugin.config.events &&
          plugin.config.events.onStartQuiz) {
        plugin.config.events.onStartQuiz.apply (null, []);
      }
    },

    // Resets (restarts) the quiz (hides results, resets inputs, and displays first
question)
    resetQuiz: function(startButton, options) {
      var key, keyNotch, kN;
      key = internal.method.getKey (1); // how many notches == how many jQ
animations you will run
```

```
            keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
            kN = keyNotch; // you specify the notch, you get a callback function for your
animation

            $quizResults.fadeOut(300, function() {
              $(_element + ' input').prop('checked', false).prop('disabled', false);

              $quizLevel.attr('class', 'quizLevel');
              $(_element + ' ' +
_question).removeClass(correctClass).removeClass(incorrectClass).remove(completeCl
ass);
              $(_element + ' ' +
_answer).removeClass(correctResponseClass).removeClass(incorrectResponseClass);

              $(_element + ' ' + _question         + ',' +
                _element + ' ' + _responses        + ',' +
                _element + ' ' + _response         + ',' +
                _element + ' ' + _nextQuestionBtn   + ',' +
                _element + ' ' + _prevQuestionBtn
              ).hide();

              $(_element + ' ' + _questionCount + ',' +
                _element + ' ' + _answers + ',' +
                _element + ' ' + _checkAnswerBtn
              ).show();

              $quizArea.append($(_element + ' ' + _questions)).show();

              kN(key,1).apply (null, []);

              plugin.method.startQuiz({callback:
plugin.config.animationCallbacks.startQuiz},$quizResults); // TODO: determine why
$quizResults is being passed
            });

            internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});
          },

          // Validates the response selection(s), displays explanations & next question
button
          checkAnswer: function(checkButton, options) {
            var key, keyNotch, kN;
```

```
        key = internal.method.getKey (2); // how many notches == how many jQ
animations you will run
        keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
        kN = keyNotch; // you specify the notch, you get a callback function for your
animation

        var questionLI    = $($(checkButton).parents(_question)[0]),
           answerLIs     = questionLI.find(_answers + ' li'),
           answerSelects = answerLIs.find('input:checked'),
           questionIndex = parseInt(questionLI.attr('id').replace(/(question)/, ''), 10),
           answers       = questions[questionIndex].a,
           selectAny     = questions[questionIndex].select_any ?
questions[questionIndex].select_any : false;

        answerLIs.addClass(incorrectResponseClass);

        // Collect the true answers needed for a correct response
        var trueAnswers = [];
        for (i in answers) {
           if (answers.hasOwnProperty(i)) {
              var answer = answers[i],
                 index  = parseInt(i, 10);

              if (answer.correct) {
                 trueAnswers.push(index);

answerLIs.eq(index).removeClass(incorrectResponseClass).addClass(correctResponseC
lass);
              }
           }
        }

        // TODO: Now that we're marking answer LIs as correct / incorrect, we might
be able
        // to do all our answer checking at the same time

        // NOTE: Collecting answer index for comparison aims to ensure that HTML
entities
        // and HTML elements that may be modified by the browser / other scrips
match up

        // Collect the answers submitted
        var selectedAnswers = [];
        answerSelects.each( function() {
```

```
            var id = $(this).attr('id');
            selectedAnswers.push(parseInt(id.replace(/(.*\_question\d{1,}_)/, ''), 10));
          });

          if (plugin.config.preventUnanswered && selectedAnswers.length === 0) {
            alert(plugin.config.preventUnansweredText);
            return false;
          }

          // Verify all/any true answers (and no false ones) were submitted
          var correctResponse = plugin.method.compareAnswers(trueAnswers,
selectedAnswers, selectAny);

          if (correctResponse) {
            questionLI.addClass(correctClass);
          } else {
            questionLI.addClass(incorrectClass);
          }

          // Toggle appropriate response (either for display now and / or on completion)
          questionLI.find(correctResponse ? _correctResponse :
_incorrectResponse).show();

          // If perQuestionResponseMessaging is enabled, toggle response and
navigation now
          if (plugin.config.perQuestionResponseMessaging) {
            $(checkButton).hide();
            if (!plugin.config.perQuestionResponseAnswers) {
              // Make sure answers don't highlight for a split second before they hide
              questionLI.find(_answers).hide({
                duration: 0,
                complete: function() {
                  questionLI.addClass(completeClass);
                }
              });
            } else {
              questionLI.addClass(completeClass);
            }
            questionLI.find('input').prop('disabled', true);
            questionLI.find(_responses).show();
            questionLI.find(_nextQuestionBtn).fadeIn(300, kN(key,1));
            questionLI.find(_prevQuestionBtn).fadeIn(300, kN(key,2));
            if (!questionLI.find(_prevQuestionBtn).length) kN(key,2).apply (null, []); //
2nd notch on key must be passed even if there's no "back" button
          } else {
```

```
            kN(key,1).apply (null, []); // 1st notch on key must be on both sides of
if/else, otherwise key won't turn
            kN(key,2).apply (null, []); // 2nd notch on key must be on both sides of
if/else, otherwise key won't turn
          }

          internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});
        },

        // Moves to the next question OR completes the quiz if on last question
        nextQuestion: function(nextButton, options) {
          var key, keyNotch, kN;
          key = internal.method.getKey (1); // how many notches == how many jQ
animations you will run
          keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
          kN = keyNotch; // you specify the notch, you get a callback function for your
animation

          var currentQuestion = $($(nextButton).parents(_question)[0]),
            nextQuestion    = currentQuestion.next(_question),
            answerInputs    = currentQuestion.find('input:checked');

          // If response messaging has been disabled or moved to completion,
          // make sure we have an answer if we require it, let checkAnswer handle the
alert messaging
          if (plugin.config.preventUnanswered && answerInputs.length === 0) {
            return false;
          }

          if (nextQuestion.length) {
            currentQuestion.fadeOut(300, function(){
              nextQuestion.find(_prevQuestionBtn).show().end().fadeIn(500,
kN(key,1));
              if (!nextQuestion.find(_prevQuestionBtn).show().end().length)
kN(key,1).apply (null, []); // 1st notch on key must be passed even if there's no "back"
button
            });
          } else {
            kN(key,1).apply (null, []); // 1st notch on key must be on both sides of
if/else, otherwise key won't turn
            plugin.method.completeQuiz({callback:
plugin.config.animationCallbacks.completeQuiz});
          }
```

```
            internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});
        },

        // Go back to the last question
        backToQuestion: function(backButton, options) {
            var key, keyNotch, kN;
            key = internal.method.getKey (2); // how many notches == how many jQ
animations you will run
            keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
            kN = keyNotch; // you specify the notch, you get a callback function for your
animation

            var questionLI = $($(backButton).parents(_question)[0]),
                responses  = questionLI.find(_responses);

            // Back to question from responses
            if (responses.css('display') === 'block' ) {
                questionLI.find(_responses).fadeOut(300, function(){

questionLI.removeClass(correctClass).removeClass(incorrectClass).removeClass(compl
eteClass);
                    questionLI.find(_responses + ', ' + _response).hide();
                    questionLI.find(_answers).show();

questionLI.find(_answer).removeClass(correctResponseClass).removeClass(incorrectR
esponseClass);
                    questionLI.find('input').prop('disabled', false);
                    questionLI.find(_answers).fadeIn(500, kN(key,1)); // 1st notch on key
must be on both sides of if/else, otherwise key won't turn
                    questionLI.find(_checkAnswerBtn).fadeIn(500, kN(key,2));
                    questionLI.find(_nextQuestionBtn).hide();

                    // if question is first, don't show back button on question
                    if (questionLI.attr('id') != 'question0') {
                        questionLI.find(_prevQuestionBtn).show();
                    } else {
                        questionLI.find(_prevQuestionBtn).hide();
                    }
                });

            // Back to previous question
            } else {
```

```
            var prevQuestion = questionLI.prev(_question);

            questionLI.fadeOut(300, function() {

prevQuestion.removeClass(correctClass).removeClass(incorrectClass).removeClass(co
mpleteClass);
                prevQuestion.find(_responses + ', ' + _response).hide();
                prevQuestion.find(_answers).show();

prevQuestion.find(_answer).removeClass(correctResponseClass).removeClass(incorrect
ResponseClass);
                prevQuestion.find('input').prop('disabled', false);
                prevQuestion.find(_nextQuestionBtn).hide();
                prevQuestion.find(_checkAnswerBtn).show();

                if (prevQuestion.attr('id') != 'question0') {
                  prevQuestion.find(_prevQuestionBtn).show();
                } else {
                  prevQuestion.find(_prevQuestionBtn).hide();
                }

                prevQuestion.fadeIn(500, kN(key,1));
                kN(key,2).apply (null, []); // 2nd notch on key must be on both sides of
if/else, otherwise key won't turn
            });
          }

          internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});
        },

        // Hides all questions, displays the final score and some conclusive information
        completeQuiz: function(options) {
          var key, keyNotch, kN;
          key = internal.method.getKey (1); // how many notches == how many jQ
animations you will run
          keyNotch = internal.method.getKeyNotch; // a function that returns a jQ
animation callback function
          kN = keyNotch; // you specify the notch, you get a callback function for your
animation

          var score       = $(_element + ' ' + _correct).length,
            displayScore = score;
          if (plugin.config.scoreAsPercentage) {
            displayScore = (score / questionCount).toFixed(2)*100 + "%";
```

```
        }

        if (plugin.config.disableScore) {
            $(_quizScore).remove()
        } else {
            $(_quizScore + ' span').html(plugin.config.scoreTemplateText
                .replace('%score', displayScore).replace('%total', questionCount));
        }

        if (plugin.config.disableRanking) {
            $(_quizLevel).remove()
        } else {
            var levels    = [
                            quizValues.info.level1, // 80-100%
                            quizValues.info.level2, // 60-79%
                            quizValues.info.level3, // 40-59%
                            quizValues.info.level4, // 20-39%
                            quizValues.info.level5  // 0-19%
                        ],
                levelRank = plugin.method.calculateLevel(score),
                levelText = $.isNumeric(levelRank) ? levels[levelRank] : '';

            $(_quizLevel + ' span').html(levelText);
            $(_quizLevel).addClass('level' + levelRank);
        }

        $quizArea.fadeOut(300, function() {
            // If response messaging is set to show upon quiz completion, show it now
            if (plugin.config.completionResponseMessaging) {
                $(_element + ' .button:not(' + _tryAgainBtn + '), ' + _element + ' ' +
_questionCount).hide();
                $(_element + ' ' + _question + ', ' + _element + ' ' + _answers + ', ' +
_element + ' ' + _responses).show();
                $quizResults.append($(_element + ' ' + _questions)).fadeIn(500,
kN(key,1));
            } else {
                $quizResults.fadeIn(500, kN(key,1)); // 1st notch on key must be on both
sides of if/else, otherwise key won't turn
            }
        });

        internal.method.turnKeyAndGo (key, options && options.callback ?
options.callback : function () {});

        if (plugin.config.events &&
```

```
            plugin.config.events.onCompleteQuiz) {
          plugin.config.events.onCompleteQuiz.apply (null, [{
            questionCount: questionCount,
            score: score
          }]);
        }
      },

      // Compares selected responses with true answers, returns true if they match
exactly
      compareAnswers: function(trueAnswers, selectedAnswers, selectAny) {
        if ( selectAny ) {
          return $.inArray(selectedAnswers[0], trueAnswers) > -1;
        } else {
          // crafty array comparison (http://stackoverflow.com/a/7726509)
          return ($(trueAnswers).not(selectedAnswers).length === 0 &&
$(selectedAnswers).not(trueAnswers).length === 0);
        }
      },

      // Calculates knowledge level based on number of correct answers
      calculateLevel: function(correctAnswers) {
        var percent = (correctAnswers / questionCount).toFixed(2),
          level   = null;

        if (plugin.method.inRange(0, 0.20, percent)) {
          level = 4;
        } else if (plugin.method.inRange(0.21, 0.40, percent)) {
          level = 3;
        } else if (plugin.method.inRange(0.41, 0.60, percent)) {
          level = 2;
        } else if (plugin.method.inRange(0.61, 0.80, percent)) {
          level = 1;
        } else if (plugin.method.inRange(0.81, 1.00, percent)) {
          level = 0;
        }

        return level;
      },

      // Determines if percentage of correct values is within a level range
      inRange: function(start, end, value) {
        return (value >= start && value <= end);
      }
    };
```

```
        plugin.init = function() {
            // Setup quiz
            plugin.method.setupQuiz.apply (null, [{callback:
plugin.config.animationCallbacks.setupQuiz}]);

            // Bind "start" button
            $quizStarter.on('click', function(e) {
                e.preventDefault();

                if (!this.disabled && !$(this).hasClass('disabled')) {
                    plugin.method.startQuiz.apply (null, [{callback:
plugin.config.animationCallbacks.startQuiz}]);
                }
            });

            // Bind "try again" button
            $(_element + ' ' + _tryAgainBtn).on('click', function(e) {
                e.preventDefault();
                plugin.method.resetQuiz(this, {callback:
plugin.config.animationCallbacks.resetQuiz});
            });

            // Bind "check answer" buttons
            $(_element + ' ' + _checkAnswerBtn).on('click', function(e) {
                e.preventDefault();
                plugin.method.checkAnswer(this, {callback:
plugin.config.animationCallbacks.checkAnswer});
            });

            // Bind "back" buttons
            $(_element + ' ' + _prevQuestionBtn).on('click', function(e) {
                e.preventDefault();
                plugin.method.backToQuestion(this, {callback:
plugin.config.animationCallbacks.backToQuestion});
            });

            // Bind "next" buttons
            $(_element + ' ' + _nextQuestionBtn).on('click', function(e) {
                e.preventDefault();
                plugin.method.nextQuestion(this, {callback:
plugin.config.animationCallbacks.nextQuestion});
            });

            // Accessibility (WAI-ARIA).
```

```
        var _qnid = $element.attr('id') + '-name';
        $quizName.attr('id', _qnid);
        $element.attr({
          'aria-labelledby': _qnid,
          'aria-live': 'polite',
          'aria-relevant': 'additions',
          'role': 'form'
        });
        $(_quizStarter + ', [href = "#"]').attr('role', 'button');
      };

      plugin.init();
    };

    $.fn.slickQuiz = function(options) {
      return this.each(function() {
        if (undefined === $(this).data('slickQuiz')) {
          var plugin = new $.slickQuiz(this, options);
          $(this).data('slickQuiz', plugin);
        }
      });
    };
})(jQuery);
```

Файл slickQuiz-config.js

```
var quizJSON = {
  "info": {
    "name":    "Test Your Knowledge!!",
    "main":    "<p>Think you're smart enough to be on Jeopardy? Find out with this super crazy knowledge quiz!</p>",
    "results": "<h5>Learn More</h5><p>Etiam scelerisque, nunc ac egestas consequat, odio nibh euismod nulla, eget auctor orci nibh vel nisi. Aliquam erat volutpat. Mauris vel neque sit amet nunc gravida congue sed sit amet purus.</p>",
    "level1":  "Jeopardy Ready",
    "level2":  "Jeopardy Contender",
    "level3":  "Jeopardy Amateur",
    "level4":  "Jeopardy Newb",
    "level5":  "Stay in school, kid..." // no comma here
  },
  "questions": [
    { // Question 1 - Multiple Choice, Single True Answer
      "q": "What number is the letter A in the English alphabet?",
      "a": [
        {"option": "8",     "correct": false},
        {"option": "14",    "correct": false},
```

```
            {"option": "1",      "correct": true},
            {"option": "23",     "correct": false} // no comma here
        ],
        "correct": "<p><span>That's right!</span> The letter A is the first letter in the alphabet!</p>",
        "incorrect": "<p><span>Uhh no.</span> It's the first letter of the alphabet. Did you actually <em>go</em> to kindergarden?</p>" // no comma here
    },
    { // Question 2 - Multiple Choice, Multiple True Answers, Select Any
        "q": "Which of the following best represents your preferred breakfast?",
        "a": [
            {"option": "Bacon and eggs",              "correct": false},
            {"option": "Fruit, oatmeal, and yogurt",  "correct": true},
            {"option": "Leftover pizza",              "correct": false},
            {"option": "Eggs, fruit, toast, and milk", "correct": true} // no comma here
        ],
        "select_any": true,
        "correct": "<p><span>Nice!</span> Your cholestoral level is probably doing alright.</p>",
        "incorrect": "<p><span>Hmmm.</span> You might want to reconsider your options.</p>" // no comma here
    },
    { // Question 3 - Multiple Choice, Multiple True Answers, Select All
        "q": "Where are you right now? Select ALL that apply.",
        "a": [
            {"option": "Planet Earth",          "correct": true},
            {"option": "Pluto",                "correct": false},
            {"option": "At a computing device", "correct": true},
            {"option": "The Milky Way",        "correct": true} // no comma here
        ],
        "correct": "<p><span>Brilliant!</span> You're seriously a genius, (wo)man.</p>",
        "incorrect": "<p><span>Not Quite.</span> You're actually on Planet Earth, in The Milky Way, At a computer. But nice try.</p>" // no comma here
    },
    { // Question 4
        "q": "How many inches of rain does Michigan get on average per year?",
        "a": [
            {"option": "149",    "correct": false},
            {"option": "32",     "correct": true},
            {"option": "3",      "correct": false},
            {"option": "1291",   "correct": false} // no comma here
        ],
        "correct": "<p><span>Holy bananas!</span> I didn't actually expect you to know that! Correct!</p>",
```

```
        "incorrect": "<p><span>Fail.</span> Sorry. You lose. It actually rains approximately 32 inches a year in Michigan.</p>" // no comma here
      },
      { // Question 5
        "q": "Is Earth bigger than a basketball?",
        "a": [
          {"option": "Yes",    "correct": true},
          {"option": "No",     "correct": false} // no comma here
        ],
        "correct": "<p><span>Good Job!</span> You must be very observant!</p>",
        "incorrect": "<p><span>ERRRR!</span> What planet Earth are <em>you</em> living on?!?</p>" // no comma here
      } // no comma here
    ]
};
```

Київський національний торговельно-економічний університет
Кафедра програмної інженерії та кібербезпеки

РЕФЕРАТ

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення тестування знань англійської мови»

Студента 2м курсу, 5 групи, спеціальності 121«Інженерія програмного забезпечення», спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Деремешка Максима Михайловича

Науковий керівник кандидат технічних наук, професор _____ підпис керівника — Пашорін Валерій Іванович

Гарант освітньої програми доктор технічних наук, професор _____ підпис гаранта — Криворучко Олена Володимирівна

КИЇВ – 2019

Аннотація

Актуалність даної роботи полягає у тому, то англійська - це одна з найбільш використовуваних мов світу. Якщо враховувати усіх, для кого англійська стала другою мовою, виходить близько 1 мільярда англомовних жителів світу. Крім того, англійська є офіційною мовою в 67 країнах світу, і ще є 27 країн, де англійська є другою офіційною мовою. Щоб сам процес тестування був зручним для користувачів та максимально об’єктивним, доцільно використовувати саме методи та засоби web-розробки, спираючись на рекомендації правильного формату тестування. Сучасне суспільство має доступ до мережі Інтернет майже завжди та майже в буд-якій точці світу, тому мережева сторінка не матиме проблем з доступом.

Метою роботи є створення інтерфейсу користувача та модуля тестування, які дадуть змогу користувачеві, використовуючи функціонал отримувати можливість тестувати свої знання англійської мови.

Annotation

The relevance of this work is that English is one of the most used languages in the world. If you consider everyone for whom English has become a second language, there are about 1 billion English-speaking inhabitants of the world. In addition, English is the official language in 67 countries, and there are still 27 countries where English is the second official language. In order for the testing process to be user-friendly and as objective as possible, it is advisable to use web development methods and tools based on the recommendations of the correct testing format. Modern society has access to the Internet almost always and almost anywhere in the world, so the web page will not have access problems.

The purpose of the work is to create a user interface and a testing module that will allow the user to use the functionality to test their English skills.

Загальна характеристика роботи

Актуальність: англійська - це одна з найбільш використовуваних мов світу. Якщо враховувати усіх, для кого англійська стала другою мовою, виходить близько 1 мільярда англомовних жителів світу. Крім того, англійська є офіційною мовою в 67 країнах світу, і ще є 27 країн, де англійська є другою офіційною мовою.

Щоб сам процес тестування був зручним для користувачів та максимально об’єктивним, доцільно використовувати саме методи та засоби web-розробки, спираючись на рекомендації правильного формату тестування. Сучасне суспільство має доступ до мережі Інтернет майже завжди та майже в буд-якій точці світу, тому мережева сторінка не матиме проблем з доступом. Крім того, далеко не кожен користувач має бажання завантажувати та встановлювати на власному комп’ютері невідоме програмне забезпечення.

Об'єкт дослідження: тестування знань англійської мови.

Предмет дослідження: розробка можливої моделі програмного забезпечення тестування.

Мета роботи: розробити модуль тестування знань англійської мови та зручний інтерфейс користувача.

Задачі дослідження: розробити інтерфейс користувача та модуль тестування, які дають змогу користувачеві використовувати функціонал програмного забезпечення та отримувати результати тестування в зручному вигляді.

Методи і технології розробки: аналіз алгоритмів пошуку, веб-програмування, техногологія front-end розробки, візуальне і об'єктно-орієнтоване програмування, порівняння, спостереження, побудова програмного продукту на основі проаналізованих данних.

Результат роботи: Створення програмного забезпечення тестування знань англійської мови.

ОСНОВНИЙ ЗМІСТ ДИПЛОМНОЇ РОБОТИ

У **вступі** випускної кваліфікаційної роботи обґрунтовано актуальність теми, визначено об’єкт, предмет, мету і завдання роботи.

Перший розділ «Аналіз процесу тестування знань англійської мови». У даному розділі розглянуто доцільність знання англійської мови в сучасному світі, в тому числі для амбіційної та енергічної частки населення, а саме молоді. Визначено, що знання англійської мови на необхідному рівні дає можливість вести комунікацію зі світом, покращувати свої професійні навички, можливість вирушити у подорож, або навіть просто потренувати свій мозок.

Досліджено процес та форми тестування. Завдяки можливості використання процесу тестування як апарату перевірки знань можна отримати такі переваги:

− можливість застосування як засобу усіх видів контролю, а саме базового та початкового, поточного та тематичного, рубіжного та залікового, підсумкового та екзаменаційного, а також самоконтролю;

− можливість детальної перевірки рівня засвоєння кожного змістового модуля;

− наявність чіткої однозначної відповіді;

− економія навчального часу при здійсненні поточного контролю знань та об’єктивність оцінювання результатів навчання;

− мінімізація емоційного впливу;

− індивідуальний характер контролю, можливість здійснення контролю за роботою кожного, його особистою навчальною діяльністю;

− можливість регулярного систематичного проведення тестового контролю на всіх етапах процесу навчання;

− можливість поєднання його з іншими традиційними формами педагогічного контролю;

− можливість проведення комп’ютеризованого у локальній мережі та паперового варіанта тестування;

− урахування індивідуальних особливостей, що потребує застосування відповідно до цих особливостей різної методики розробки тесту й тестових завдань.

Другий розділ «Проектування програмного забезпечення тестування знань англійської мови». Проведено постановку задачі, визначено усі необхідні вимоги до моделі програмного забезпечення тестування знань англійської мови. Було створено необхідну модель. Проаналізовано шляхи реалізації даної моделі, на основі даного аналізу було обрано найбільш слушні методи подальшої розробки. Було вирішено розробити модель веб-сторінки, що має за мету швидко, зручно та надійно тестувати знання англійської мови користувачів. Архітектура веб-сайту має передбачати незалежність реалізації системи від апаратної платформи і серверної операційної системи.

Третій розділ «Розробка програмного забезпечення тестування знань англійської мови». Було досліджено та прийнято до уваги усі переваги використання мови розмітки сторінок HTML, мови веб-програмування JavaScript та її бібліотек JQuery та Ajax, а також мову для оформлення структурованих документів CSS. За допомогою використання даних технологій було створено програмний продукт, що повністю виконує необхідний функціонал поєднаний зі зручним та інтуїтивно зрозумілим користувальницьким інтерфейсом.

Висновки та пропозиції

У процесі розробки програмного забезпечення тестування знань англійської мови було проаналізовано доцільність знання англійської мови в сучасному світі. Визначено, що знання англійської мови на необхідному рівні дає можливість вести комунікацію зі світом, покращувати свої професійні навички, можливість вирушити у подорож, або навіть просто потренувати свій мозок.

Досліджено процес та форми тестування та визначено найголовніші переваги тестування як форми контролю знань.

Спроектовано модель процесу тестування знань англійської мови, при цьому було визначено необхідні функціональні потреби програмного забезпечення. Завдяки цьому було створено технічне завдання, на основі якого було розроблено програмний продукт. У процесі розробки було розглянуто доцільність використання технологій веб-програмування, таких як мова розмітки сторінок HTML, мова оформлення структурних документів CSS, мова веб-

програмування JavaScript та її бібліотеки JQuery та Ajax, та визначено, що дані технології повністю підходять для поєднання інтуїтивно зрозумілого користувальницького інтерфейсу з необхідним функціоналом.

Розроблене програмне забезпечення повністю відповідає заявленому функціоналу, проте є відкритим до будь-яких модифікацій, тому проект є досить гнучким та може бути задіяний для різноманітних цілей. Також за будь-яких умов завдяки гнучкості використаних технологій є можливість налаштувати інтерфейс, так як того потребуватиме користувач.

Список використаних джерел

1. Аванесов В. С. Научные основы тестового контроля знаний / В. С. Аванесов. – М. : Исследовательский центр, 1994. – 135 с
2. Гершунский Б.С. Компьютеризация в сфере педагогики. – М.: Педагогика, 1987. – 264 с.
3. Петлюшкин А.В., HTML в Web-дизайне. – СПб.: БВХ-Петербург, 2004. – 400 с.: ил.
4. Габова О.В. Тестування - одна з форм діагностики та перевірки успішності навчання / О.В. Габова, А.А. Русаков. - Київ : САММІТ-КНИГА, 2005. - 340 с.
5. Майоров А.Н. Теорія і практика створення тестів для системи навчання (як вибирать, створювать і використовувати тести для цілей в навчанні): підручник / А.Н. Майоров. - Москва - «Интеллект- центр», 2001 - 296 с.
6. Булах І. Є. Створюємо якісний тест : навч. посіб. / І. Є. Булах, М. Р. Мруга. – К. : Майстер-клас, 2006. – 160 с.
7. Гуревич Р.С., Кадемія М.Ю. Інформаційно-телекомунікаційні технології в навчальному процесі та наукових дослідженнях: навчальний посібник для студентів педагогічних ВНЗ. – Вінниця: ООО “Планер”, 2005. – 366 с.
8. Федорук П.І. Адаптивні тести: статистичні методи обробки результатів тестового контролю знань / // Математичні машини і системи: навч. посібник / П.І. Федорчук; за ред. О.В. Русланової - Редакційно-видавничий відділ (РВВ) НТУ «ХПІ» , 2007. -138 с.

9. Тестування як засіб перевірки й оцінки знань, умінь і навичок учнів - Режим доступу: http://osvita.ua/school/lessons_summary/proftech/39187/

10. Порядок створення web-сайта - Режим доступу: http://www.usif.org.ua/uk/nashitenderi/652-2017-08-28-14-38-28.html

11. Інформаційно-новинний портал про програмування – Режим доступу: http://www.e-helper.com.ua/ .

12. Веб-документація - Режим доступу: https://developer.mozilla.org/uk/ .

13. The ad-blocking usage report 2019 British Edition. Undertaken by YouGov, commissioned by eyeo. – Режим доступу: https://eyeo.com/2019-uk-ad-blocking-usage-report/.

14. Справочник по HTML - Режим доступу: http://htmlbook.ru/html.

15. DevDocs API Documentation - Режим доступу: https://devdocs.io/.

16. Testing knowledge base - Режим доступу: https://strongqa.com/qa-portal/knowledge-base.

17. 50+ кращих доповнень до Bootstrap – Режим доступу: https://habr.com/ru/company/dataart/blog/258301//.

18. GUI Testing Tutorial: User Interface (UI) TestCases with Examples – Режим доступу: https://www.guru99.com/gui-testing.html.

РЕЦЕНЗІЯ
на випускну кваліфікаційну роботу
студента 2м курсу 5 групи ФОАІС
освітнього ступеня «магістр»
спеціальності 121 «Інженерія програмного забезпечення»
Деремешка Максима Михайловича
на тему: «Розробка програмного забезпечення тестування знань англійської мови»

Випускна кваліфікаційна робота Деремешка Максима Михайловича, яку подано на рецензію, виконана у відповідності до наукової тематики кафедри та у встановлений термін. Актуальність теми магістерської випускної кваліфікаційної роботи полягає у створенні інтерфейсів користувачів за допомогою сучасних технологій створення інтерфейсів, які дозволяють зручно користуватися всіма функціями програмного забезпечення для управління діяльністю деканату.

Робота складається із вступу, в якому розкриті цілі, актуальність випускної кваліфікаційної роботи, опису проведених досліджень предметної області, розробленого технічного завдання, в якому зазначено вимоги до створеної бази даних, проведено аналіз існуючих технологій розробки інтерфейсів та вибір технології JQuery для створення інтерфейсу, а також опис процесу розробки, програмна реалізація, інструкції користувача, висновки, перелік джерел інформації. Робота містить: 10 рисунків роботи з інтерфейсом користувача.

Студент використовуючи бібліотеку JQuery розробив інтерфейс веб-додатку для тестування знань англійської мови. Суттєві зауваження щодо оформлення та структури випускної кваліфікаційної роботи відсутні. Вважаю, що робота виконана на достатньому рівні, задовольняє вимогам Вищої школи та може бути допущена до захисту, а її автору, при успішному захисті, може бути присвоєно кваліфікацію освітнього ступеня «магістр» за спеціальністю «Інженерія програмного забезпечення».

К.т.н., Доцент кафедри
комп'ютерних наук КНТЕУ **Демідов П.Г.**