ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «РОЗРОБКА SPA-ДОДАТКУ, ВИКОРИСТОВУЮЧИ REACT НА ТЕМУ «ОНЛАЙН-ПОРТАЛ ДЛЯ ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ»

Виконала: студентка 5 курсу, групи ППЗ-52 спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

_____ Шпильова О.В._____

(прізвище та ініціали)

Керівник __Залива В.В.____ _____

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Напрям підготовки – 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

Негоденко О. В.

“____” _____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Шпильова Олеся Вячеславівна

(прізвище, ім’я, по батькові)

1.Тема роботи: «Розробка SPA-додатку, використовуючи REACT на тему «Онлайн-портал для вивчення англійської мови»

Керівник роботи: асистент кафедри ІПЗ, аспірант Залива В. В.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 12 березня 2021 року, №65.

2. Строк подання студентом роботи 01 червня 2021 року.

3. Вихідні дані до роботи: SPA-додаток, використовуючи REACT

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Огляд та аналіз засобів проєктування та розробки онлайн порталу з вивчення англійської мови

4.2 Аналіз та проєктування адаптивного веб- орієнтованого навчального середовища

4.3 Розробка і тестування додатку

5. Перелік графічного матеріалу
5.1 Титульний слайд.

5.2 Об’єкт, предмет та дослідження.

5.3 Актуальність вибраної теми.
5.4 Автоматизація роботи онлайн-школи
5.5 Програмні засоби для вирішення поставленої задачі
5.6 Алгоритм роботи програми
5.7 Схема бази даних
5.12 Результати роботи ПЗ
5.13 Висновки

6. Дата видачі завдання 19 квітня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.2021	Виконано
2	Вимоги до встановленого додатку	23.04.2021	Виконано
3	Оцінка якості тестування до систем	29.04.2021	Виконано
4	Концепція та архітектура додатку	02.05.2021	Виконано
5	Вступ, висновки, реферат	05.05.2021	Виконано
6	Розробка презентаційних матеріалів	06.05.2021	Виконано
7	Попередній захист роботи	11.05.2021	
8	Здача роботи	01.06.2021	

Студент _____ Шпильова О.В
(підпис) (прізвище та ініціали)

Керівник роботи _____ Залива В. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи: 75 с., 27 рис., 5 табл., 2 додатки, 22 джерела.

Об'єкт дослідження. Робота онлайн порталу з вивчення англійської мови.

Предмет дослідження. Можливість автоматизації роботи онлайн-школли та функції автоматизованих систем, які задовольняють вимоги до автоматизації роботи онлайн-занять

Метою роботи є створення онлайн порталу з вивчення англійської мови, в якому буде максимально доступна інформація, викладена доступною мовою

Актуальність роботи полягає у тому, що він робить спрощеним процес вивчення англійської мови, в якому буде максимально доступна інформація, викладена доступною мовою. На ньому буде можливість звернутися до викладача особисто і отримати відповіді на питання.

Для досягнення даної мети мають бути вирішені наступні **завдання**:

1. Розробити навчальний веб-ресурс SPA-додаток, використовуючи REACT, який буде відображати відповідний дизайн в залежності від віку дитини і залежний від рівня її підготовки алгоритм вивчення предмету.

2. Реалізувати фільтри підбору сценарію навчання на початку роботи з ресурсом.

3. Зробити інтерфейс максимально зрозумілим для дітей з різними рівнями комп'ютерної грамотності.

4. Дослідити та проаналізувати існуючі аналоги.

5. Протестувати роботу системи в цілому.

Стислий опис результатів дослідження: реалізовано веб-додаток, який готовий до роботи у форматі подання матеріалів для вивчення та взаємодії користувачів під час вивчення англійської мови.

Ключові слова: ВЕБ-ДОДАТОК, АНГЛІЙСЬКА МОВА, ВИВЧЕННЯ МОВИ, REACT, SPA-додаток.

ЗМІСТ

ВСТУП

Онлайн-сервіси дистанційної освіти стали сьогодні актуальними не тільки через необхідність у зв'язку з виникненням пандемії. На сайтах, які можуть бути для людини доброю альтернативою електронним іграшкам, оскільки вони мають на меті виконання своїх передусім розвивальної, та навчальної, виховної функцій, міститься навчальний контент з величезною кількістю вправ, спрямованих на формування і розвиток інтелектуальних здібностей дітей, на засвоєння і повторення знань. Такого спрямування веб-ресурс є конкурентноспроможним на ринку освітніх послуг, він плекає покоління розумних, ерудованих, працьовитих, які прагнуть до саморозвитку і націлених на творчість людей.

Актуальність теми дослідження. В даний час великої популярності набули електронні освітні ресурси. Процес інформатизації суспільства зробив необхідністю створення освітніми установами власних ресурсів, що представляють собою комунікаційний центр, який дозволяє проводити всілякі маніпуляції з інформацією, спрямованою на вирішення проблем освітнього характеру. У зв'язку з цим, та освітня установа або керівник приватного освітнього проєкту, які прагнуть бути конкурентоспроможними, мати привабливий імідж і ефективну систему роботи з інформацією для забезпечення внутрішніх потреб освітнього процесу, стикається з проблемою створення свого інтернет-представництва, а саме зі створенням власного Веб-ресурсу, онлайн порталу з вивчення англійської мови, за допомогою якого можна навчати дітей різного віку за різними принципами, методиками і програмами

Об'єкт проєктування – методи та засоби створення онлайн порталу з вивчення англійської мови.

Предмет проєктування – онлайн портал з вивчення англійської мови.

Область застосування – педагогіка, психологія, додаткова дошкільна та шкільна освіта, родина, розвивальні гуртки, дитячі заклади, самоосвіта дорослих.

Метою даної роботи – є створення онлайн порталу з вивчення англійської мови, в якому буде максимально доступна інформація, викладена доступною мовою. Портал підійде тим, для кого вивчення англійської мови – це страшніше, ніж похід до стоматолога. Тим, хто вже багато разів починав, але не зміг досягнути бажаного результату, тому, що наштовхувався на нерозуміння якихось понять. На цьому онлайн-порталі буде можливість звернутися до викладача особисто і отримати відповіді на питання.

SPA-додаток, використовуючи REACT, був написаний, аби зекономити час для викладача англійської. Інформація в додатку викладена так само, як би викладач розповів учням в класі, тож учень може ознайомитися з матеріалом, поставити питання, якщо вони є та закріпити інформацію вправами.

Маршрутизація в додатку була реалізована за допомогою модулю react-router-dom та об’єктів Router, Switch, Route, Link

Основні завдання дипломної роботи:

1. Розробити навчальний веб-ресурс SPA-додаток, використовуючи REACT, який буде відображати відповідний дизайн в залежності від віку дитини і залежний від рівня її підготовки алгоритм вивчення предмету.

2. Реалізувати фільтри підбору сценарію навчання на початку роботи з ресурсом.

3. Зробити інтерфейс максимально зрозумілим для дітей з різними рівнями комп’ютерної грамотності.

4. Дослідити та проаналізувати існуючі аналоги.

5. Протестувати роботу системи в цілому.

6. Зробити висновки до виконаної роботи.

Методи дослідження. Для вирішення поставлених завдань використовувалися такі методи: аналіз літератури, згідно з досліджуваною темою; аналіз інформаційних систем, Інтернет-ресурсів, необхідних для створення та публікації веб-ресурсу.

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ ЗАСОБІВ ПРОЄКТУВАННЯ ТА РОЗРОБКИ ОНЛАЙН ПОРТАЛУ З ВИВЧЕННЯ АНГЛІЙСЬКОЇ МОВИ

1.1 Призначення та область застосування об’єкту проєктування

Проблема створення навчального освітнього простору актуальна і відповідає сучасним цілям і завданням розвитку освіти. У зв’язку з цим і виникає необхідність створення проєкту освітнього розвивального середовища як умова самореалізації дитини. Проєкт спрямований на організацію спеціально спроєктованого освітнього простору в ДНЗ через ігри для дітей дошкільного та молодшого шкільного віку та системи, в якій будуть задіяні і на ігровому освітньому рівні пов’язані всі учасники освітньо-розвивального процесу.

Ігровий освітній простір розглядається як комплекс освітніх, психолого-педагогічних, морально-етичних, екологічних, фізкультурно-оздоровчих заходів, що забезпечують дитині психічне і фізичне благополуччя, комфортне, пізнавальне та побутове середовище в дитячому садку.

Головна ідея проєкту-забезпечити реалізацію моделі ігрового освітнього простору, що забезпечує умови для самореалізації дитини, підвищення якості освіти дошкільника відповідно до сучасних вимог дошкільної освіти…

По-перше, відмова від копіювання звичайних форм організації навчання. Це буде освітня діяльність, стійкий інтерес дитини до освоєння середовища, стимулюючої пізнавальну активність дітей, що вимагає від дитини самостійності в прийнятті рішень і оптимальному виборі способів самоосвіти, де і з ким.

По-друге, це ігрова діяльність дітей за інтересами, бажанням і вибором простору, яка буде забезпечувати розвиток у дошкільнят рухових, розумових і мовних навичок, розвивати творчі здібності дитини, так як вона буде відбуватися за власним «сценарієм» дитини. У процесі гри дитина реалізує свої пізнавальні, естетичні та моральні потреби, прокладе власний освітній маршрут.

Це позиція підвищення якості освітнього процесу за рахунок створення ігрового простору з урахуванням освітніх завдань, що надає дитині можливість

вибору виду діяльності, ігор, іграшок і творчого партнера-дорослого. Здійснення освітньої діяльності у формах, специфічних для дітей дошкільного віку, у формі гри. Дитині потрібно не почути інформацію, а прожити, тобто пограти в щось в сюжетно-рольовій грі, театралізованої грі, де дитина образно бере участь, перевтілюється в когось. Дитині потрібно вирішувати якісь завдання, поставлені перед ним в дидактичній, розвиваючій грі, в грі з правилами.

Новизна проєкту – це побудова новітньої педагогічної форми освітнього процесу, варіативного розвиваючого освіти, орієнтованого на можливість вільного вибору дітьми ігор, матеріалів, видів активності, учасників спільної діяльності та спілкування, як з дітьми різного віку, так і з дорослими, а також свободу у вираженні своїх почуттів і думок. Це визначення компонентів і принципів проєктування ігрового освітнього простору, педагогічної підтримки індивідуальної траєкторії освіти, забезпечення особистісної та соціальної успішності кожної дитини.

Інформаційно-навчальним сервісом (ІС), реалізуваноти обмежена кількість освітніх функцій або курсів. Наприклад, його можна розглядати як корпоративну, розподілений реалізацію якогось одного навчального курсу з фіксованим обмеженням функціональних можливостей.

З усіх існуючих засобів організації електронного навчання можна виділити наступні основні групи:

- авторські програмні продукти (Authoring Packages),
- системи управління контентом (Content Management Systems, CMS),
- системи управління навчанням (Learning Management Systems, LMS),
- системи управління навчальним контентом (Learning Content Management Systems, LCMS).

CMS підтримують функціонування каталогів бібліотек і депозитаріїв навчальних матеріалів. LMS допомагають організувати навчальний процес і оцінити його результати (управлінський облік, управління контингентом і система оцінки). На відміну від LMS, LCMS концентруються на завданнях управління вмістом контентом інформаційно-навчальних середовищ і сервісів, а

не процесом навчання. і орієнтовані не на управлінців від освіти і навчаються, а на розробників контенту, фахівців з методологічної компонуванні контенту, керівників проєктів навчання.

Дослідники О. М. Карпенко, А.В. Лук'янова, А.В. Абрамова, В.А. Басова вказують наступні тенденції гейміфікації в галузі освіти [14]:

- розробка комп'ютерних навчальних Ігор;
- гейміфікація систем управління навчанням (LMS) та навчальним контентом (LCMS);
- гейміфікація як спосіб підвищення мотивації учнів.

Інформаційно навчальні веб-середовища і сервіси більш всього відносяться до авторських програмних продуктів у силу того, що їх розробка здійснюється під замовлення і не передбачає, як правило, використання стандартних, типових, універсальних програмних продуктів. Функціонально веб-середовища і сервіси можуть повторювати окремі можливості або цілі LMS і LCMS продуктів. Найбільший інтерес представляє реалізація функцій управління навчальним контентом.

1.2 Огляд способів і засобів вирішення поставлених задач

Веб-ресурс – це сучасна, доступна для широкого кола користувачів платформа для розміщення контенту, наукових і публіцистичних статей, публікацій. Він надає можливість продемонструвати особливості організації, розповісти про досягнення і надати всім користувачам цікаву для них інформацію [8]. У зв'язку з важливістю наявності веб-ресурсу для будь-якої організації та проєкту, зупинимося докладніше на самому понятті веб-ресурсу і його видах.

З точки зору користувача веб-ресурс розуміється як сукупність веб-сторінок, що містять необхідну користувачеві інформацію і елементи управління, а процес роботи з ними – як перехід між сторінками, керований даних, заданих у відповідь на запит. Традиційний термін "сайт" відноситься до

користувацької сторони веб-ресурсів [22]. Таким чином, веб-ресурс найчастіше називають веб-сайтом. Розглянемо різні визначення даного поняття.

У різних джерелах даються схожі визначення, які можна представити наступним чином: Веб-сайт-логічний вузол системи WWW, який визначається своїм URL-ім'ям і представляє собою організовану сукупність HTML-сторінок, пов'язаних один з одним гіпертекстовими посиланнями [17].

Відзначимо, що діапазон можливостей веб-ресурсів з точки зору використання визначає їх головну характеристику з точки зору внутрішньої структури і технології розробки – двоїстий характер [2].

З одного боку, основним змістом веб-ресурсів є дані, заради представлення яких певний ресурс створюється. З цього випливають завдання проєктування контенту і організації взаємодії з користувачем. З іншого боку, сучасні веб-ресурси представляють собою програмні комплекси, і інформація, що пред'являється користувачеві, є результат роботи відповідних програм. Звідси випливають завдання проєктування цих ресурсів як програм [23].

В даний час існує величезна різноманітність веб-ресурсів різних типів. У зв'язку з цим, перед проєктуванням важливо точно визначитися з видом Веб-ресурсу з урахуванням певних потреб. Для цього розглянемо існуючі класифікації веб-сайтів.

Веб-ресурси, що представляють собою найбільш актуальний клас сучасних інформаційних систем, одночасно є одним з найбільш складних в розробці класом, що об'єднує в собі властивості цілого ряду комп'ютерних продуктів. Це породжує специфічні проблеми розробки і вимагає вибору адекватних цій специфіці засобів.

У процесі створення веб-сторінок можуть застосовуватися різноманітні програмні інструменти:

Текстові (символьні) редактори загального призначення (наприклад, «Блокнот»).

Спеціалізовані текстові HTML-редактори, в середовищах яких використовується колірне виділення тегів, атрибутів і їх значень, а також

виконується синтаксичний контроль мовних конструкцій (наприклад, UniRed, Bred, HtmlPad, CoffeCup HTML Editor).

1.3. Постановка задачі на розробку

Автоматизована веб-орієнтована розвивально-навчальна система освітнього середовища створюється для певного підприємства чи організації. Проте, є багато спільних рис в структурі різних підприємств, а також в типах зв'язків (функціональних, інформаційних, зовнішніх) між елементами цієї структури. Це дозволяє сформулювати єдині принципи і шляхи побудови інформаційних систем для підприємства чи організацій.

Перед нами поставлені такі завдання:

1. Розробити навчальний адаптивний веб-ресурс, який буде відображати відповідний дизайн в залежності від віку дитини і залежний від рівня її підготовки алгоритм вивчення предмету.

2. Реалізувати фільтри підбору сценарію навчання на початку роботи з ресурсом.

3. Зробити інтерфейс максимально зрозумілим для дітей з різними рівнями комп'ютерної грамотності.

- Для дітей від 3 до 5-ти років
- Для дітей від 5 до 8-ти років
- Для дітей від 8 років і більше

Можна виділити такі етапи створення і функціонування (життєвого циклу) ІС веб-орієнтованого навчального середовища (Рисунок 1.1).

На першому етапі проводиться обстеження об'єкта, вивчаються форми вхідних та вихідних документів, методики розрахунків необхідних показників. Проводяться також науково-дослідні роботи щодо оцінки реалізації вимог замовника: здійснюється підбір необхідних засобів моделювання процесів, які комп'ютеризуються, пошук відповідних програмних засобів, оцінка альтернативних проєктів.

Рисунок 1.1 – Схема етапів розробки веб-порталу

В процесі розробки інформаційного забезпечення визначається:

– склад інформації (перелік інформаційних одиниць, необхідних для розв’язання комплексу задач);

– структуру інформації та закономірності її перетворення, тобто правила формування показників і документів;

– характеристики руху інформації (обсяг та інтенсивність потоків, маршрути руху, часові характеристики);

– характеристики якості інформації (систему кількісних оцінок значущості, повноти, своєчасності, вірогідності інформації);

– способи перетворення інформації;

– уніфіковану систему первинної документації;

– масиви інформації, що використовуються для розв’язання задач управління;

– методичні й інструктивні матеріали для ведення документів.

На цьому ж етапі розробник погоджує із замовником вимоги до ІС веб-орієнтованого навчального середовища, її функції, необхідні витрати на

розробку, терміни виконання. Завершується перший етап складанням звіту про проведені роботи, на основі якого в подальшому буде розроблено технічний проєкт.

На другому етапі формується технічне завдання, яке є підставою для розробки інформаційної системи і приймання її в експлуатацію. Воно визначає основні вимоги до самої системи та процесу її розробки і розробляється для системи в цілому. Додатково можуть розроблятись технічні завдання на окремі частини ІС.

На третьому етапі розробляється концепція інформаційної бази, створюється інфологічна і датологічна моделі, формуються вимоги до структури інформаційних масивів, технічних засобів. Вказуються характеристики програмного забезпечення, систем класифікації та кодування. Результатом даного етапу є комплект проєктної документації (технічний проєкт). В ньому вказується постановка задачі, алгоритм її розв'язання, описується інформаційне, організаційне, технічне та програмне забезпечення, тощо. Після затвердження технічного проєкту розробляється робочий проєкт (внутрішній).

Одночасно з розробкою проєкту веб-орієнтованого навчального середовища створюються класифікатори техніко-економічної інформації на основі погодженої системи класифікації і кодування техніко-економічної інформації.

На четвертому етапі здійснюється розробка програмного забезпечення у відповідності з проєктною документацією для веб-орієнтованого навчального середовища. Результатом цього етапу є готовий програмний продукт – веб-орієнтоване навчальне середовище.

На п'ятому етапі проводиться перевірка програмного забезпечення на предмет відповідності вимогам, вказаним в технічному завданні. Дослідна експлуатація (тестування) дозволяє виявити недоліки, які можуть появитись при експлуатації системи. На цьому ж етапі проводиться підготовка персоналу до роботи в інформаційній системі. Навчання персоналу здійснюється або силами розробника, або за допомогою спеціальних курсів. Підготовлюється робоча

документація, проходять приймальні випробування, і система здається в експлуатацію замовнику.

Шостий етап організовується на підставі гарантійних зобов'язань розробника. У цей період здійснюється сервісне обслуговування системи, усуваються недоліки, які можуть бути виявлені при експлуатації, і завершуються роботи по даному проєкту.

Всі етапи розробки і впровадження ІС повинні бути обумовлені у відповідних угодах між замовником і розробником, а також у технічному завданні.

Кабінет веб-орієнтованого навчального середовища в залежності від вікової групи та серйозності завдання передбачає реєстрацію або від батьків з дитиною або дитини старшого віку з "емуляцією" захисту персональних даних (для даного проєкту це критично, оскільки успішність це тут – ще й психологічний фактор щодо розголошення). У кабінеті веб-орієнтованого навчального середовища має бути структуризація визначених системою та підтверджених у виборі освітніх рівнів, ступеня проходження та успішності. Для контентної складової веб-орієнтованого навчального середовища фахівцями дошкільної освіти та вчителями початкової школи спеціально повинні підбиратись і розроблятись навчальні та розвиваючі вправи, керуючись провідними програмами, що діють в освітніх установах.

Одночасно над контентною складовою сайту повинні працювати спеціалісти – ілюстратори, дизайнери, створюючи спільно з педагогами і психологами контент, корисний для розвитку дітей і відповідний їх віковим особливостям. Велика робота повинна бути проведена з технічного боку – зі створення сайту. Планується після створення проєкту постійно доповнювати зміст сайту новими вправами та навчальними матеріалами, статтями та рекомендаціями фахівців з дошкільного та підліткового навчання та розвитку.

Висновки до першого розділу

Інформаційно-навчальне веб-середовище – це середовище, спеціально сформоване для того, щоб у ній здійснювалася освітня діяльність за кількома або багатьма напрямками, програмами. Приставка «веб» означає застосування дистанційних технологій і визначає формат подання навчального контенту.

Інформаційно-навчальним сервісом (ІС) реалізована обмежена кількість освітніх функцій або курсів. Наприклад, його можна розглядати як корпоративний, розподілений проєкт для реалізації якогось одного навчального курсу. Спочатку проводиться обстеження об’єкта, вивчаються форми вхідних та вихідних документів, методики розрахунків необхідних показників. Проводяться також науково-дослідні роботи щодо оцінки реалізації вимог замовника: здійснюється підбір необхідних засобів моделювання процесів, які комп’ютеризуються, пошук відповідних програмних засобів, оцінка альтернативних проєктів. Потім, після розробки технічного завдання, розробки, впровадження та тестування і налаштувань – супровід проєкту.

РОЗДІЛ 2 АНАЛІЗ ТА ПРОЄКТУВАННЯ АДАПТИВНОГО ВЕБ- ОРІЄНТОВАНОГО НАВЧАЛЬНОГО СЕРЕДОВИЩА

2.1 Моделювання інтерфейсу та функціоналу веб-ресурсу

Основний принцип управління освітнім та розвивальним контентом проєктованого сайту веб-орієнтованого навчального середовища «Education.googlle.com.ua» заключається в розділенні розвивального та навчального контенту на окремі елементи з подальшим вибудовуванням ієрархії цих елементів і зв'язків між ними. Таким чином, елементи контенту стають сущностями опису ІС і знаходять відображення в інформаційних моделях, таких як, наприклад, er модель (entity relationship diagram, ERD). Приклад фрагмента er-моделі з елементами навчального контенту приведений на рисунку 2.1.

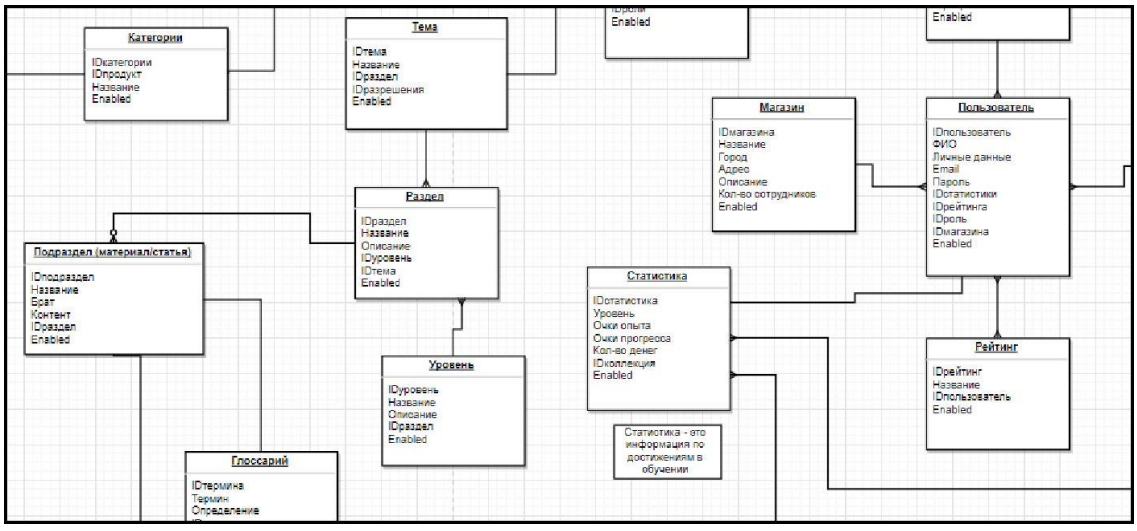


Рисунок 2.1 – Фрагмент Er моделі проєкту

Питання організації текстового та ігрового контенту в веб-орієнтоване навчальних середовищах і сервісах розглянуті на прикладі інформаційно веб-середовища умовної компанії GMF. Гейміфікація процесу навчання у інформаційно-навчального веб-середовища:

- полегшує засвоєння знань та набуття практичних навичок, підвищує інтерес співробітників до компанії та її продукції;

- покликане допомогти дітям та їх батькам використовувати інформаційно-навчальні ресурси, що надаються у веб-середовищі, а також ігрові сценарії в щоденній роботі з клієнтами і партнерами, підвищити якість обслуговування клієнтів і особистий обсяг продажів.

Вимоги до організації контенту інформаційно-освітніх систем визначаються на початкових стадіях життєвого циклу інформаційної системи. Форми і види ігрових завдань при використанні в веб-орієнтованому навчальному середовищі з використанням гейміфікації визначаються переліком передбачуваних концепцією ігор (ігрових моментів) і скрипти. Наприклад, при створенні інформаційно-навчального веб-середовища для дітей у нашому проєкті був запропонований сценарій проведення ігор у iovs карта віртуального подорожі того, кого навчають, до світу тварин, рослин, тощо.

Відповідно до заданих сценарієм та переліком ігор проєктуються форми і рекомендуються види ігрових завдань, задаються вимоги до контенту, правила його організації та використання, порядок застосування в ігрових ситуаціях, а також визначаються категорії користувачів, зокрема категорії що навчаються, права та правила доступу які навчаються до навчального і ігровому контенту.

Традиційними моделями життєвого циклу ІС прийнято вважати [5]:

- функціонально модульна модель або каскадний модель;
- поетапна модель з проміжним контролем;
- спіральна модель.

Функціонально модульний підхід (каскадна модель) діє на підставі принципу алгоритмічної декомпозиції, відповідно до якого процеси ІС діляться на модулі з функціональної належності, передбачає строго послідовний порядок дій. Недоліком підходу є жорстка вимога виконання послідовності робіт. Проблеми, що виникають на який-небудь з стадій можуть бути вирішені лише в рамках поточної стадії і не зачіпають попередні стадії. Фактична відсутність зворотного зв’язку призводить до деформацій початкового задуму і реалізації системи. Перехід на наступний етап означає повне завершення робіт на попередньому етапі.

Позитивною стороною застосування каскадного підходу (Рисунок 3) є те, що на кожному етапі формується набір проєктної документації, який відповідає критерієм повноти, що дозволяє планувати витрати і терміни завершення робіт. Каскадний модель добре зарекомендувала себе у випадках, коли на початку розробки системи можна точно і повно сформулювати вимоги до неї. Вартість внесення змін до проєкту висока, тому що доводиться чекати завершення всього проєкту. Тим не менш, фіксована вартість проєкту часто переважує мінуси підходу.

Рисунок 2.4 – Каскадна модель ЖЦ проєкту

Розвитком каскадної моделі є поетапна модель з проміжним контролем. В цьому випадку розробка ведеться ітераціями з циклами зворотного зв’язку між етапами. Міжетапне коригування дозволяють зменшити трудомісткість процесу розробки у порівнянні з каскадної моделлю. Час життя кожного з етапів розтягується на весь період виконання інформаційного проєкту.

Спіральна модель приділяє увагу початковим етапів розробки – з передпроєктною (може розбиватися на кілька послідовних етапів) та проєктування. Реалізування концепції та технічних рішень перевіряється і обґрунтовується за допомогою створення прототипів (макетування). Кожен виток спіралі передбачає створення нової версії продукту або будь-кого з його компонента з уточненими характеристиками і цілями.

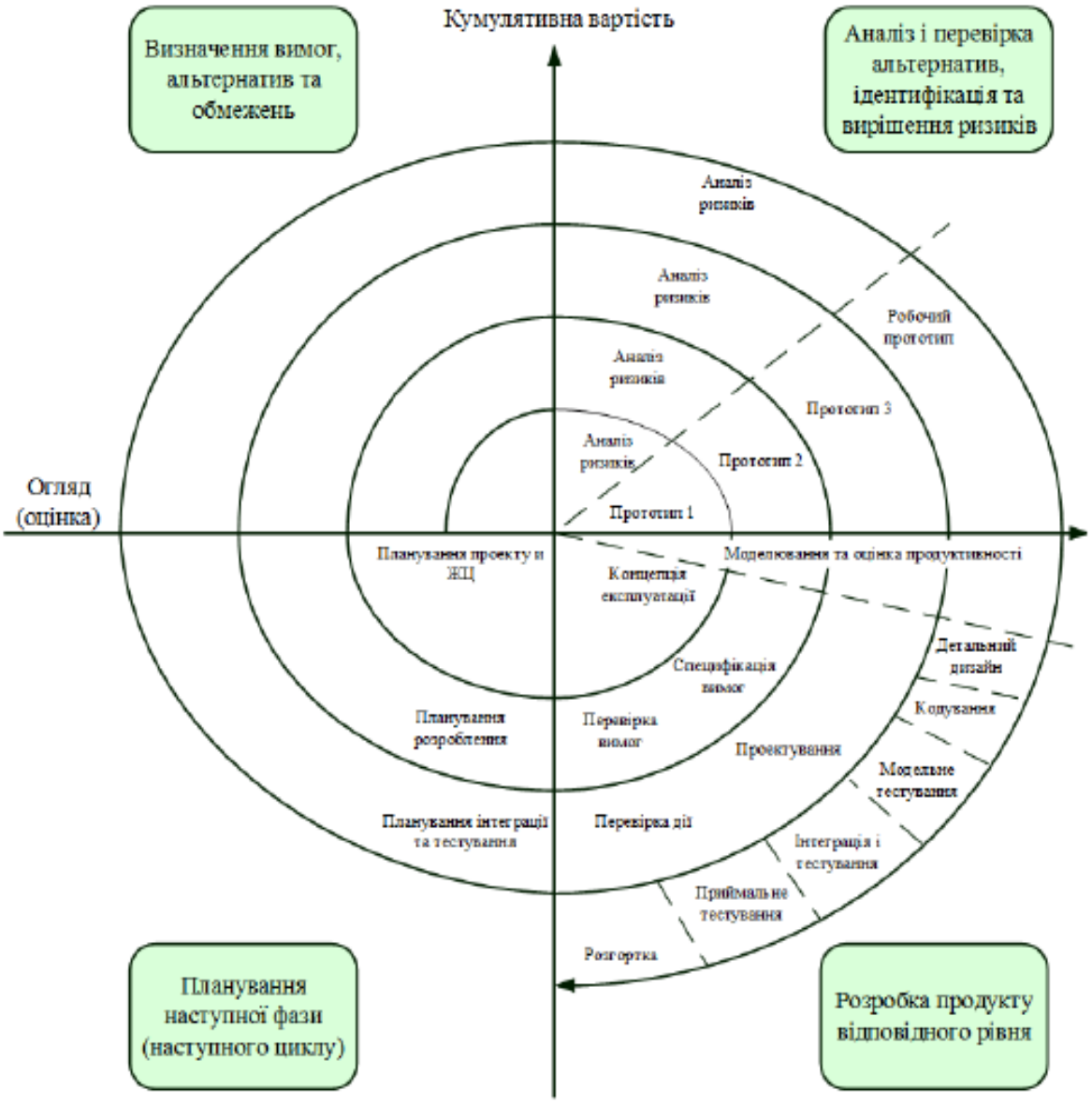


Рисунок 2.5 – Спіральна модель ЖЦ проєкту

Особливостями моделі є: можливість переходу на наступний етап, не чекаючи повного завершення роботи на поточному; контроль якості поточного етапу; планування роботи наступного витка спіралі. Завдання надання користувачам системи працездатною версії продукту з подальшою активізацією процесів уточнення та доповнення вимог. Результат послідовна конкретизація деталей проєкту, обґрунтування вибору варіанту реалізації.

В сучасній практиці використання моделей і методів розробки ІС є багатоваріантним. Не існує єдино правильною стратегії роботи з інформаційним проєктом, як не існує єдино вірних для всіх проєктів стартових умов, вимог, схем

оплати. Наприклад, методи Agile не завжди можливо застосовувати повсюдно, повністю самостійно або в повному обсязі через низку зовнішніх причин і факторів. До них слід віднести на неможливості забезпечення гнучкого фінансування, неготовність замовника до форм співпраці, передбачуваних методами Agile, рівень і якість (компетенції) кадрового складу та інші особливості проєкту.

2.2. Формування та аналіз вимог до об’єкту проєктування

Модель життєвого циклу інформаційної системи залежить від специфіки інформаційної системи і включає стадії (етапи процесу розроблення інформаційної системи), основні результати виконання робіт на кожній стадії. ключові події. Стадії це основні періоди життєвого циклу системи.

Стадії описують стан, динаміку і результати розвитку системи, структуру робіт, забезпечують стале управління інформаційним проєктом і технічними процесами.

Стадії розробки ІС проєкту включають [10]:

- передпроєктне обстеження,
- проєктування,
- створення інформаційної системи,
- введення в експлуатацію,
- експлуатація інформаційної системи,
- виведення з експлуатації.

Переліки стадій, представлені в різних джерелах, мають деякі відмінності, пов’язані з тією чи іншою мірою деталізації моделі життєвого циклу інформаційної системи, використанням термінів, зіставленням термінів зі стадіями життєвого циклу.

Так згідно з ISO / IEC 15288 життєвий цикл ІС складається з: формування концепції аналізу потреб, вибору концепції та проєктних рішень; розробки проєктування системи; реалізації виготовлення системи; експлуатації введення в

експлуатацію та використання системи; підтримки забезпечення функціонування системи; зняття з експлуатації припинення використання, демонтажу, архівування системи [11].

Схема процесу розробки інформаційної системи, запропонована Григор'євим Ю.А., включає наступні етапи: виявлення інформаційних потреб кінцевих користувачів; концептуальне проєктування; розробка архітектури ів; логічне проєктування; зневадження і тестування прикладних програм [12].

Ian Sommerville у своїй книзі "Software Engineering" (розробка програмного забезпечення) обіграє наступні види діяльності процесу створення і експлуатації програмного забезпечення:

– Опис програмного забезпечення, в ході якого клієнти та розробники програмного забезпечення (інженери-програмісти) визначають в результаті спільної роботи функціональність створюваного програмного забезпечення і що накладаються обмеження,

– Розробка програмного забезпечення, в ході якого проєктується і програмується програмне забезпечення,

– Перевірка достовірності (відповідності) програмного забезпечення, в ході якого контролюється склад і функціональність програмного забезпечення з метою переконатися, що воно саме таке, яке хотів клієнт,

– Подальша розробка програмного забезпечення, в ході якого програмне забезпечення змінюється, щоб задовольняти мінливих вимогам клієнта, вимогам ринку і мінливих умов бізнесу [6].

Стадії або етапи життєвого циклу не залежать від типу моделі життєвого циклу, фактично використовуваного підходу до розробки, інформаційних систем. Моделі (підходи) можуть використовувати різні версії декомпозиції життєвого циклу.

- проєктування інформаційної системи.

Результатом передпроєктного етапу «є фіксація того, що отримає замовник у разі згоди фінансувати проєкт; дата закінчення розробки продукту та графік виконання робіт; вартість виконання проєкту. Виконана на даному етапі робота

дозволяє відповісти на запитання, які з вимог замовника можуть бути задоволені та умови їх задоволення. Цей етап передбачає дослідження бізнес-процесів компанії-замовника та інформації, необхідної для їх виконання (сутностей, їх атрибутів і зв'язків). На цьому етапі створюється інформаційна модель, а на наступному за ним етапі проєктування модель даних» [8].

Вся інформація про майбутню систему, зібрана на предпроєктному етапі, формалізується на етапі проєктування і уточнюється на наступних етапах. Як правило, замовник не відразу формує вимоги до системи, може змінювати свою позицію в процесі узгодження. Важливо, щоб всі основні домовленості і уявлення про системі були уточнені і формалізовані до початку етапу розробки системи.

2.3. Проєктування інформаційної системи

Вибір архітектурного стилю та проєктування веб-орієнтованого навчального середовища відбувався таким чином: обирались структурні складові, які далі розподілялися за перевагою значимості при проходженні всього шляху від незареєстрованого користувача системи до переможця серед успішно пройшовших курс користувачів.

Для об'єктно-орієнтованого моделювання проблемної сфери широко використовується уніфікована мова моделювання UML (Unified Modeling Language), яка розроблена групою провідних комп'ютерних фірм світу OMG (Object Management Group) і фактично є стандартом з об'єктно-орієнтованих технологій.

Мова UML реалізована багатьма фірмами-виробниками програмного забезпечення в рамках CASE-технологій, на-приклад Rational Rose (Rational), Natural Engineering Workbench (Software AG), ARIS Toolset (IDS prof. Scheer) та ін.

Уніфікована мова моделювання є графічною мовою для візуалізації, специфікації, конструювання і документування систем, у яких велика роль належить програмному забезпеченню.

За допомогою мови UML можна розробити детальний проєкт створюваної системи, що відображає:

– концептуальні елементи: системні функції і бізнес-процеси;

– конкретні особливості реалізації: класи, написані спеціальними мовами програмування, схеми баз даних, програмні компоненти багаторазового використання.

Система об'єктно-орієнтованих моделей відповідно до нотацій UML включає наступні діаграми:

1) *діаграму варіантів використання* (Use-case diagram), що відображає функціональність ІС у вигляді сукупності послідовностей, що виконуються, транзакцій;

2) діаграму класів об'єктів (Class diagram), що відображає структуру сукупності взаємозалежних класів об'єктів аналогічно до ER-діаграми функціонально-орієнтованого підходу;

3) *діаграми станів* (State chart diagrams), кожна з яких відображає динаміку станів об'єктів одного класу і пов'язаних з ними подій;

4) *діаграми взаємодії* об'єктів (Interaction diagrams), кожна з яких відображає динамічну взаємодію об'єктів у рамках одного прецеденту використання;

5) *діаграми діяльності* (Activity diagrams), що відображають потоки робіт у взаємозалежних прецедентах використання (можуть декомпозуватися на більш детальні діаграми);

6) *діаграми пакетів* (Package diagrams), що відображають розподіл об'єктів за функціональними чи забезпечувальними підсистемами (можуть декомпозуватися на більш детальні діаграми);

7) *діаграму компонентів* (Component diagram), що відображає фізичні модулі програмного коду;

8) *діаграму розміщення* (Deployment diagram), що відображає розподіл об’єктів між вузлами обчислювальної мережі.

Діаграма варіантів використання виявляє основні бізнес-процеси як послідовності транзакцій, які повинні виконуватися повністю, коли виконання відособленої підмножини дій не має значення без виконання всієї послідовності. Варіанти використання ініціюються із зовнішнього середовища користувачами ІС, так званими акторами. На цьому рівні моделювання не розкривається механізм реалізації процесів. Наведені сутності мають графічні позначення, наведені в табл. 2.1.

Таблиця 2.1 – Об’єкти, використовувані в діаграмі варіантів використання

Найменування об’єкта	**Опис**
Актор	Зовнішній користувач процесу
Варіант використання	Бізнес-процес
Пойменована стрілка	Позначення події

Актор ініціює виконання варіанта використання і отримує від нього результати. Взаємодія (асоціація) актора з варіантом використання здійснюється внаслідок події, яка позначається пойменованою стрілкою. Один актор може брати участь – у декількох варіантах використання, а в одному варіанті використання може бути зайнято декілька акторів.

У реалізації варіанта використання можливе виділення декількох потоків подій:

– основний потік подій, який приводить до необхідного результату найбільш коротким шляхом;

– альтернативні потоки подій.

Основний і альтернативний потоки подій у моделі варіантів використання описуються у вигляді неформальних текстових коментарів.

Декілька варіантів використання можуть мати спільну частину, що виділяється в самостійний варіант використання, з яким установлюються відносини використання (uses).

З іншого боку, певні варіанти використання можуть бути розширені деталями. У такому разі створюється додатковий варіант використання, з яким встановлюються відносини розширення (extends).

Діаграми класів об’єктів відображають статичну структуру класів об’єктів. Ця діаграма розглядає внутрішню структуру предметної сфери, ієрархію класів об’єктів, статичні зв’язки об’єктів.

Класи об’єктів можуть мати різні стереотипи поведінки: об’єкти сутності, керівні об’єкти, інтерфейсні об’єкти (табл.2.3).

Об’єкти, відображені в діаграмі класів об’єктів, пов’язуються статичними відносинами, які відбивають постійні зв’язки між об’єктами незалежно від виконання конкретного бізнес-процесу. До статичних відносин відносяться узагальнення, агрегація, асоціація об’єктів.

Таблиця 2.2 – Об’єкти, використовувані в діаграмі класів об’єктів

Найменування об’єкта	**Опис**
Об’єкт-сутність	Пасивний об’єкт, над яким виконуються операції обробки процесу
Керівний об’єкт	Активний об’єкт, що координує виконання функцій
Відносини асоціації	Відносини типу 0.. 1:1 . 1 :M; M:N. Відносини можуть бути пойменованими. 0...1 – необов’язковість зв’язку; * – множинність зв’язку
Відносини узагальнення	Відносини спадкоємства
Відносини агрегації	Відносини «ціле – частина»

Діаграми станів відображають поведінку об’єктів одного класу в динаміці, зв’язок станів об’єктів з подіями і визначає:

- які типові стани проходить об’єкт;

- які події ведуть до зміни стану об’єкта;

- які дії об’єкт виконує, коли він отримує повідомлення про зміну стану;

- як об’єкти створюються і знищуються (вхідні і вихідні точки діаграми).

У табл.2.3 наведені використовувані в діаграмі станів поняття і їх графічні позначення.

Таблиця 2.3 – Об’єкти, використовувані в діаграмі станів

Найменування об’єкта	**Опис**
Вхідна точка	Визначає подію, яка утворює початковий стан об’єкта. У точку входу не можна перейти зі стану об’єкта
Стан	Передає ситуацію, протягом якої виконується безперервна діяльність або об’єкт перебуває в стаціонарному положенні. Стан визначається як набір значень атрибутів і відносин, пов’язаних із об’єктом. Ім’я стану повинне бутті унікальним лише всередині класу об’єкта, для якого воно визначається
Перехід станів	Визначає зміну в стані об’єкта, яка відбувається внаслідок події, що виникла в той час, коли об’єкт перебував у даному стані. Кожен перехід станів повинен мати унікальне ім’я
Вихідна точка	Визначає завершення існування об’єкта. З точки виходу немає переходу стану

З кожним станом пов’язана одна або більше подій, які можуть його змінити. Для стану задаються імена всіх пов’язаних з ним переходів в інші стани.

Перехід станів описується наступними атрибутами.

Призначення – стан об’єкта, в який перейде об’єкт після переходу стану.

Виклик – ім’я події, яка спричиняє перехід станів. Імена подій повинні бути ідентичними у визначенні класу і стану. Події, що спричиняють перехід, можуть бути або зовнішніми, здійснюваними акторами, або внутрішніми, пов’язаними з поведінкою інших об’єктів, або тимчасовими, пов’язаними із закінченням заданого інтервалу часу.

Умова переходу – це логічний вираз, пов’язаний з атрибутами об’єкта, які повинні бути перевірені для вибору переходу стану. Умова переходу задається в тому випадку, якщо відбувається подія, внаслідок якої може відбутися неоднозначний перехід станів. Умови переходів для одного початкового стану повинні бути взаємовиключними. Дія – це атрибут, що інформаційно описує сутність дії, яка повинна виконуватися під час переходу станів. Цій дії відповідатиме певна процедура, що реалізовує метод класу об’єктів.

Перехід станів графічно позначається лінією, на якій задається принаймні один із наступних атрибутів: виклик, умова переходу, дія.

Для кожного варіанта використання може бути побудована модель динамічної взаємодії об’єктів, яка подається однією з двох форм:

- формою діаграми послідовностей (sequence diagram), що показує послідовність взаємодій на графі;

- формою кооперативної діаграми (collaboration diagram), що показує взаємодію об’єктів у табличній формі.

У діаграмі послідовностей взаємодія об’єктів відтворюється у вигляді стрілки між об’єктами, яка відповідає події або повідомленню від одного об’єкта до другого, який викликає виконання об’єкта, що реагує на подію (повідомлення). Номер стрілки відповідає номеру події в послідовності.

Діаграма кооперативної поведінки подається в табличному вигляді за наступними правилами:

1. У стовпчиках таблиці вказуються об’єкти всіх типів використання, що беруть участь у реалізації варіанта. Порядок розташування активних і пасивних об’єктів довільний і повинен бути зручним для розуміння моделі. Актори прецеденту використання відображаються на правій і лівій межах таблиці.

2. По горизонталі проводяться пойменовані стрілки, що відбивають взаємодію (комунікацію) об’єктів у рамках однієї операції. Ця стрілка означає, що перший об’єкт у рамках виконуваної операції посилає повідомлення другому об’єкту про необхідність виконання дії. У разі отримання повідомлення другий об’єкт виконує дію.

3. На перетині рядків і стовпчика вертикально відображається умовний відрізок часу, протягом якого виконується та або інша дія над об’єктом.

Діаграми взаємодій не відбивають детально порядку виконання операцій в частині розгалужень, циклічних повторень, паралельності дій.

Діаграма діяльностей передбачає відображення даних можливостей. Під діяльністю мається на увазі певна робота, яка може бути розбита на сукупність дій. Діаграма діяльностей може відбивати взаємодію об’єктів з декількох варіантів використання, що, зокрема, реалізовують окремо стандартні й альтернативні шляхи обробки об’єктів. Блок, відповідний одній діяльності, може

відбивати декілька подій і бути декомпозований аналогічно до блоку структурного підходу.

Поняття і їх графічні позначення, використовувані в діаграмі діяльностей, наведені в табл.2.4.

Таблиця 2.4 – Об’єкти, використовувані в діаграмі діяльностей

Найменування	Опис
Діяльність	Робота, яка може бути розбита на сукупність дій
Потік	Від діяльності до діяльності
Поділ потоку	Поділ потоку на діяльності, виконувані паралельно або довільно
Розв’язання	Вибір одного з потоків залежно від події
Синхронізація	Перехід до однієї діяльності після виконуваних паралельно
Ітерація	Повторне виконання діяльності
Вихід	Визначає завершення існування

В об’єктному підході пакет містить безліч взаємопов’язаних класів об’єктів і відповідає поняттю “підсистема” в структурному підході. Один варіант використання може потребувати класи об’єктів з різних пакетів. Клас об’єктів зазвичай призначається одному пакету, але з позиції досягнення різних підцілей може входити до складу різних пакетів.

Пакетна технологія групування класів об’єктів дозволяє спростити:

- розробку й експлуатацію ІС;
- гнучку адаптацію типових компонентів з позиції їх повторного використання;
- оптимізацію клієнт-серверної архітектури ІС.

Зазвичай ІС розбивається на функціональні й забезпечувальні пакети. Функціональні пакети, відповідні вирішуваним проблемам (завданням), об’єднуються в один спільний пакет “Предметна область”. Кожен пакет, у свою чергу, може бути розбитий на підпакети відповідно до семантичної близькості й щільності взаємодії класів об’єктів. Зазвичай пакети предметної області містять ієрархії узагальнення й агрегації. Класи об’єктів, потрібні в декількох підсистемах, виділяються в самостійні пакети. В одному пакеті визначається не більше 20 компонентів, зазвичай 5 – 15.

Із точки зору забезпечувальних пакетів ІС розбивають на п’ять основних пакетів:

- інтерфейс, об’єкти якого реалізують функції взаємодії користувачів з ІС щодо введення-виведення інформації й обміну повідомленнями між підсистемами;
- база даних, об’єкти якої виконують доступ до даних у зовнішній пам’яті;
- управління завданнями, об’єкти якого здійснюють функції диспетчеризації і маршрутизації обробки об’єктів, наприклад у системі управління робочими потоками;
- утиліти, об’єкти якого здійснюють допоміжні функції, наприклад перетворення форматів даних;
- забезпечувальні пакети, тобто ті, що працюють за принципом “клієнт-серверної” архітектури, виконують серверні функції для функціональних об’єктів-клієнтів. Таким чином, забезпечувальні пакети звільняють користувача від знання деталей програмно-технічної реалізації ІС.

Діаграма компонентів відображає залежності програмних компонентів, які подаються у вигляді початкових програмних кодів об’єктів, що відкомпілювалися, і виконуваних кодів. Один компонент, як правило, відповідає програмному коду одного пакета класів об’єктів.

Компонент у своєму складі має інтерфейсний клас об’єктів, через який здійснюється доступ до решти класів об’єктів компоненти. За допомогою інтерфейсу об’єкти інших компонентів звертаються не до конкретних об’єктів даного компоненту, а до його інтерфейсного об’єкта. Таким чином спрощується взаємодія компонентів між собою, коли в разі доступу до компоненти з інших компонентів не потрібно знати внутрішню структуру цієї компоненти. Компонент, до якого здійснюється звернення, може бути не об’єктно-орієнтованою. Достатньо, щоб у такого компонента був лише один інтерфейсний клас об’єктів, який транслює запити до компонента у виклики звичайних процедур. У компонентів може бути декілька інтерфейсів.

Класичними результатами аналізу, проведеного на передпроєктному етапі і етапі проєктування, є:

- ієрархія функцій, яка розбиває процеси на складові частини;

- модель сутність зв’язок (ERD), яка описує сутності, атрибути і зв’язки між ними.

До результатів також слід віднести діаграми потоків даних і діаграми життєвих циклів сутностей, наявність яких дозволить уникнути помилок при подальшому проєктуванні та розробці системи.

На етапі проєктування формуються моделі даних логічна і фізична. Отримана раніше інформаційна модель спочатку перетворюється в логічну, а потім у фізичну модель даних. При розробці схеми бази даних можлива зміна інформаційної моделі через неочевидності окремих деталей проєкту на передпроєктному етапі.

Проєктування передбачає визначення користувачів системи та їхніх прав, проєктування екранних форм і звітів, які будуть забезпечувати виконання запитів до даних і багато іншого, що стосується обліку конкретної середовища або використовуваних технологій, конфігурації апаратних засобів, використовуваної архітектури, способів обробки даних і т. п.

У разі проєктування інформаційно-навчальних систем і сервісів на цих етапах відбувається формування повного уявлення про склад, вид і формах відео повчального контенту, знаходить своє відображення у всіх що формуються на етапі моделях.

До терміну, крім обов’язкового визначення, можна додати зображення у форматі і об’єму, заданими системою.

Редагування термінів та їх визначень можливо тільки при переході до матеріалів відповідного підрозділу (звідки був внесений термін) в режимі адміністрування.

Використання додаткових інформаційних джерел для формування контенту підрозділів. В Інтернеті знаходиться багато інформації, яка має відношення до конкретної галузі знань. Можна вельми успішно використовувати її при

створенні контенту шляхом безпосереднього приєднання частині інформації до вже існуючого контенту або введення посилань на Web джерела. В першому випадку інформація, що залучається, повинна бути піддана значній переробці як в частині змісту, так і в частині подання матеріалу: приведена до формату, використовуваному для формування конкретного навчального модуля (підрозділу).

Перш ніж внести до підрозділу матеріали, знайдені в Інтернет, зробити їх доступними, а значить рекомендувати їх клієнтам і співробітникам, Спробуйте відповісти на запропоновані нижче питання. Отримані відповіді дозволять вам визначити надійність джерела, достовірність і новизну інформації. Переліку запропонованих питань зроблений за стандартами, заведеним у всьому світі, і відображає наявність певної культури використання веб-матеріалів. В кінцевому підсумку рішення приймає розробник контенту.

Основним методом виявлення сфер знання і незнання в будь-якій предметної області залишається статистичний аналіз даних виконання тестів, в нашому випадку через використання ігрових ситуацій гейміфікації. Професором С.Аванесовим було виділено 18 різних форм знань [15]. Тестуванню може бути піддана будь-яка з них. Наприклад:

- знання назв, імен;
- знання сенсу назв і імен;
- знання визначень; асоціативні знання;
- причинні знання, знання причинно-наслідкових відносин, принципів класифікації;
- наукові знання;
- практичні знання тощо.

Проєктування тестів (наборів ігрових завдань) для проведення Ігор має на увазі створення ігрових завдань (питання та відповіді) різних форм і типів.

Ігрове завдання може використовуватися залежно від дидактичних цілей:

– для контролю результатів освоєння підрозділу\підрозділів:

– для самоперевірки. Тести супроводжуються внутрішньої зворотним зв’язком: результати тестування аналізуються самим які навчаються. В поточному релізі не застосовується.

Визначають п’ять загальних вимог до ігрових завдань читання:

– валідність;

– визначеність (загальнозрозумілість);

– простота;

– однозначність;

– надійність.

Валідність це адекватність. Розрізняють змістовну і функціональну валідність: перша це відповідність ігрового завдання змістом контрольованого навчального матеріалу, друга відповідність типа игрового завдання оцінюваного рівню діяльності (рівнем труднощі матеріалу).

Виконання вимоги визначеності (загальнодоступності) ігрового завдання необхідно не тільки для розуміння кожним які навчаються того, що він має виконати, але й для виключення правильних відповідей, що відрізняються від еталона. Вимога простоти ігрового завдання означає, що тест має складатися із завдань одного рівня і форми, тобто не має бути комплексним і складатися з декількох завдань різного рівня труднощі або різних форм. Однозначність визначають як однаковість оцінки якості виконання ігрового завдання різними експертами. Для виконання цієї вимоги кожне ігрове завдання повинен мати еталон\правильну відповідь. Вимога надійності полягає в забезпеченні сталості результатів багаторазового тестування одного і того ж катованого.

При створенні ігрових завдань необхідно дотримуватися деяких загальних правил, принципів розробки ігрового завдання:

- визначення значущості тих, кого перевіряють знань у загальній системі тих, кого перевіряють знань (можна використовувати відповідність рівнями засвоєння матеріалу);

- взаємозв’язок змісту і форми ігрових завдань;

- змістовна правильність ігрових завдань;

- комплексність і збалансованість тесту;
- системність змісту тесту;
- варіативність змісту тесту;
- зростаюча трудність ігрових завдань.

Під варіативністю розуміється залежність тестових результатів від повноти відображення навчальної дисципліни в тесте. Отже, йдеться про ефективність тестування та достовірності отриманих результатів [13]. Трудність визначається у відповідність з матеріалом, описаним вище.

Логічні вимоги до змісту ігрового завдання. Питання ігрового завдання є смисловими твердженнями. Отже, логічними вимогами до формулюванні питання тесту є:

- визначеність,
- логічна несуперечливість,
- логічна і змістовна правильність,
- послідовність,
- обґрунтованість.

Зміст ігрового завдання має бути зорієнтоване на отримання від тестованого однозначного заключення. Основні терміни тестового завдання повинні бути явно і ясно визначені.

Формулювання ігрових завдань (питань) повинні:

- бути прагматично коректними і розраховані на оцінку рівня навчальних досягнень студентів по конкретному з підрозділом, розділу, темі,
- представлятися у вигляді коротких суджень,
- не включати прямі цитати з книг,
- покликані виявляти лише один, певний аспект (Ключове поняття, термін, правило, визначення тощо), тобто не повинні мати багатоцільову спрямованість;
- уникати вимоги від катованого розгорнутих висновків;
- мати не більше 10 слів, але без загрози спотворення понятійної структури ігровій ситуації.

Висновки до другого розділу

Вимоги до організації контенту інформаційно-освітніх систем визначаються на початкових стадіях життєвого циклу інформаційної системи. Форми і види ігрових завдань при використанні в веб-орієнтованому навчальному середовищі з використанням гейміфікації визначаються переліком передбачуваних концепцією ігор (ігрових моментів) і скрипти. Наприклад, при створенні інформаційно-навчального веб-середовища для дітей у нашому проєкті був запропонований сценарій проведення ігор у iovs карта віртуального подорожі того, кого навчають, до світу тварин, рослин, тощо.

Вибір архітектурного стилю та проєктування веб-орієнтованого навчального середовища відбувався таким чином: обирались структурні складові, які далі розподілялися за перевагою значимості при проходженні всього шляху від незареєстрованого користувача системи до переможця серед тих, хто успішно пройшли курс користувачів.

Проєктування передбачає визначення користувачів системи та їхніх прав, проєктування екранних форм і звітів, які будуть забезпечувати виконання запитів до даних і багато іншого, що стосується обліку конкретної середовища або використовуваних технологій, конфігурації апаратних засобів, використовуваної архітектури, способів обробки даних і т. п.

У разі проєктування інформаційно-навчальних систем і сервісів на цих етапах відбувається формування повного уявлення про склад, вид і формах відео повчального контенту, знаходить своє відображення у всіх що формуються на етапі моделях.

РОЗДІЛ 3 РОЗРОБКА І ТЕСТУВАННЯ ДОДАТКУ

3.1 Розробка системи

Перед тим, як перейти до розгляду розробки системи та її компонентів, зазначимо, що основний компонент додатку – App.js, компонент заголовку, який містить слоган, компонент, який є тілом додатку, компонент з цитатою, футер додатку, компонент з лексикою,

Компонент Drag and Drop ч1

Компонент Drag and Drop ч2

Компонент Drag and Drop ч3

Компонент Drag and Drop ч4

Компонент з окремою граматичною темою

Компонент з теорією з граматики

Компонент з контактами

Нижче перелічимо компоненти, їх основні властивості та події, при здійсненні яких виконуються запрограмовані дії (подієві процедури).

Компонент Form (екранна форма). Форма представляє не тільки вигляд вікна програми, але й сама є повноцінним компонентом з власними властивостями і подіями, хоча на палітрі компонентів її немає.

Основні властивості компонента Form:

Властивість Значення

Align – Режим вирівнювання об’єктів всередині форми

BorderStyle – Стиль обрамлення форми, поведінка форми (змінює розміри вікна)

Caption – Заголовок вікна форми

Color – Колір форми

Font – Атрибути шрифту форми

Ви можете встановити значення властивостей або у вікні властивостей об’єкта, або у програмі.

Приклад використання у програмі:

Form1.Color:= clRed; {визначення кольору форми}

Основна подія компон Form:

OnCreate Під час завантаження форми

Компонент Label (напис або мітка). Призначення – нести на собі напис. Можна використовувати для виводу відповіді або пояснення даних, що вводяться. Відноситься до групи Standard.

Основні властивості компонента Label:

Властивість Значення

Caption – Заголовок напису на екрані

Alignment – Режим вирівнювання тексту мітки

AutoSize – Якщо ця властивість має значення True, розміри мітки будуть автоматично змінюватися, щоб відповідати розмірам напису

Font – Шрифт для відображення тексту

Visible – Якщо ця властивість має значення True, то напис на екрані буде видно, а якщо False, то не видно

WordWrap – Якщо ця властивість має значення True, буде зроблено розбивку і перенесення незмінних рядків, а якщо False, то ні. Слід узгоджувати значення цієї властивості з властивістю AutoSize, яке в цьому випадку повинно мати значення False

OnClick – Відбувається, коли користувач клацає основною (лівою) кнопкою миші на мітці

Компонент Edit (поле редагування). Використовується для введення/виводу чисел і тексту до програми. Відноситься до групи Standard.

Основні властивості компонента Edit:

Властивість Значення

AutoSize – Якщо True, розміри компонента Edit будуть автоматично змінюватися при зміні розміру шрифту

BorderStyle – Стиль обрамлення поля

Text – Вміст рядка редагування

MaxLength – Максимальна кількість символів, що вводяться в поле

ReadOnly – Якщо True, текст редагування буде заборонено

Основна подія компонента Edit:

OnChange – Відбувається, коли користувач змінює текст

Компонент Button (командна кнопка). Використовується для завдання реакції на подію. Відноситься до групи Standard.

Основні властивості Button:

Властивість Значення

Caption – Назва кнопки

Height – Висота кнопки

Width – Ширина кнопки

Left – Відстань від лівої межі кнопки до лівої межі форми

Top – Відстань від верхньої межі кнопки до верхньої межі форми

Основна подія компон Button:

OnClick – Відбувається, коли користувач клацає основною (лівою) кнопкою миші на кнопці

Приклад використання у програмі:

Procedure TForm1.Button2Click(Sender: TObject);

Close {завершує виконання програми}

Немає необхідності запам’ятовувати і описувати всі властивості кожного об’єкта, розташованого на формі, оскільки значення всіх властивостей встановлюються за замовчуванням. На етапі проєктування їх можна змінити за допомогою вікна Object Inspector, а при написанні програмного коду після набору імені класу або об’єкта з точкою Delphi відобразить перелік властивостей, методів і подій, визначених для цього класу.

Для проєктування програми було використано мову графічного опису об’єктного моделювання. UML Була побудована поділом взаємодії зовнішнього актора з програмою у вигляді діаграми варіантів використання.

Під час проєктування було виділено одного актора "Користувач".

Користувач – це користувач програми, якому доступна можливість використовувати функціонал програми.

Найчастіше веб-додатки складаються як мінімум з трьох основних компонентів:

1) Клієнтська частина веб-програми – це графічний інтерфейс. Це те, що ви бачите на сторінці. Графічний інтерфейс відображається у переглядачі. Користувач взаємодіє з веб-додатком саме через браузер, клікаючи за посиланнями і кнопками.

2) Серверна частина веб-програми – це програма або скрипт на сервері, що обробляє запити користувача (точніше, запити браузера).

Найчастіше серверна частина веб-програми програмується на PHP. При кожному переході користувача за посиланням браузер надсилає запит до сервера. Сервер обробляє цей запит, викликаючи певний PHP-скрипт, який формує веб-сторінку, описану мовою HTML, і відсилає клієнту по мережі. Переглядач тут же відображає отриманий результат у вигляді чергової веб-сторінки.

3) База даних (БД, або система управління базами даних, СУБД) – програмне забезпечення на сервері, що займається зберіганням даних і їх видачею в потрібний момент. У разі форуму або блогу, що зберігаються в БД дані – це пости, коментарі, новини, і так далі. База даних розташовується на сервері. Серверна частина веб-програми (тобто PHP скрипт) звертається до бази даних, отримуючи дані, які необхідні для формування сторінки, запитаної користувачем.

Переглядач через Інтернет надсилає HTTP-запити веб-серверу. Веб-сервер викликає PHP-скрипт, написаний розробником веб-програми.

PHP-скрипт звертається до бази даних, якщо це потрібно. У результаті PHP скрипт повертає клієнтові веб-сторінку, яку відображає переглядач [4].

Основна мова, якою описується графічний інтерфейс веб-програми – це HTML. Ця мова описує структуру веб-сторінки, розташування на ній

компонентів. Вигляд веб-сторінок, їх стиль та схема кольорів описуються в таблицях стилів – CSS.

Для "пожвавлення" графічного інтерфейсу, додання йому динамічності, використовуються додаткові технології: скрипти JavaScript, а також вбудовані у веб-сторінку компоненти, створені на Flash, Java або Silverlight.

Відсутність необхідності повністю перезавантажувати сторінку після кожного отримання даних від сервера може суттєво прискорити роботу веб-програми. Така концепція має назву Asynchronous JavaScript and XML (асинхронний JavaScript і XML, Ajax). При використанні даного підходу динамічні запити до сервера відбуваються без видимого перезавантаження веб-сторінки: користувач не помічає, коли його браузер запитує дані [5].

Доменні об’єкти – це об’єкти в об’єктно-орієнтованих комп’ютерних програмах, що виражають сутності з моделі предметної області, що відноситься до програми, і реалізують бізнес-логіку програми. Доменні об’єкти інкапсулюють всю необхідну для програми інформацію про об’єкт предметної області [6].

Роль БД при створенні веб-додатків Згідно з класичним визначенням, база даних – це впорядкована сукупність інформації, що зберігається у вигляді безлічі, кожна з яких містить записи уніфікованого виду. Системи управління базами даних (СУБД) надають програмісту найпотужніший інструментарій для створення, оновлення та обробки великих обсягів інформації, що має складну структуру.

У класичній теорії виділяють три типи, три структури баз даних: ієрархічну, мережеву і реляційну. В даний час домінуюче положення займають реляційні бази даних. Лідером серед баз даних, що застосовуються для розробки WEB-додатків, на сьогоднішній день, безумовно, є MySQL. Головна перевага MySQL (плавно переходить у недолік:) – її простота. Як наслідок – найвища швидкість виконання SQL-запитів і необхідність явного програмування основних правил підтримки цілісності і непротирічності даних на рівні сервера додатків.

Серед інших баз даних, що застосовуються для WEB-розробок, відзначимо Oracle і PostgreSQL. PostgreSQL – вільна СУБД з відкритим кодом, орієнтована головним чином на роботу в UNIX-подібних системах [7].

Популярні фреймфорки для розробки веб-програм Фреймворк (framework) – це програмна оболонка, що дозволяє спростити і прискорити вирішення типових завдань, характерних для даної мови програмування. Саме слово framework означає "каркас" у перекладі з англійської. Дійсно, фреймворки і покликані бути готовими каркасами програм, на які тільки і залишається навісити стіни і вікна [8].

Сам фреймворк пропонує нам вже вбудовані класи для: роботи з базою даних, створення функціональних форм, валідації, логування та ін. Всі ці класи можете легко використовувати у всіх ваших проєктах, при цьому їх підключення і використання буде максимально простим.

Ще один з плюсів – структурування архітектури вашого додатку [9].

Maven Maven (мавен) – це інструмент для складання Java проєкту: компіляції, створення jar, створення дистрибутиву програми, генерації документації.

Переваги Maven:

1) Незалежність від OS. Збірка проєкту відбувається в будь-якій операційній системі. Файл проєкту один і той самий.

2) Керування залежностями. Рідко які проєкти пишуться без використання сторонніх бібліотек (залежностей). Ці сторонні бібліотеки часто теж у свою чергу використовують бібліотеки різних версій. Мавен дозволяє керувати такими складними залежностями. Що дозволяє вирішувати конфлікти версій і в разі необхідності легко переходити на нові версії бібліотек.

3) Можливе збирання з командного рядка. Таке часто необхідне для автоматичного збирання проєкту на сервері (Continuous Integration).

4) Хороша інтеграція з середовищами розробки. Основні середовища розробки на java легко відкривають проєкти які збираються з допомогою maven. При цьому часто проєкт налаштовувати не потрібно – він відразу готовий до

подальшої розробки. Як наслідок – якщо з проєктом працюють в різних середовищах розробки, то maven зручний спосіб зберігання налаштувань.

Налаштуваний файл середовища розробки і для збирання один і той же – менше дублювання даних і відповідно помилок.

5) Декларативний опис проєкту [10].

Vaadin – вільно поширюваний фреймворк для створення RIA-веб-додатків, що розробляється однойменною фінською компанією. На відміну від бібліотек на Javascript і специфічних плагінів для браузерів, Vaadin пропонує сервер-орієнтовану архітектуру, що базується на

Java Enterprise Edition. «Використання JEE дозволяє виконувати основну частину логіки програми на стороні сервера, тоді як технологія AJAX, що використовується на боці браузера, дозволяє інтерактивно взаємодіяти з користувачем, не відстаючи від аналогічних десктоп-додатків. Для відображення елементів користувацького інтерфейсу та взаємодії з сервером на боці клієнта Vaadin використовує Google Web Toolkit» [11].

Структурно Vaadin складається з серверного API, клієнтського API, набору компонентів користувацького інтерфейсу з обох сторін, механізму тем для оформлення інтерфейсу і моделі даних, що дозволяє пов’язувати серверні компоненти безпосередньо з даними. Можна застосовувати дві основні моделі розробки: на боці сервера і на боці клієнта (браузера).

Spring Framework (або коротко Spring) – «універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Spring Java-розробникам у проєктуванні; крім того, він надає добре документовані і легкі у використанні засоби вирішення проблем, що виникають при створенні додатків корпоративного масштабу» [11].

Між тим, особливості ядра Spring застосовні в будь-якому Java-додатку, і існує безліч розширень і удосконалень для побудови веб-програм на Java Enterprise платформі. З цих причин Spring набув великої популярності і визнається розробниками як стратегічно важливий фреймворк.

Одним з компонентів фреймворку є «Spring Security, який представляє з себе інструмент, що надає механізми побудови систем автентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework» [13].

На даний момент розроблено функціонал серверної та клієнтської сторони, який дозволяє адмініструвати і переглядати дані за допомогою веб-браузера.

Додаток включає в себе:

– Таблицю розкладу

– Таблицю навчального плану

– Для кожного елемента таблиць реалізовано 4 операції:

– Додавання

– Читання

– Редагування

– Вилучення

У додатку є можливість відображення розкладу, навчального плану, пошуку за викладачем, пошуку по групі.

Подальшим розвитком програми буде реалізація системи реєстрації, доопрацювання користувацького інтерфейсу, інтеграція з мобільними месенджерами.

Початковий код програми знаходиться на доданому електронному носії.

Створення сутностей

Для встановлення відповідності між елементами бази даних і POJO класами (визначення дивись далі) за допомогою фреймворку Hibernate використовується термін сутність.

Сутність (Entity) – простий POJO об’єкт, який потрібно зберегти в реляційній базі даних. Щоб оголосити клас POJO сутністю, його потрібно відзначити анотацією @Entity Кожна сутність має свій первинний ключ. Ви можете встановити властивість первинного ключа, що зберігається за допомогою анотації @ Id.

Сутність керує його станом, використовуючи або поля, або get і set методи. Це залежить від того, де використовується анотація.

Для кожної сутності доменної моделі створюється відповідний його POJO клас мовою java.

POJO (англ. Plain Old Java Object) – "простий Java-об’єкт у старому стилі", простий Java-об’ єкт, не успадкований від якогось специфічного об’єкта і не реалізує ніяких службових інтерфейсів понад тих, які потрібні для бізнес-моделі.

У поточній реалізації POJO клас складається з: полів конструкторів методів get методів set перевизначеного методу toString.

Проєктування бази даних

У рамках дипломної роботи графічний інтерфейс представлений для ролей Гість, Адміністратор. Гість має можливість переглядати таблицю розкладу і використовувати функціонал пошуку по групі і викладачу. Адміністратор має можливість редагувати і видаляти записи в таблицях, переглядати інформацію в таблиці навчальний план.

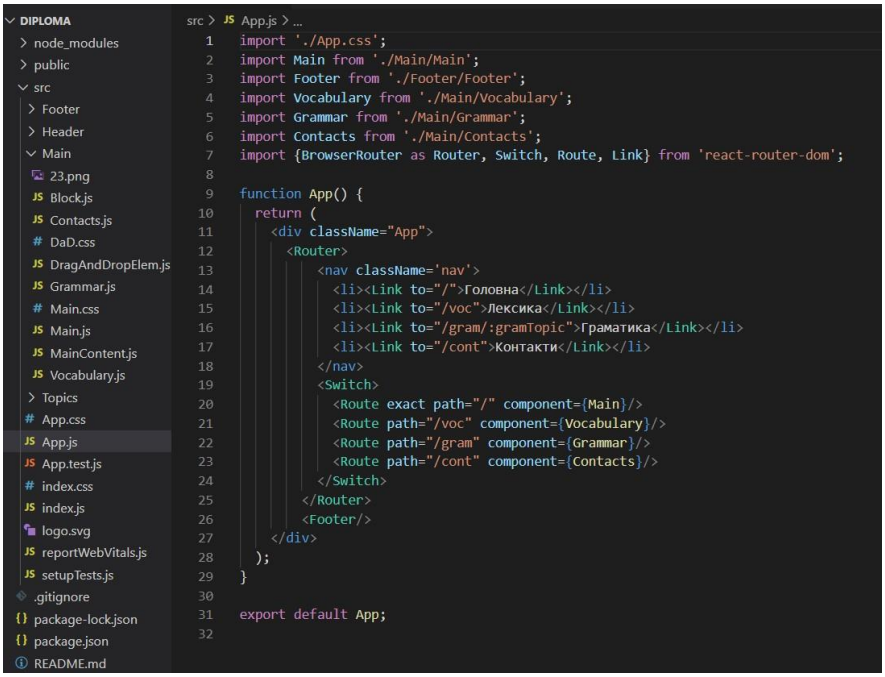


Рисунок 3.1. Основний компонент додатку – App.js

```
src > Header > JS Header.js > ...
1   import './Header.css'
2
3   function Header(){
4       return(
5           <div className="header">
6               <p className="slogan inl">Навчайся яскраво!</p>
7           </div>
8       )
9   }
10
11  export default Header;
```

Рисунок 3.2. Компонент заголовку, який містить слоган

```
src > Main > JS Main.js > [@] default
1   import './Main.css';
2   import Header from '../Header/Header';
3   import MainContent from './MainContent';
4   import Block from './Block';
5
6   function Main(){
7       return(
8           <div className='main'>
9               <Header/>
10              <Block/>
11          </div>
12      )
13  }
14
15  export default Main;
```

Рисунок 3.3. Компонент, який є тілом додатку.

```
# Footer.css   JS Footer.js   JS Block.js   # Main.css
src > Main > JS Block.js > [@] default
1   function Block(){
2       return(
3           <div className='main-block'>
4               <p className='text'> Кожен, хто перестає вчитися, старіє, – не важливо, в 20 або 80 років, – а будь-який інший, хто продовжує вчитися,
                залишається молодим. Найважливіше в житті – це зберегти мозок молодим. Генрі Форд
5               </p>
6           </div>
7       )
8   }
9
10  export default Block;
```

Рисунок 3.4. Компонент з цитатою

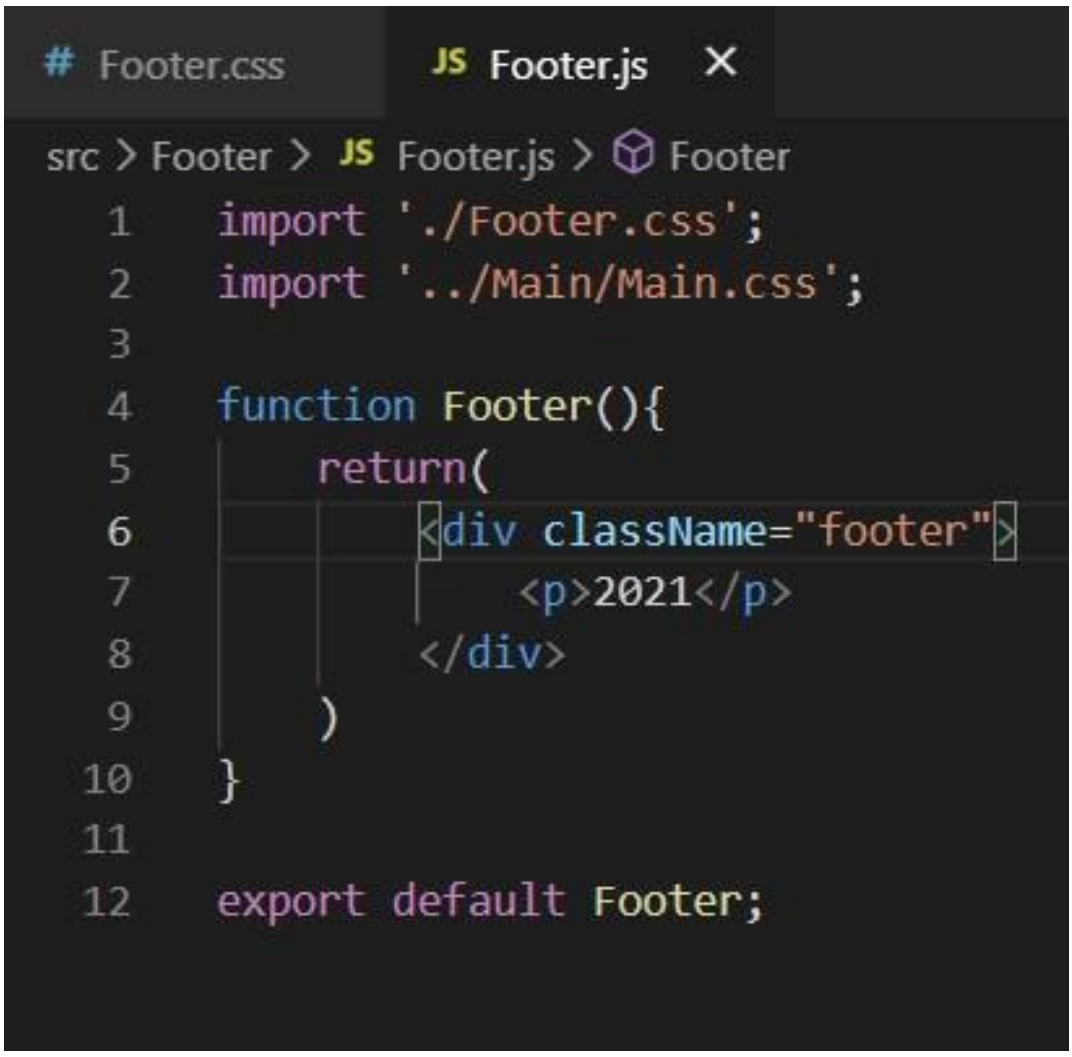


Рисунок 3.5. Футер додатку

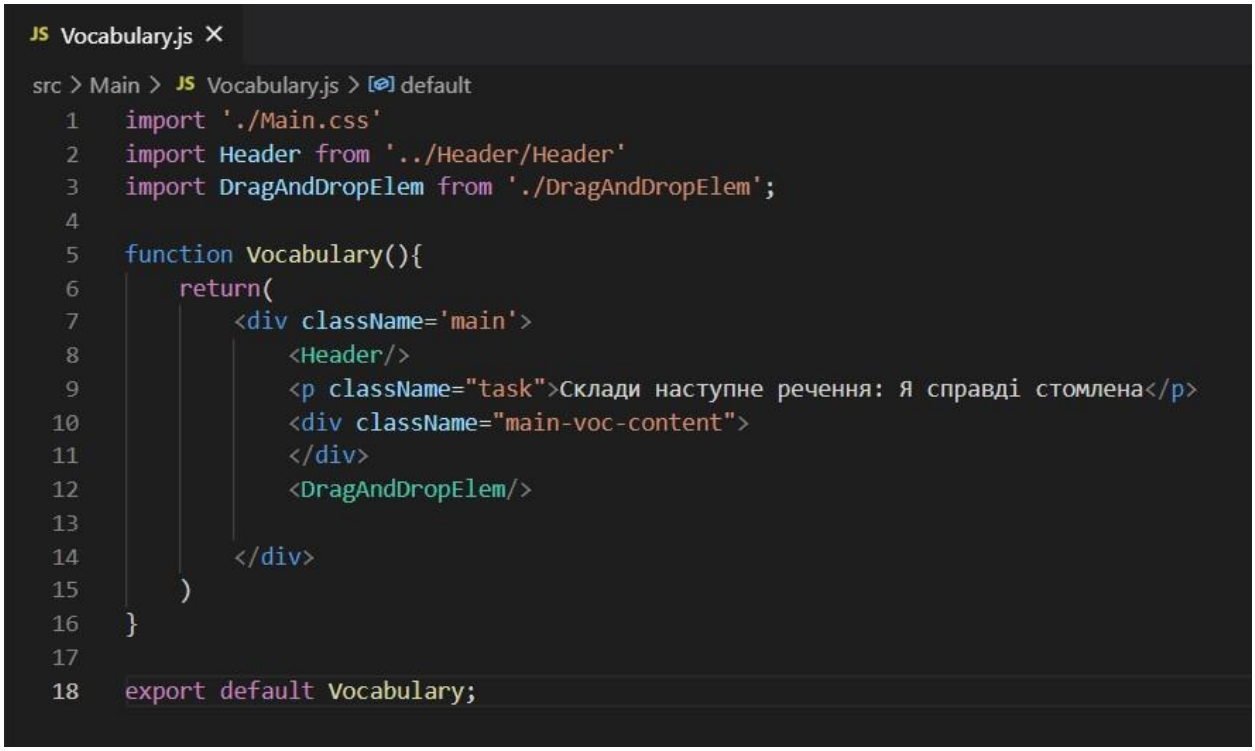


Рисунок 3.6. Компонент з лексикою

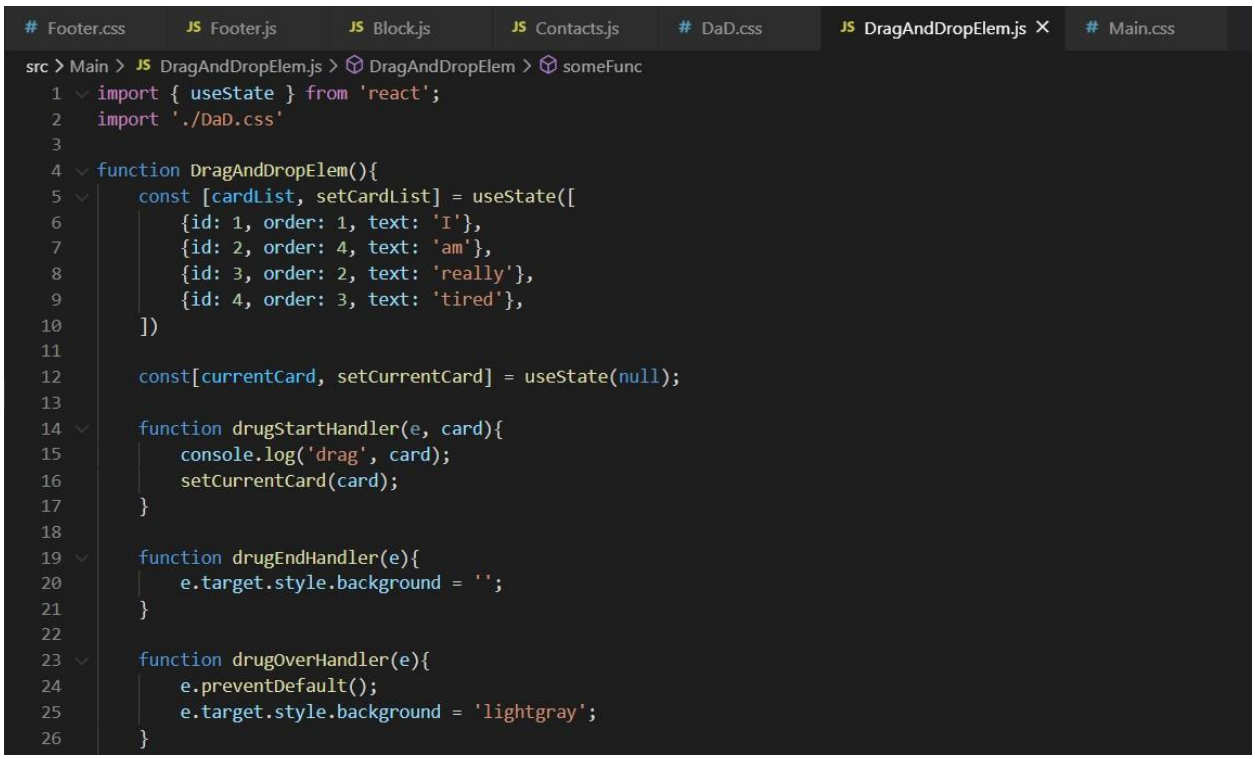


Рисунок 3.7. Компонент Drag and Drop ч1

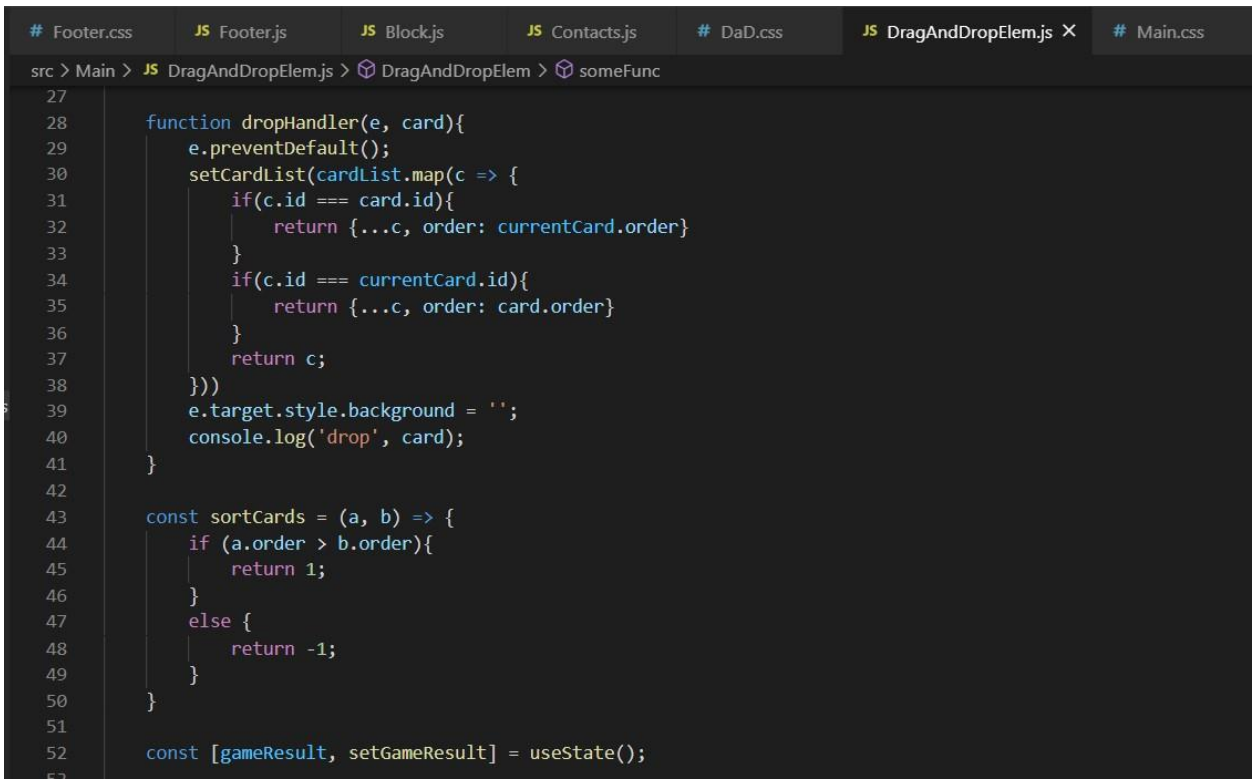


Рисунок 3.8. Компонент Drag and Drop ч2

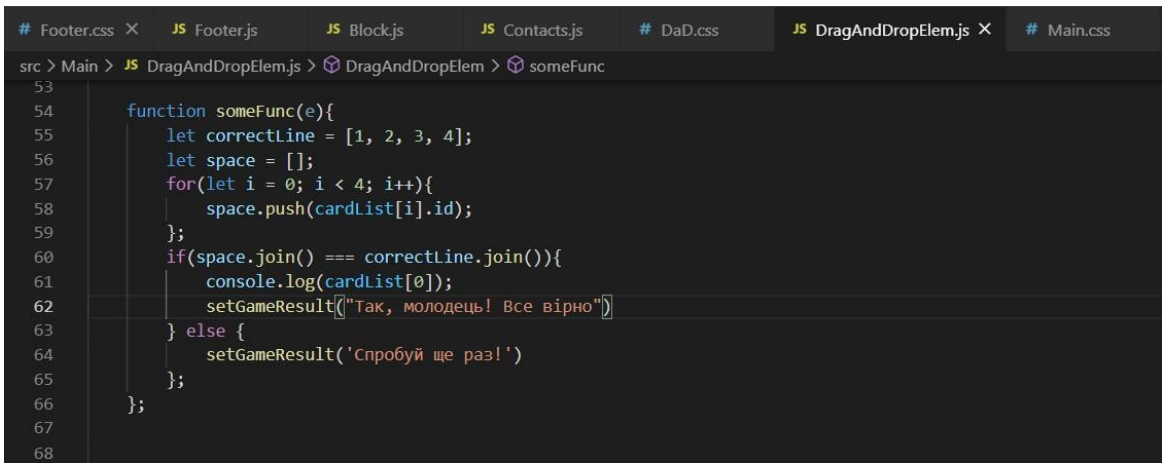


Рисунок 3.9. Компонент Drag and Drop ч3

У корені проекту файл index.php і .htaccess. У теці css лежать стилі у файлі main.css. У теці js - бібліотека jquery.js і головний файл програми main.js. У теці pages лежать html-файли з вмістом сайту - на кожну сторінку за одним файлом.

Жодних head, body, html, script тут немає - тільки розмітка, що відноситься до конкретної сторінки.

На нашому сайті буде одна-єдина фізична сторінка - index. Але нас цікавлять і такі адреси, як site.ru/about, site.ru/contacts та інше. Але сторінок about і contacts докорінно нашого сайту немає. Тека pages з набором html-файлів – це не повноцінні сторінки, а просто шматки html-коду, які вбудовуються всередину загального каркасу.

Тому, щоб при зверненні до site.ru/about не посипалися 500, 403 і інші помилки, ми повинні всі вхідні запити на сайт перенаправляти на index.php. index.php у нас - це звичайна html-розмітка без єдиного рядка php-коду (але це тільки поки що).

Спочатку виводимо меню. Далі йде заголовок сторінки. І порожній div з id = content

У # content будуть звантажуватися динамічно вміст сторінок з файлів pages/* .html.

Тільки звернути увагу на атрибути data-menu і data-link = "ajax" біля посилань. Вони введені для того, щоб відрізняти посилання-навігації від звичайних зовнішніх посилань, які на нашому сайті теж будуть. data-link = "ajax"

означає, що при кліку за цим посиланням ми перехопимо стандартну поведінку браузера і візьмемо роботу з посиланням в свої руки. А data-menu означає, який пункт головного меню буде виділено при кліку на це посилання. Тут data-menu дублюється з атрибутом href, але можливі й інші варіанти. Наприклад, коли ми заліземо в розділ frontend блогу, то ми вкажемо data-menu = "blog".

Ми маємо окремий модуль app, який при завантаженні сторінки запускає свою функцію init. У ній навішуємо обробники подій і виконуємо ще деякий код. Також бачимо 2 об'єкти: config и ui. В ui будуть закешовані всі dom-елементи, які знадобляться нам у роботі.

$ menu нам потрібно, щоб виділяти окремі пункти меню, $ pceTitle будемо змінювати динамічно при переході між сторінками, а в $ content буде завантажуватися вміст файлів pages/* .html

siteTitle:'Webdevkin SPA' - заголовок сайту, використовується в декількох місцях.

mainPage:'main' - вказуємо стартову сторінку сайту, ту, яка відкриється при заході на site.ru. Зараз це головна сторінка main, але можна поставити стартову, наприклад, "Про проект" - about.

Найважливіше - це об'єкт pages. Кожне поле об'єкта описує одну сторінку сайту. Зараз нам знадобляться тільки 2 пункти: заголовок сторінки і меню, до якого вона відноситься. Ключі об'єкта, тобто сторінки, збігаються з назвами файлів у pages/* .html і атрибутами href у внутрішніх посиланнях.

І нарешті, напишемо код, заради якого всі і затіяли. Коду, що виконує безпосередньо роботу з обслуговування сайту, набагато менше, ніж підготовка до його написання.

Функція init, має прив'язування потрібних подій _bindHandlers

При явному кліку за посиланням _navigate ми зупиняємо спливання події кліка і скасовуємо дефолтну поведінку браузера (перехід за посиланням). Потім ми визначаємо сторінку, яку ми хочемо завантажити (розуміємо по атрибуту href), і викликаємо нову функцію _loadPage. Вона і зробить всю основну роботу із завантаження контенту, зміни заголовка та інше-інше. І в кінці через

history.pushState додаємо новий запис в історії браузера. Так, ми самі, явним чином створюємо історію браузера. Тоді, коли вважаємо за потрібне. І зберігаємо дані про завантажену сторінку в об’єкт {page: page}. Ці дані нам знадобляться в наступній функції _popState.

У _popState ідея аналогічна: шукаємо потрібну сторінку і запускаємо ту ж _loadPage.

e.state & & e.state.page - витягує нам сторінку з об’єкта історії, яку ми завбачливо записали в _navigate. Якщо ж об’єкт e.state недоступний (наприклад, коли ми перший раз зайшли на сайт site.ru і ще не встигли побродити по ньому), то беремо сторінку, зазначену головною в нашому конфігу - config.mainPage.

Функція history.pushState і той факт, що в window.onpopstate доступний об’єкт e.state з даними, записаними в pushState, - це все, що достатньо знати про History API.

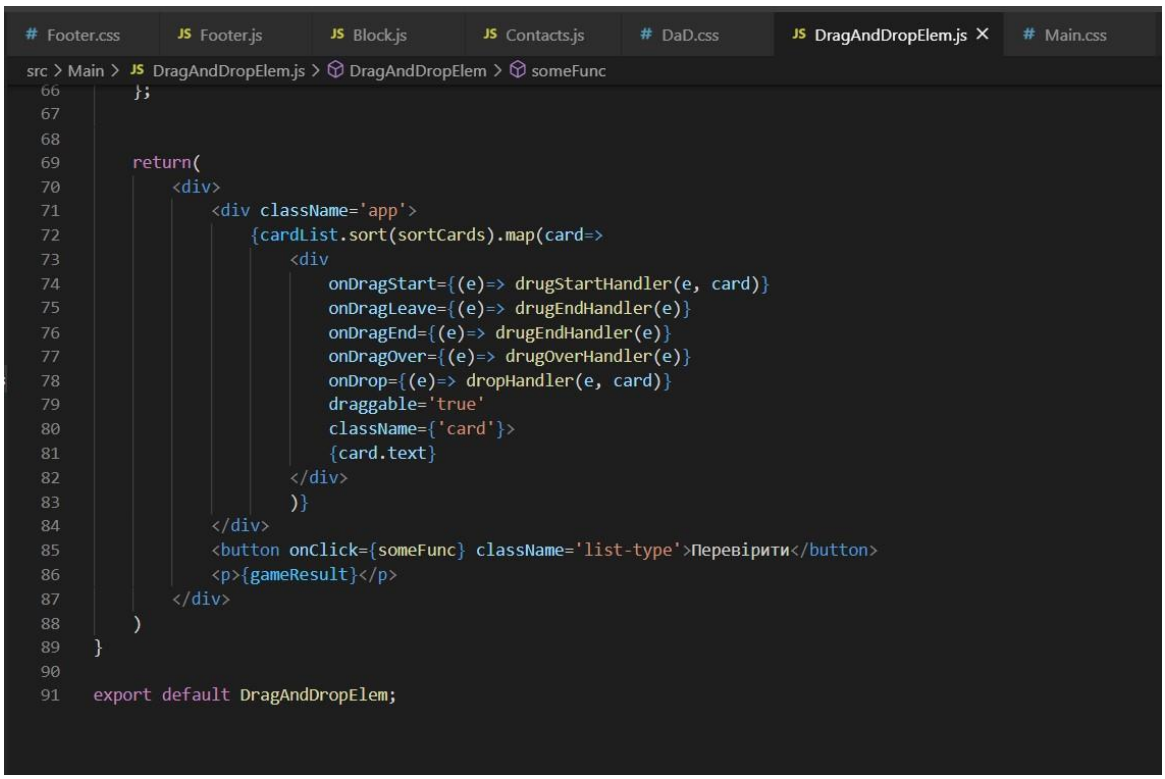
Спочатку формуємо url, тобто шлях, за яким ми завантажимо html для сторінки. Потім витягнемо з конфіга заголовок сторінки і пункт меню, що підлягає виділенню. Далі отримуємо html-вміст сторінки через банальний jQuery.get і виконуємо ще ряд нехитрих дій.

а) Оновлюємо заголовок сторінки

б) У 2 рядки виділяємо потрібний пункт меню

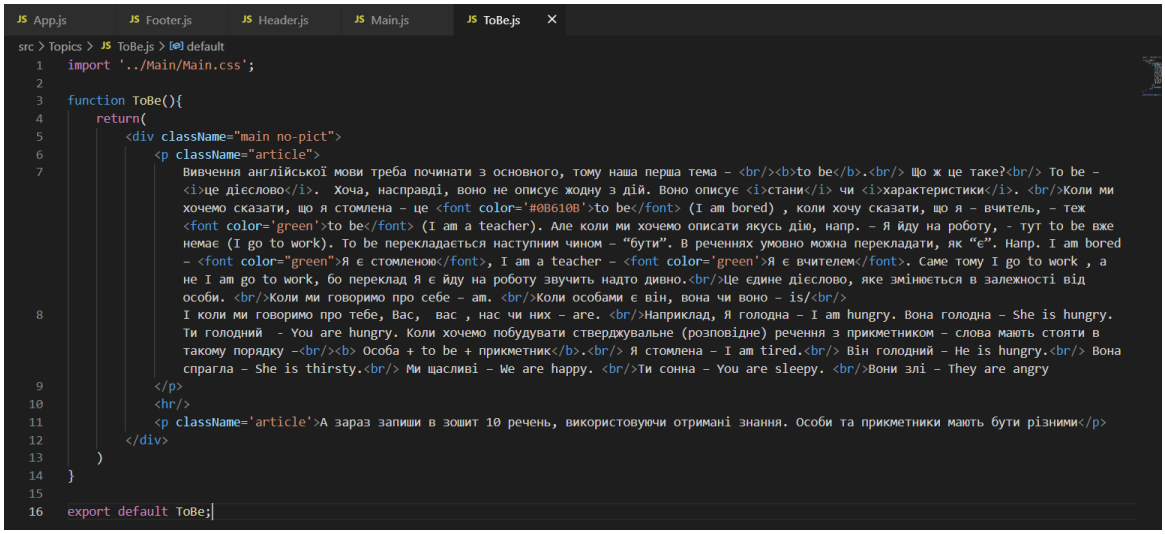
в) Змінюємо заголовок вже на самій сторінці, в html-коді

г) Завантажуємо власне html-вміст сторінки, отриманий з url



```
# Footer.css  JS Footer.js  JS Block.js  JS Contacts.js  # DaD.css  JS DragAndDropElem.js  # Main.css
src > Main > JS DragAndDropElem.js > DragAndDropElem > someFunc
66      };
67
68
69      return(
70          <div>
71              <div className='app'>
72                  {cardList.sort(sortCards).map(card=>
73                      <div
74                          onDragStart={(e)=> drugStartHandler(e, card)}
75                          onDragLeave={(e)=> drugEndHandler(e)}
76                          onDragEnd={(e)=> drugEndHandler(e)}
77                          onDragOver={(e)=> drugOverHandler(e)}
78                          onDrop={(e)=> dropHandler(e, card)}
79                          draggable='true'
80                          className={'card'}>
81                          {card.text}
82                      </div>
83                      )}
84              </div>
85              <button onClick={someFunc} className='list-type'>Перевірити</button>
86              <p>{gameResult}</p>
87          </div>
88      )
89  }
90
91  export default DragAndDropElem;
```

Рисунок 3.10. Компонент Drag and Drop ч4



```
JS App.js  JS Footer.js  JS Header.js  JS Main.js  JS ToBe.js
src > Topics > JS ToBe.js > default
1   import '../Main/Main.css';
2
3   function ToBe(){
4       return(
5           <div className="main no-pict">
6               <p className="article">
7                   Вивчення англійської мови треба починати з основного, тому наша перша тема – <br/><b>to be</b>.<br/> Що ж це таке?<br/> To be –
                    <i>це дієслово</i>.  Хоча, насправді, воно не описує жодну з дій. Воно описує <i>стани</i> чи <i>характеристики</i>. <br/>Коли ми
                    хочемо сказати, що я стомлена – це <font color='#0B610B'>to be</font> (I am bored) , коли хочу сказати, що я – вчитель, – теж
                    <font color='green'>to be</font> (I am a teacher). Але коли ми хочемо описати якусь дію, напр. – Я йду на роботу, - тут to be вже
                    немає (I go to work). To be перекладається наступним чином – “бути”. В реченнях умовно можна перекладати, як “є”. Напр. I am bored
                    – <font color="green">Я є стомленою</font>, I am a teacher – <font color='green'>Я є вчителем</font>. Саме тому I go to work , а
                    не I am go to work, бо переклад Я є йду на роботу звучить надто дивно.<br/>Це єдине дієслово, яке змінюється в залежності від
                    особи. <br/>Коли ми говоримо про себе – am. <br/>Коли особами є він, вона чи воно – is/<br/>
8                   І коли ми говоримо про тебе, Вас,  вас , нас чи них – are. <br/>Наприклад, Я голодна – I am hungry. Вона голодна – She is hungry.
                    Ти голодний  - You are hungry. Коли хочемо побудувати стверджувальне (розповідне) речення з прикметником – слова мають стояти в
                    такому порядку –<br/><b> Особа + to be + прикметник</b>.<br/> Я стомлена – I am tired.<br/> Він голодний – He is hungry.<br/> Вона
                    спрагла – She is thirsty.<br/> Ми щасливі – We are happy. <br/>Ти сонна – You are sleepy. <br/>Вони злі – They are angry
9               </p>
10              <hr/>
11              <p className='article'>А зараз запиши в зошит 10 речень, використовуючи отримані знання. Особи та прикметники мають бути різними</p>
12          </div>
13      )
14  }
15
16  export default ToBe;
```

Рисунок 3.11. Компонент з окремою граматичною темою

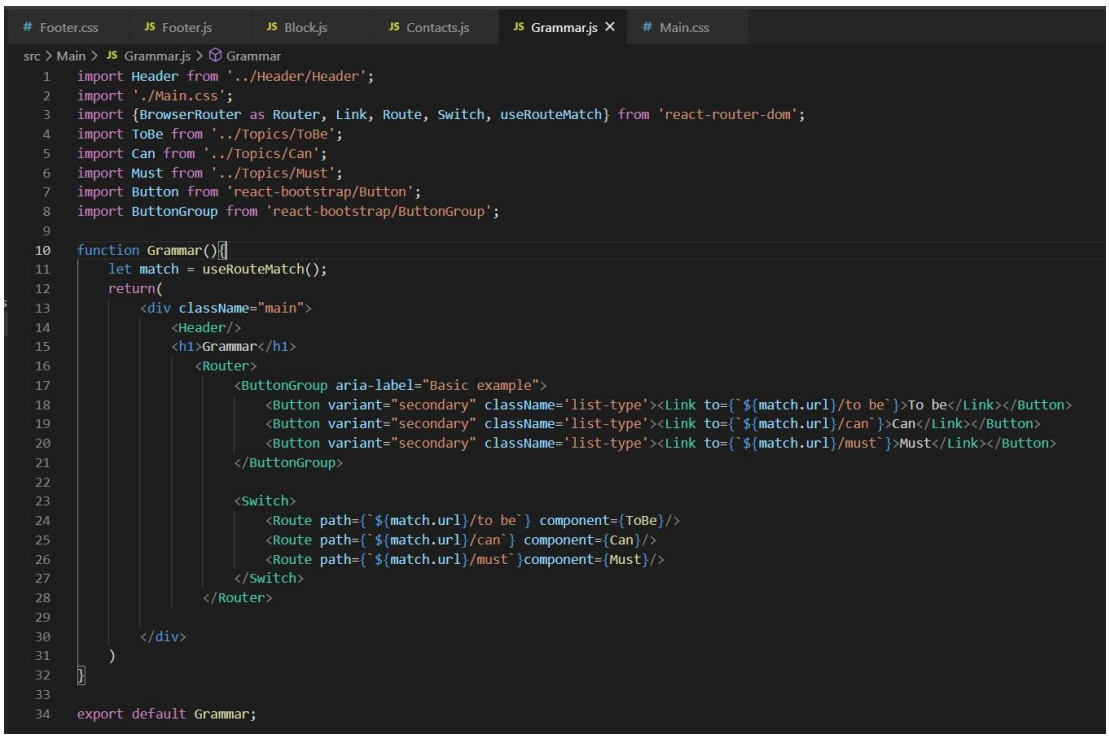


Рисунок 3.12. Компонент з теорією з граматики

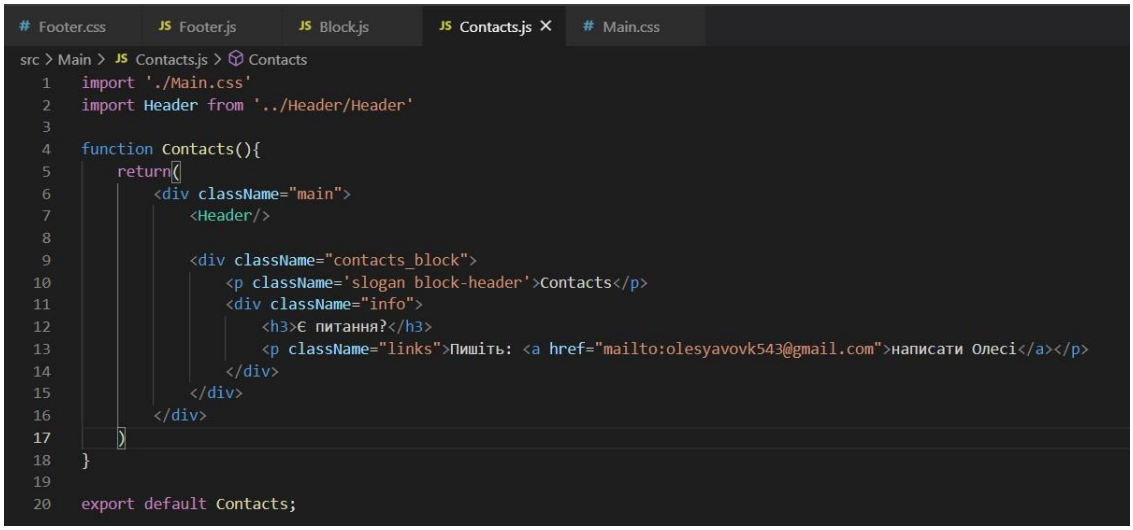


Рисунок 3.13. Компонент з контактами

CSS – код :

src > # App.css > .nav > li

```css
*{
  font-family: 'Comfortaa', cursive;
}

.App {
  text-align: center;
  min-height: 700px;
}

.header{
  height: 50px;
  background: #B404AE;
}

.nav {
  float: right;
  margin: 26px 40px 5px 0;
}

.nav > li {
  font-size: 22px;
  color: #f7fe2e;
  list-style-type: none;
  float: left;
  width: 130px;
  height: 35px;
  margin: 0 20px;
}

.nav > li::after{
  clear: both;
}
```

Рисунок 3.14. App.css

```css
.nav > li::after{
  clear: both;
}

.nav > li > a {
  text-decoration: none;
  color: black;
}

.nav > li > a:active{
  color: #8A0886;
}

.nav > li > a:hover{
  letter-spacing: 2px;
}
```

Рисунок 3.15. App.css

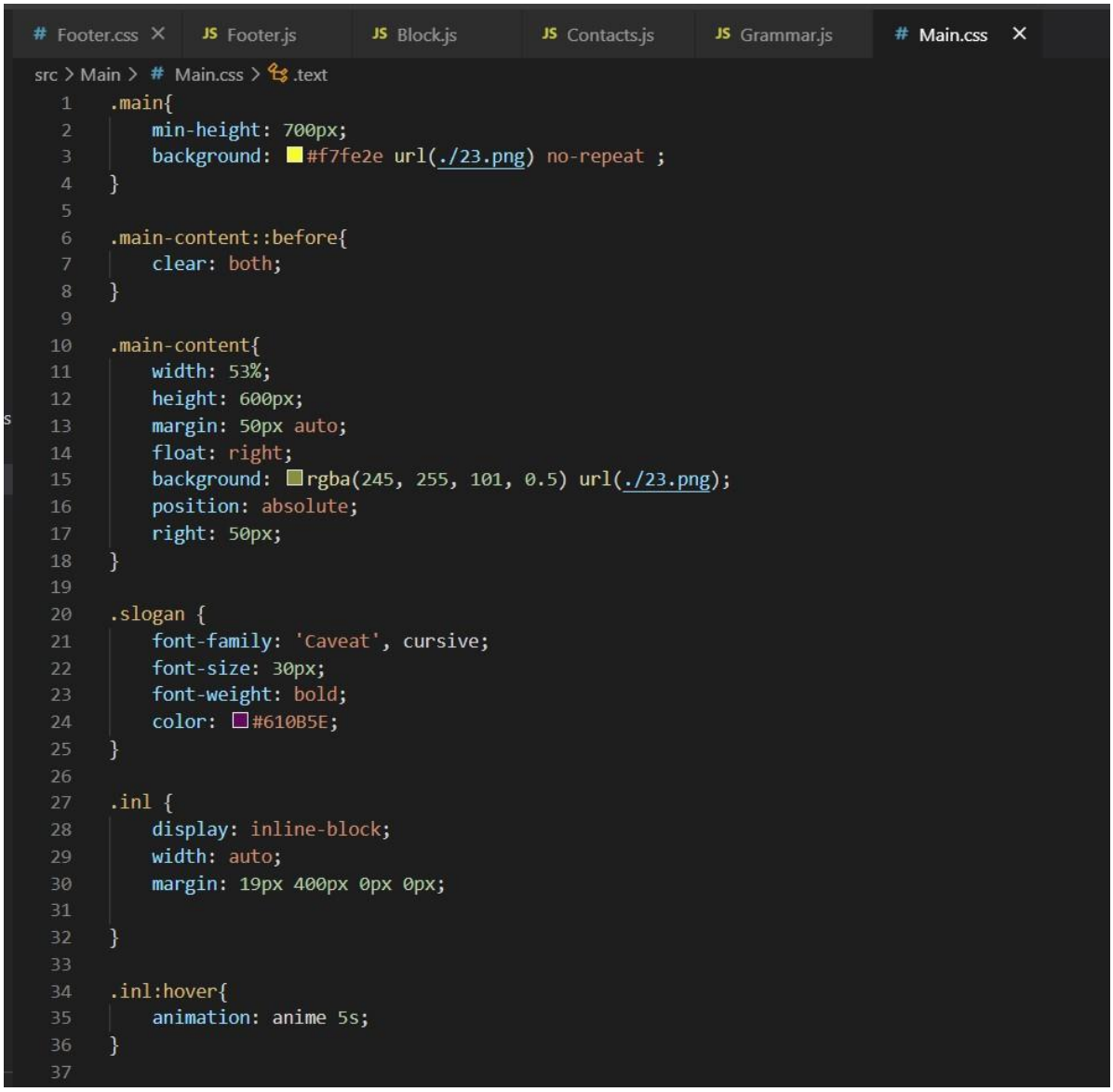


Рисунок 3.16. Main.css

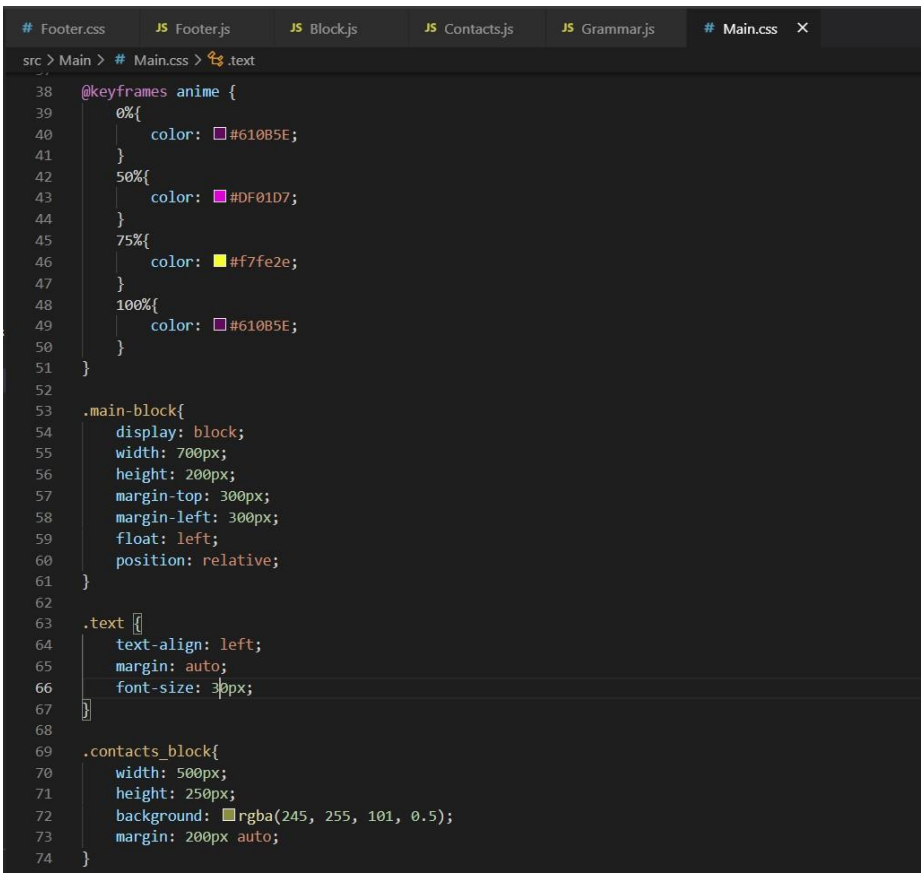


Рисунок 3.17. Main.css

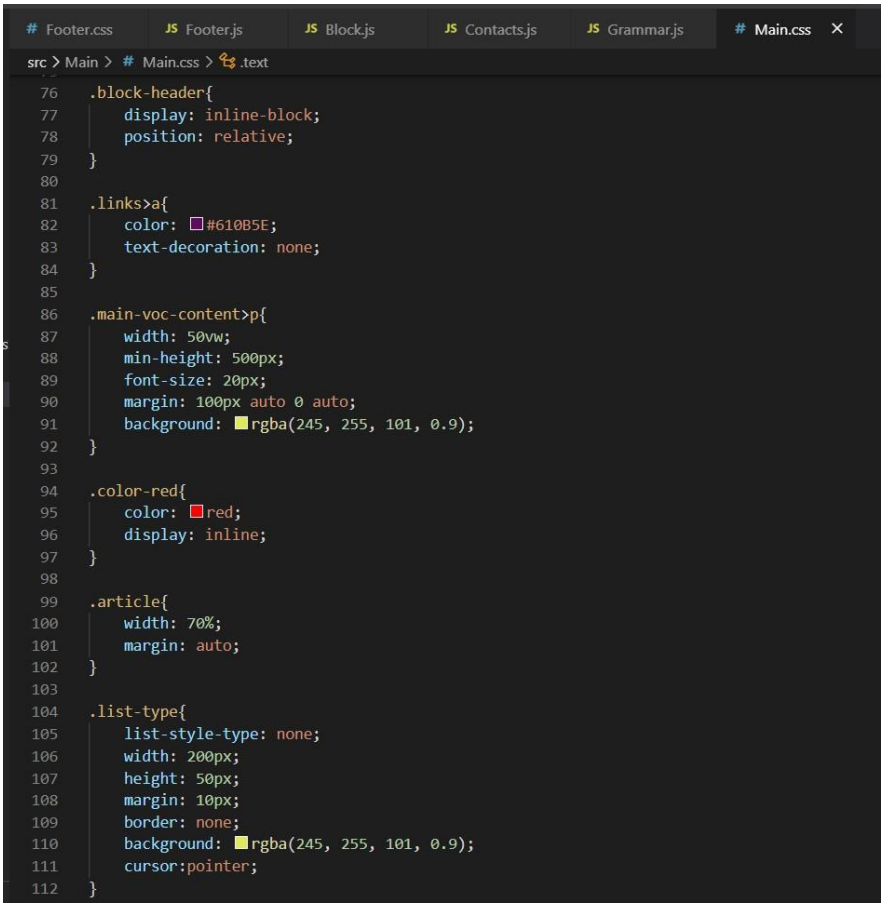


Рисунок 3.18. Main.css

Footer.css | Footer.js | Block.js | Contacts.js | Grammar.js | Main.css

src > Main > # Main.css > .text

```
113
114  .list-type a{
115      text-decoration: none;
116      text-align: center;
117      font-size: 25px;
118  }
119
120  .list-type a:active{
121      color: #8A0886;
122    }
123
124  .no-pict{
125      background: rgba(245, 255, 101, 0.9);
126      background-image: none;
127      width: 70%;
128      padding-top: 30px;
129      margin: 50px auto 0 auto;
130  }
131
132  .task{
133      width: 70%;
134      height: 60px;
135      font-size: 20px;
136      padding-top: 30px;
137      margin: 100px auto 0 auto;
138      background-color: rgba(245, 255, 101, 0.9);
139  }
```

Рисунок 3.19. Main.css

Header.js | Header.css

src > Header > # Header.css > ...

```
1   .header{
2       height: 50px;
3       background: #B404AE;
4   }
5
6   .nav > ul {
7       float: right;
8       margin: 30px 40px 5px 0;
9   }
10
11  .nav > ul > li {
12      font-size: 17px;
13      color: #8A0886;
14      list-style-type: none;
15      float: left;
16      width: 130px;
17      height: 35px;
18      background: #DF01D7;
19      margin: 0 20px;
20  }
21
22  .nav > ul > li > a {
23      text-decoration: none;
24  }
25
```

Рисунок 3.20. Header.css

```css
.footer{
    height: 50px;
    background: #DF01D7;
}

.footer p{
    font-size: 20px;
    padding-top: 12px;
    margin-top: 0;
    font-family: 'Caveat';
}
```

Рисунок 3.21. Footer.css

```css
.app{
    width: 100vw;
    height: 25vh;
    display: flex;
    align-items: center;
    justify-content: center;
}

.app>p{
    display: block;
    flex: none;
    color: white;
}

.card{
    width: 200px;
    height: 100px;
    border: 1px solid;
    display: flex;
    justify-content: center;
    align-items: center;
    margin: 20px;
    cursor: grab;
}
```

Рисунок 3.22. Drag and Drop.css

Таким чином, ми розглянули етапи створення SPA веб-програм. Правильне і детальне технічне завдання гарантує бажаний результат. У ньому ви описуєте, що хочете отримати від майбутньої програми, починаючи від дизайну кнопок і закінчуючи функціональними можливостями.

Залежно від бюджету, яким ви володієте, ви можете найняти як повноцінну команду програмістів і дизайнерів, так і окремо взятих експертів.

Робота над додатком відбувається таким чином. Спочатку створюється макет майбутньої програми, після чого починається робота над MVP (minimum viable product - мінімально життєздатний продукт). MVP необхідний для отримання зворотного зв’язку від перших користувачів, що в подальшому дозволяє розробити повноцінний програмний продукт без ризиків і додаткових витрат.

3.2 Тестування веб-орієнтованого навчального середовища

З метою того, щоб додаток працював без збоїв і помилок, необхідно провести тестування, виявити і усунути всі дефекти. Існує альфа- і бета-тестування. Перший етап проводиться ще під час розробки самою командою розробників, а другий вже є завданням цільової аудиторії.

Після того, як додаток було створено і запущено, робота не припиняється. Програмний продукт – це живий організм, який повинен постійно розвиватися і пристосовуватися до нових умов. Критерії відбору тестів можуть використовуватися як для створення набору тестів, так і для перевірки, наскільки вибрані тести адекватні розв’язуваним завданням (тестування). При цьому, обговорювані критерії допомагають визначити, коли можна або необхідно припинити тестування.

Тестування – це спостереження за виконанням програми, запущеної з метою тестування із заданими параметрами, за заданим сценарієм або з іншими заданими початковими умовами або цілями тестування. Ефективність тесту може бути визначена тільки в контексті заданих умов.

Тестування для ідентифікації дефектів (Testing for defect identification)

Даний випадок тестування має на увазі успішність процедури тестування, якщо дефект знайдений. Це відрізняється від підходу в тестуванні, коли тести запускаються для демонстрації того, що програмне забезпечення задовольняє

пропонованим вимогам і, відповідно, тест вважається успішним, якщо не знайдено дефектів.

Проблема оракула (the oracle problem). “Оракул”, в даному контексті, будь – який агент (людина або програма), що оцінює поведінку програми, формулюючи вердикт-тест пройдено (“pass”) чи ні (“fail”).

Теоретичні та практичні обмеження тестування (Theoretical and practical limitation of testing)

Теорія тестування виступає проти необгрунтованого рівня довіри до серії успішно пройдених тестів. На жаль, більшість встановлених результатів теорії тестування-негативні, означаючи, за словами Дейкстри (Dijkstra), те, що “тестування програми може використовуватися для демонстрації наявності дефектів, але ніколи не покаже їх відсутність”. Основна причина цього в тому, що повне (всеосяжне) тестування недосяжне для реального програмного забезпечення. Через це сучасні методики говорять, що тестування має визначатися на основі аналізу ризиків і може розглядатися в якості стратегії управління ризиками.

Тестованість (Testability) – це поняття може мати на увазі дві різні ідеї. Перша описує ступінь легкості опису критеріїв покриття тестами для заданої програмної системи. Друга визначає можливість ймовірність, можливість статистичного вимірювання того, що при тестуванні проявиться збій програмної системи. Обидві інтерпретації цього поняття однаково важливі для тестування.

Тестування зазвичай проводиться протягом всієї розробки і супроводу на різних рівнях. Рівень тестування визначає” над чим “ проводяться тести: над окремим модулем, групою модулів або системою, в цілому. При цьому жоден з рівнів тестування не може вважатися пріоритетним. Важливі всі рівні тестування, незалежно від використовуваних моделей і методологій.

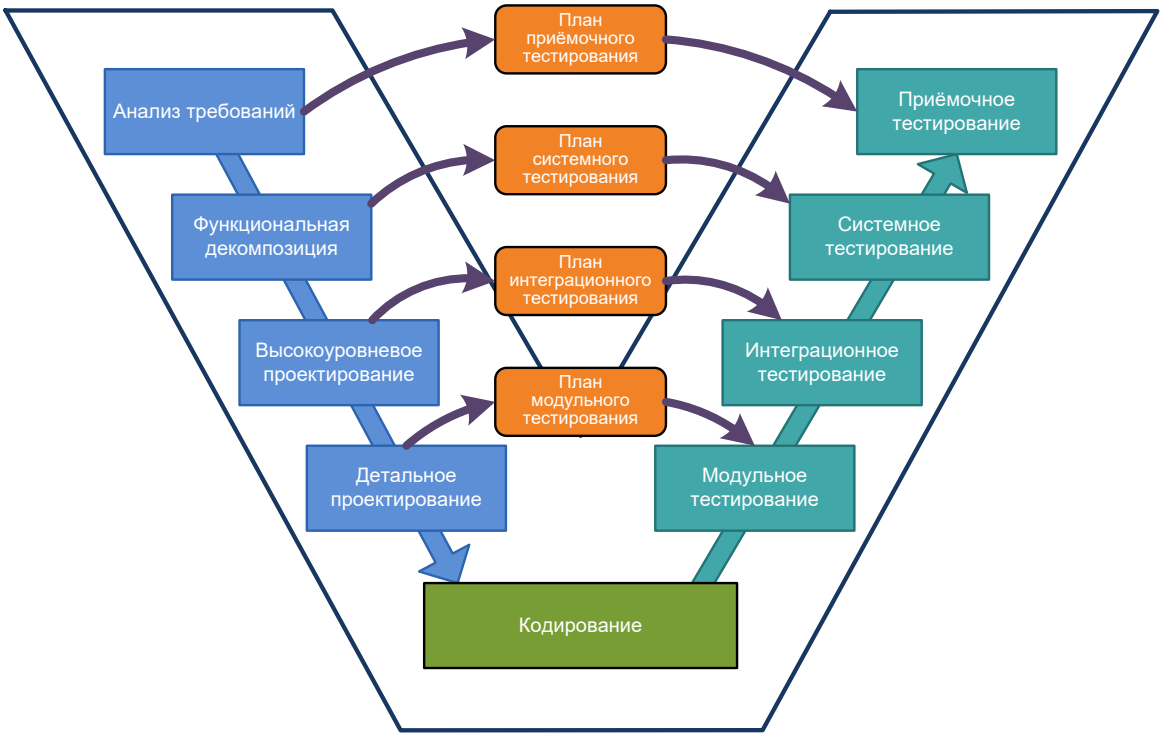


Рисунок 3.9 – V-модель розробки та Тестування ПЗ

Модульне тестування (Unit testing) – цей рівень тестування дозволяє перевірити функціонування окремо взятого елемента системи. Що вважати елементом-модулем системи визначається контекстом. Найбільш повно даний вид тестів описаний в стандарті IEEE Std 1008-1987 (R2003)”Standard for Software Unit Testing”, що задає інтегровану концепцію систематичного і документованого підходу до модульного тестування.

Інтеграційне тестування (Integration testing) Даний рівень тестування є процесом перевірки взаємодії між програмними компонентами / модулями.

Класичні стратегії інтеграційного тестування – “зверху-вниз” і “знизу–вгору” – використовуються для традиційних, ієрархічно структурованих систем і їх складно застосовувати, наприклад, до тестування слабозв’язаних систем, побудованих в сервісно-орієнтованих архітектурах (SOA).

Сучасні стратегії більшою мірою залежать від архітектури тестованої системи і будуються на основі ідентифікації функціональних “потоків” (наприклад, потоків операцій і даних).

Інтеграційне тестування – постійно проведена діяльність, що передбачає роботу на досить високому рівні абстракції. Найбільш успішна практика інтеграційного тестування базується на інкрементальному підході, що дозволяє уникнути проблем проведення разових тестів, пов’язаних з тестуванням результатів чергового тривалого етапу робіт, коли кількість виявлених дефектів призводить до серйозної переробки коду (традиційно, негативний досвід випуску і тестування тільки великих релізів називають “big bang testing.”).

Системне тестування охоплює цілком всю систему. Більшість функціональних збоїв має бути ідентифіковано ще на рівні модульних та інтеграційних тестів. У свою чергу, системне тестування, зазвичай фокусується на нефункціональних вимогах – безпеки, продуктивності, точності, надійності тощо.

На цьому рівні також тестуються інтерфейси до зовнішніх додатків, апаратного забезпечення, операційному середовищі і т. д.

Тестування проводиться відповідно до певних цілей (можуть бути задані явно або неявно) і різним рівнем точності. Визначення мети точним чином, що виражається кількісно, дозволяє забезпечити контроль результатів тестування.

Тестові сценарії можуть розроблятися як для перевірки функціональних вимог (відомі як функціональні тести), так і для оцінки нефункціональних вимог. При цьому, існують такі тести, коли кількісні параметри і результати тестів можуть лише опосередковано говорити про задоволення цілям тестування (наприклад, “usability” – легкість, простота використання, в більшості випадків, не може бути явно описана кількісними характеристиками).

Можна виділити наступні, найбільш поширені і обґрунтовані цілі (а, відповідно, види) тестування:

Приймальне тестування(Acceptance / qualification testing)

Перевіряє поведінку системи на предмет задоволення вимог замовника. Це можливо в тому випадку, якщо замовник бере на себе відповідальність, пов’язану з проведенням таких робіт, як сторона “приймаюча” програмну

систему, або специфіковані типові завдання, успішна перевірка (тестування) яких дозволяє говорити про задоволення вимог замовника.

Такі тести можу проводитися як із залученням розробників системи, так і без них.

Установче тестування (Installation testing) – тести проводяться з метою перевірки процедури інсталяції системи в цільовому оточенні.

Перед тим, як випускається програмне забезпечення (призначене для масового використання), як мінімум, воно повинно проходити стадії альфа (внутрішнє пробне використання) і бета (пробне використання із залученням відібраних зовнішніх користувачів) версій. Звіти про помилки, що надходять від користувачів цих версій продукту, обробляються відповідно до певних процедур, що включають підтверджуючі тести (будь-якого рівня), що проводяться фахівцями групи розробки.

Допомагаючи ідентифікувати причини збоїв, тестування має на увазі і підвищення надійності програмних систем. Випадково генеруються сценарії тестування можуть застосовуватися для статистичної оцінки надійності. Обидві цілі-підвищення і оцінка надійності – можуть досягатися при використанні моделей підвищення надійності.

Регресійне тестування – це повторне вибіркове тестування системи або компонент для перевірки зроблених модифікацій не повинно призводити до непередбачених ефектів.

На практиці це означає, що якщо система успішно проходила тести до внесення модифікацій, вона повинна їх проходити і після внесення таких. Основна проблема регресійного тестування полягає в пошуку компромісу між наявними ресурсами і необхідністю проведення таких тестів у міру внесення кожної зміни. Певною мірою, завдання полягає в тому, щоб визначити критерії “масштабів” змін, з досягненням яких необхідно проводити регресійні тести.

Для визначення успішності тестів їх результати повинні оцінюватися, аналізуватися. У більшості випадків,” успішність “ тестування має на увазі, що тестоване програмне забезпечення функціонує так, як очікувалося і в процесі

роботи не призводить до непередбачених наслідків. Не всі такі наслідки обов'язково є збоями, вони можуть сприйматися як "перешкоди". Перед усуненням виявленого збою, необхідно визначити і зафіксувати ті зусилля, які необхідні для аналізу проблеми, налагодження та усунення. Це дозволить надалі забезпечити велику глибину вимірювань, а, відповідно, в перспективі, мати можливість поліпшення самого процесу тестування.

Звіти про проблеми / журнал тестування (Problem reporting / test log)

У багатьох випадках, в процесі тестової діяльності ведѐтся журнал тестування, що фіксує інформацію про відповідних роботах: коли проводиться тест, який тест, ким проводиться, для якої конфігурації програмної системи (в термінах параметрів і в термінах ідентифікованої версії контексту конфігураційного управління) і т. п. Несподівані або некоректні результати тестів можуть записуватися в спеціальну підсистему ведення звітності по збоїв (problem-reporting system, забезпечуючи формування бази даних, використовуваної для налагодження, усунення проблем і подальшого тестування. Крім того, аномалії (перешкоди), які не можна ідентифікувати як збої, також можуть фіксуватися в журналі та/або системі ведення звітності по збоях. Звіти по тестах можуть бути входом для процесу управління змінами і генерації запитів на зміни (change request) в рамках процесів конфігураційного управління (див.далі відповідну область знань «конфігураційне управління»).

Відстеження дефектів (Defect tracking). Збої, виявлені в процесі тестування, найчастіше породжуються дефектами і помилками, присутніми в тестованої програмній системі (також вони можуть бути наслідком поведінки операційного та/або тестового оточення). Такі дефекти можуть (і, найчастіше, повинні) аналізуватися для визначення моменту і місця першої появи даного дефекту в системі, які типи помилок стали причиною цих дефектів (наприклад, погано сформульовані вимоги, некоректний дизайн, витоку пам'яті тощо) і коли вони могли б бути виявлені вперше. Вся ця інформація використовується для визначення того, як може бути поліпшений сам процес тестування і наскільки критична необхідність таких поліпшень.

ВИСНОВКИ

Об’єктом даного проєктування є методи та засоби створення онлайн порталу з вивчення англійської мови.

Предметом проєктування став онлайн портал з вивчення англійської мови.

Область застосування – педагогіка, психологія, додаткова дошкільна та шкільна освіта, родина, розвивальні гуртки, дитячі заклади.

Метою даної роботи стало створення онлайн порталу з вивчення англійської мови, в якому буде максимально доступна інформація, викладена доступною мовою. Портал призначений для тих, для кого вивчення англійської мови – це страшніше, ніж похід до стоматолога. Тим, хто вже багато разів починав, але не зміг досягнути бажаного результату, тому, що наштовхувався на нерозуміння якихось понять. На цьому онлайн-порталі буде можливість звернутися до викладача особисто і отримати відповіді на питання.

SPA був написаний, аби зекономити час для викладача англійської. Інформація в додатку викладена так само, як би викладач розповів учням в класі, тож учень може ознайомитися з матеріалом, поставити питання, якщо вони є та закріпити інформацію вправами.

Маршрутизація в додатку була реалізована за допомогою модулю react-router- dom та об’єктів Router, Switch, Route, Link

Отже, основні завдання, поставлені метою дипломної роботи виконано, а саме:

1. Розроблено навчальний SPA додаток, який буде відображати відповідний дизайн в залежності від віку користувача (дитячий, дорослий) і залежний від рівня його підготовки алгоритм вивчення предмету.

2. Реалізовано фільтри підбору сценарію навчання на початку роботи з ресурсом.

3. Зроблено інтерфейс максимально зрозумілим для користувачів з різними рівнями комп’ютерної грамотності.

4. Досліджено та проаналізовано існуючі аналоги.

5. Протестовано роботу системи SPA додатку в цілому.

6. Зроблено висновки до виконаної роботи.

Для вирішення поставлених завдань використовувалися такі методи: аналіз літератури, згідно з досліджуваною темою; аналіз інформаційних систем, Інтернет-ресурсів, необхідних для створення та публікації веб-ресурсу. Отже, поставлені завдання виконані, мету дослідження досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Авраменко В. С. Проєктування інформаційних систем / В. С. Авраменко, С. В. Голуб, В. І. Салапатов. Електронне видання. – Черкаси : ЧНУ ім. Богдана Хмельницького, 2015. – 496 с.

2. Батарова, Н. И. Технологии Web 2. 0 в формировании опыта применения информационно-коммуникационных технологий у студентов гуманитарных специальностей [Электронный ресурс] / Н.И. Батарова, М.А. Лукоянова, А.Х. Хусаинова // Современные проблемы науки и образования. – 2014. – №5. – Режим доступа: http : //goo. gl/JTKERf.

3. Безызвестных, Е.А. Электронное обучение в подготовке бакалавров педагогических направлений: опыт и перспективы /Е.А. Безызвестных, О.А. Иманова, О.Г. Смолянинова // Информатика и образование. – 2015. – № 2.

4. Брылёва, В.А. Технология организации и методического сопровождения электронной веб-ориентированной среды для учебного процесса гуманитарного профиля [Электронный ресурс] / В.А. Брылёва, О.П. Сафонова // Современные исследования социальных проблем. – Режим доступа: http://j oumal-s.org/index.php/sisp/article/view/22013 8/pdf 118.s.org/index.php/sisp/article/view/22013 8/pdf 118.

5. Власова, Е.З. Электронное обучение в современном вузе: проблемы, перспективы и опыт использования [Электронный ресурс] / Е.З. Власова // Universum: Вестник Герценовского университета. – 2014. – № 1. – Режим доступа: http://cyberleninka.ru/article/n/elektronnoe-obuchenie-v- sovremennom-vuze-problemy-perspektivy-i-opyt-ispolzovaniya.

6. Власова, Е.З. Электронное обучение в современном вузе: проблемы, перспективы и опыт использования [Электронный ресурс] / Е.З. Власова // Universum: Вестник Герценовского университета. – 2014. – № 1. – Режим доступа: http://cyberleninka.ru/article/n/elektronnoe-obuchenie-v- sovremennom-vuze-problemy-perspektivy-i-opyt-ispolzovaniya.

7. Войтович, И.К. Специфика создания электронных образовательных курсов [Электронный ресурс] / И.К. Войтович // Вестник ТГПУ. - 2015. -№ 1. – Режим доступа: http://cyberleninka.ru/article/n/spetsifika-sozdaniya-elektronnyh-obrazovatelnyh-kursov.

8. Войтович, И.К. Специфика создания электронных образовательных курсов [Электронный ресурс] / И.К. Войтович // Вестник ТГПУ. - 2015. -№ 1. – Режим доступа: http://cyberleninka.ru/article/n/spetsifika-sozdaniya-elektronnyh-obrazovatelnyh-kursov.

9. Голышева, М.Д. E-learning и дистанционное образование в России и за рубежом: проблемы и пути решения [Электронный ресурс] / М.Д. Голышева и [др.] // Филологические науки. Вопросы теории и практики 2011. – № 4. – Режим доступа: http ://cyberl eninka.ru/arti cl e/n/e-l earning-i-distantsionnoe-obrazovanie-v-rossii-i-za-rubezhom-problemv-i-puti-resheniva.

10. Иванова, Н.Ю. Системное и прикладное программное обеспечение: учебное пособие / Н.Ю. Иванова, В.Г. Маняхина. – М.: МИГУ, 2011. – 202 с.

11. Кандаурова, Н.В. Вычислительные системы, сети и телекоммуникации: учебное пособие / Н.В. Кандаурова и [др.]. – 2 изд. – М.: ФЛИНТА, 2013. – 344 с.

12. Компьютерная графика. Studopedia.org [Электронный ресурс]. – Режим доступа: http : //studopedia. info/2-3 516.html

13. Корень, А.В. Особенности разработки учебных курсов с использованием электронной образовательной среды Moodle [Электронный ресурс] / А.В. Корень // Интернет-журнал Науковедение. – 2013. – №1 (14). – Режим доступа: http : //cvberleninka.ru/article/n/osobenno sti-razrabotki-uchebnyh-kursov-s-ispolzovaniem-elektronnov-obrazovatelnov-sredv-moodle.

14. Корень, А.В. Особенности разработки учебных курсов с использованием электронной образовательной среды Moodle [Электронный ресурс] / А.В. Корень // Интернет-журнал Науковедение. – 2013. – №1 (14). – Режим доступа: http : //cvberleninka.ru/article/n/osobenno sti-razrabotki-uchebnyh-kursov-s-ispolzovaniem-elektronnov-obrazovatelnov-sredv-moodle.

15. Мукажанов, Е.Б. Электронное обучение – неотъемлемая часть современного образования [Электронный ресурс] / Е.Б. Мукажанов, Е.Е. Телебаев // Актуальные проблемы гуманитарных и естественных наук. – 2012. – № 5. – Режим доступа: http://cyberleninka.ru/article/n/elektronnoe-obuchenie-neotemlemaya-chast-sovremennogo-obrazovaniya.

16. Пресс, И.А. О некоторых психолого-педагогических аспектах применения e-Learning [Электронный ресурс] / И.А. Пресс // Высшее образование в России. - 2011. - № 10. - Режим доступа: http://cyberleninka.m/article/n/o-nekotoryh-psihologo-pedagogicheskih-aspektah-primeneniya-e-learning.

17. Ребрина, Ф.Г. Этапы разработки электронного учебного курса на платформе LMS Moodle [Электронный ресурс] / Ф.Г. Ребрина, И.А. Леонтьева // Вестник ЧГПУ. – 2014. – № 2. – Режим доступа: http://cyberleninka.ru/article/n/etapy-razrabotki-elektronnogo-uchebnogo-kursa-na-platforme-lms-moodle.

18. Роберт, И.В. Современные информационные технологии в образовании: дидактические проблемы, перспективы использования: монография / И.В. Роберт. – М.: ИИО РАО, 2010. – 140 с.

19. Стариченко, А.Е. Применение современных технических средств обучения в e-leaming [Электронный ресурс] / А.Е. Стариченко, Л.В. Сардак // Педагогическое образование в России. – 2014. – № 2. – Режим доступа: http://cvberleninka.ru/article/n/primenenie-sovremennyh-tehnicheskih-sredstv-obucheniva-v-e-learning.

20. Факеева, М.И. Использование сервисов Web 2.0 в работе учителя предметника [Электронный ресурс] / М.И. Факеева // Социальные сервисы WEB 2.0 в образовании: опыт, проблемы, перспективы: материалы II интернет-конференции – Режим доступа: http://internet-konfweb202011.blogspot.ru/2012/02/web-20 22.html.

21. Шишлина, Н.В. Автор электронного курса: учебно-методическое пособие / Н.В. Шишлина. – Ижевск: ИжГТУ имени М.Т. Калашникова, 2015

ДОДАТОК А

Рис 1 Головна сторінка

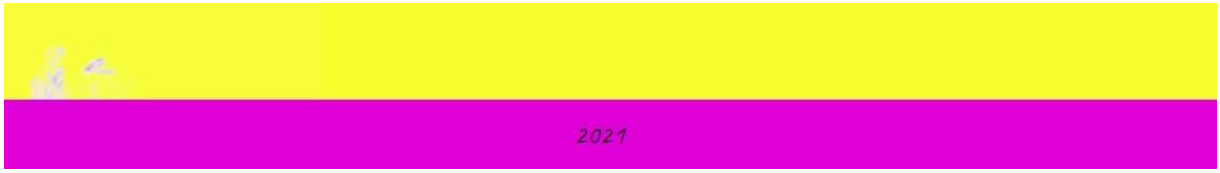


Рисунок 2. Футер

Рисунок 3. Сторінка зворотного зв’язку

ДОДАТОК Б

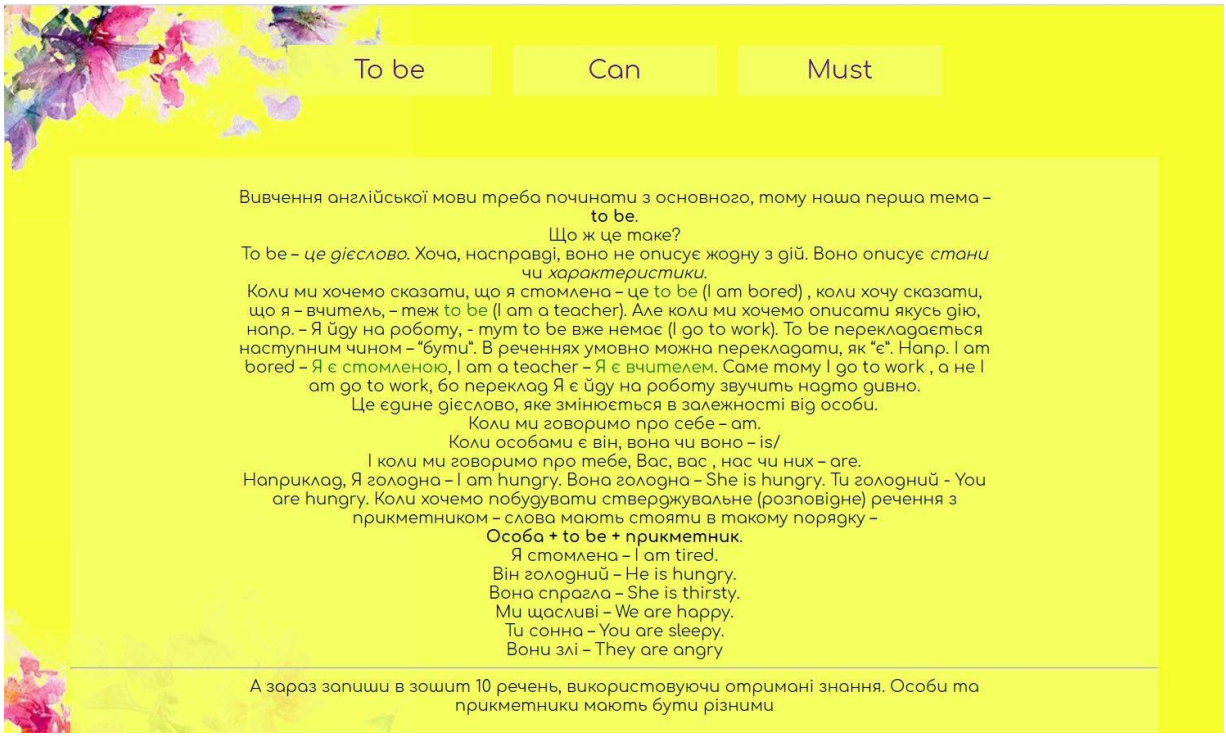


Рисунок 1. Навчальна складова – вступне заняття

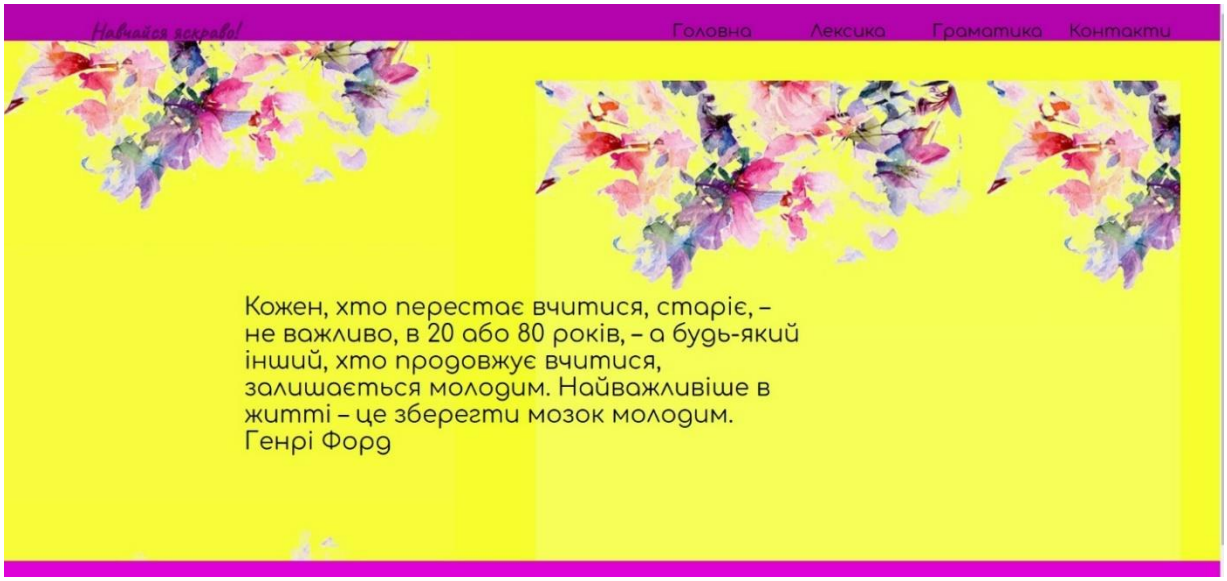


Рисунок 2. Місія проєкту – цитата Генрі Форду

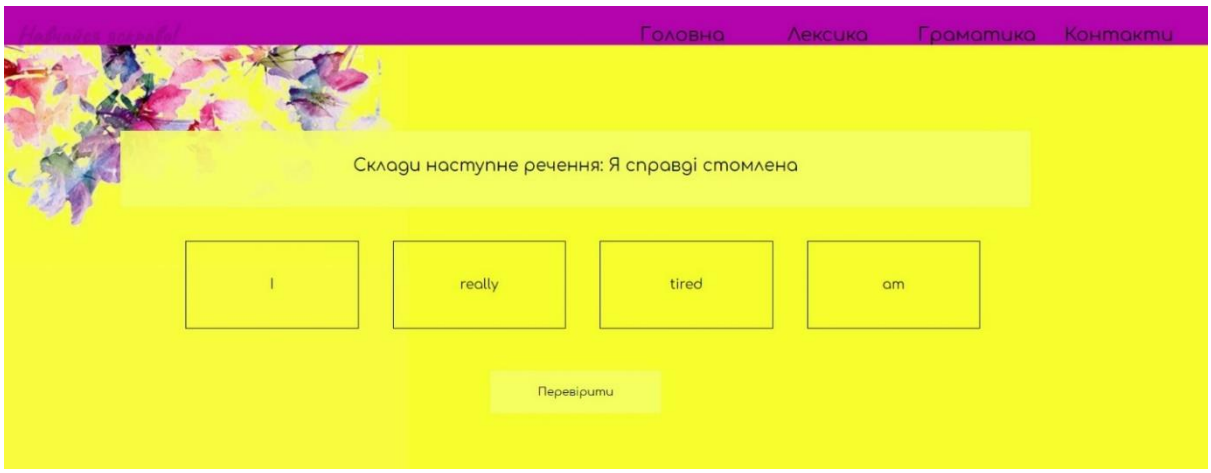


Рисунок 3. Елемент заняття

Рисунок 4. Елемент заняття

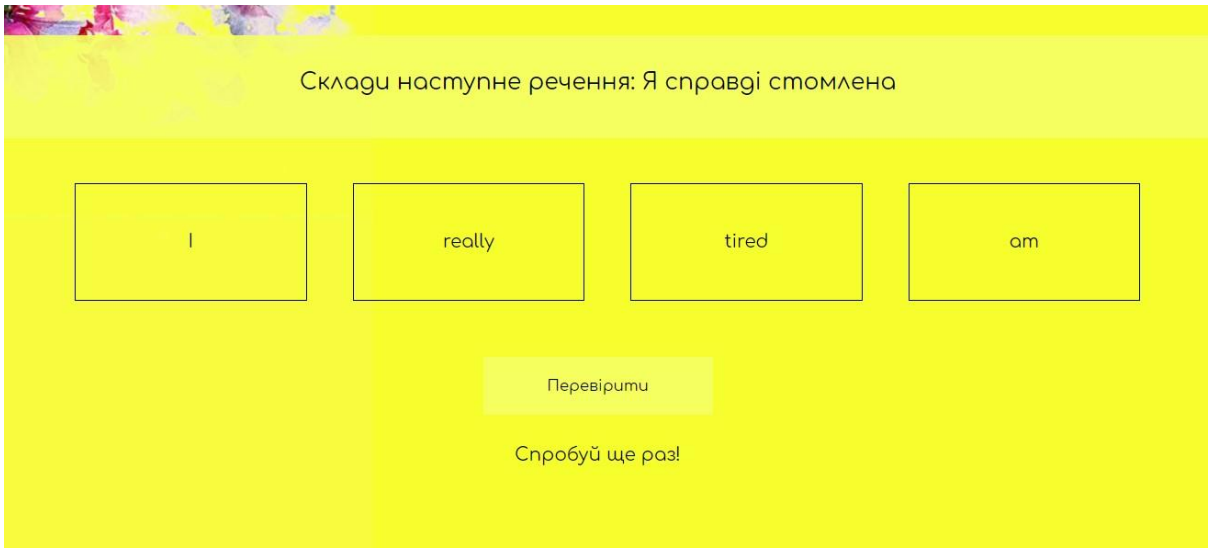


Рисунок 5. Елемент заняття

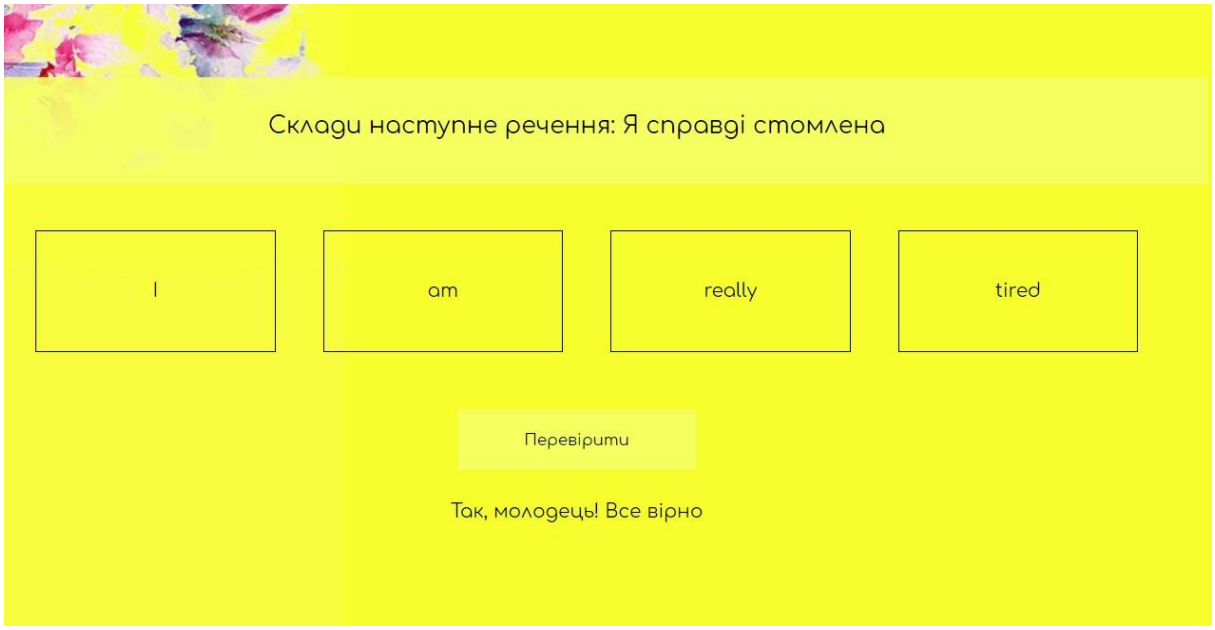


Рисунок 6. Елемент заняття