ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для інтерактивного вивчення будови автомобілів та їх компонентів мовою C#»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Богдан АНІСІМОВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Богдан АНІСІМОВ

Керівник: Світлана ШЕВЧЕНКО
к.п.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Анісімову Богдану Андрійовичу

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для інтерактивного вивчення будови автомобілів та їх компонентів мовою C#»
керівник кваліфікаційної роботи к.п.н., доцент Світлана ШЕВЧЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про процес вивчення будови автомобілів та їх компонентів; опис методів інтерактивного навчання; мова програмування C#, мови HTML, CSS, JavaScript, фреймворк ASP.NET, база даних MS SQL.

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

1. Огляд та аналіз існуючих методів та засобів для вивчення будови автомобілів.
2. Проєктування веб-застосунку для вивчення будови автомобілів та їх компонентів.

3. Програмна реалізація та опис функціонування веб-застосунку для вивчення будови автомобілів та їх компонентів.
4. Тестування веб-застосунку.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма класів.
6. Схема бази даних.
7. Мапа сайту.
8. Екранні форми.
9. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд та аналіз існуючих методів та засобів для вивчення будови автомобілів.	14.03-20.03.2024	
4	Проектування веб-застосунку для вивчення будови автомобілів та їх компонентів.	21.03-05.04.2024	
5	Програмна реалізація та опис функціонування веб-застосунку для вивчення будови автомобілів та їх компонентів.	06.04-20.04.2024	
6	Тестування веб-застосунку.	21.04-28.04.2023	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Богдан АНІСІМОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана ШЕВЧЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 48 стор., 2 табл., 31 рис., 13 джерел.

Мета роботи – спрощення процесу вивчення будови автомобілів за допомогою веб-застосунку розробленого мовою C#.

Об'єкт дослідження – процес інтерактивного вивчення будови автомобілів.

Предмет дослідження – веб-застосунок для інтерактивного вивчення будови автомобілів та їх компонентів.

Короткий зміст роботи: Дане дослідження присвячено проблемі вивчення будови автомобіля за допомогою веб-застосунку. У роботі розглянуто процес вивчення будови автомобілів та методи і засоби інтерактивного навчання у цілому. На основі аналізу розроблених інструментальних засобів для вивчення будови автомобілів: HowACarWorks, Green-Way, Avto.Perevirka було спроєктовано веб-застосунок, та програмно реалізовані ключові функціональні можливості, зокрема: перегляд навчальних статей, проходження вправ, реєстрація акаунтів, коментування, пошук статей, взаємодія з інтерактивними схемами. В роботі використано мову програмування C#, мови HTML, CSS, JavaScript, фреймворк ASP.NET, базу даних MS SQL. Проведено тестування додатку.

Сферою використання застосунку є навчальний процес студентів транспортного університету, а також застосунок можуть використовувати зацікавлені особи, які бажають вивчити будови і роботи автомобілів та їх компонентів.

КЛЮЧОВІ СЛОВА: БУДОВА АВТОМОБІЛІВ, АВТОМЕХАНІКА, РОБОТА АВТОМОБІЛІВ, ВЕБ-ЗАСТОСУНОК, ІНТЕРАКТИВНЕ НАВЧАННЯ, ТРЕНІНГ, КУРС, C#, ASP.NET.

ЗМІСТ

ВСТУП

У наш час автомобілі відіграють незамінну роль у житті людей та функціонуванні суспільства в цілому. Їх важливість простежується на всіх рівнях: від особистого комфорту до економічного розвитку націй. Знання основ обслуговування легкового автомобіля дозволить значно збільшити тривалість життя техніки і запобігти неприємним ситуаціям [1]. Потреба у відповідних знаннях робить попит на засоби навчання будівлі і роботі автомобілів та їх компонентів.

Задачею є розробка веб-застосунку для інтерактивного вивчення будови автомобілів, що дозволить новачкам та людям з досвідом отримати та закріпити знання з обраної теми.

Об’єкт дослідження - процес інтерактивного вивчення будови автомобілів.

Предмет дослідження - веб-застосунок для інтерактивного вивчення будови автомобілів.

Мета роботи – спрощення процесу вивчення будови автомобілів за допомогою веб-застосунку розробленого мовою C#.

Для досягнення цієї мети в роботі необхідно вирішити такі **завдання**:

1. Проаналізувати процес вивчення будови та роботи автомобілів і їх компонентів.
2. Розглянути існуючі програмні засоби для вивчення будови та роботи автомобілів та їх компонентів, знайти їх переваги та недоліки.
3. Розробити функціональні та нефункціональні вимоги.
4. Дослідити інструменти і технології для розробки веб-застосунку та спроєктувати застосунок.
5. Розробити веб-застосунок на основі обраних технологій і з урахуванням зазначених вимог.
6. Провести тестування веб-застосунку.

Методи дослідження. Для вирішення вищезгаданих завдань у роботі використано наступні методи: системно-структурні методи, порівняльний аналіз. Для розробки програмного забезпечення використані такі технології, інструменти та мови програмування, як C#, HTML, CSS, JavaScript, фреймворк ASP.NET, база даних MS SQL.

Наукова новизна одержаних результатів. Наукова новизна полягає у тому, що у розробленому додатку більшість процесів навчання носить інтерактивний характер, що дозволяє спростити вивчення будови автомобілів та їх компонентів.

Практична значущість результатів полягає у створенні програмного продукту у вигляді веб-застосунку, який дозволить запезпечити ефективність навчального процесу при вивченні будови автомобілів та їх компонентів. Створений програмний продукт може бути корисним у процесі вивчення даного модуля студентами транспортних закладів вищої та спеціальної освіти, а також особам, які зацікавлені у підвищенні своїх знань з будови автомобіля.

1 АНАЛІЗ СУЧАСНОГО СТАНУ У ПРАКТИЦІ ЗАСТОСУВАННЯ ІНТЕРАКТИВНОГО НАВЧАННЯ

1.1 Інтерактивне навчання: поняття, інструменти та технології

У сучасному світі, де технології постійно розвиваються, інтерактивні методи навчання стають все більш актуальними та ефективними. Однією з сфер, де ці методи можуть знайти широке застосування, є вивчення будови автомобілів. Актуальність інтерактивного вивчення будови автомобілів зумовлена не лише технологічним прогресом в автомобільній індустрії, але й необхідністю поглибленого розуміння принципів роботи та конструкції автомобілів у зв'язку з постійним підвищенням їхньої складності та технологічності.

Технологія інтерактивного навчання є однією з передових інноваційних навчальних стратегій. За дослідженнями, поняття "інтерактивна технологія" не має чіткого визначення у науковій літературі. Термін "інтерактивний" походить від англійських слів "inter", що означає взаємний, та "act", що означає діяти, тобто інтерактивна технологія спрямована на створення взаємодії та отримання результатів в процесі навчання. Також під взаємодією розуміється контакт між учнями та викладачами, що є не менш важливим процесом під час навчання, бо це дозволяє обмінюватися інформацією, досвідом, виявляти помилки, отримувати рекомендації і т.д. [2].

Розглянувши сутність інтерактивного навчання можна зрозуміти, що для підтримки найбільш ефективного отримання знань даний підхід є найбільш привабливим, але треба обрати конкретний метод. Розглянемо види інтерактивного навчання.

Зважаючи на широкий спектр інтерактивних методів навчання, їх можна класифікувати за кількома ключовими критеріями:

1. За формою взаємодії:

- пряма взаємодія: це методи, де взаємодія відбувається між учнем і вчителем, або між учнями, в реальному часі. Це може бути діалог, обговорення, спільне розв'язання завдань тощо.

- непряма взаємодія: це методи, де учні взаємодіють з навчальним матеріалом або завданнями без прямої участі вчителя або інших учнів. Наприклад, самостійне читання, вирішення завдань, використання інтерактивних вправ тощо.

2. За ступенем активності учасників:

- активні методи: це методи, у якому учні активно взаємодіють з навчальним матеріалом або між собою. Це може бути групова робота, рольові ігри, дебати, практичні вправи тощо.

- пасивні методи: методи, у яких учні в основному приймають інформацію без активної участі або взаємодії. Це може бути лекція, перегляд відео, читання матеріалів тощо.

3. За використанням технологій:

- традиційні інтерактивні методи: це методи, які не вимагають використання технологій, такі як обговорення, групова робота тощо.

- інтерактивні методи з використанням технологій: це методи, де для взаємодії використовуються різні технологічні засоби, такі як веб-платформи, мультимедійні презентації, онлайн-ігри тощо [3].

Ці класифікації допомагають систематизувати різні підходи до інтерактивного навчання та обрати найбільш придатні методи для конкретних навчальних цілей та контексту.

Через те, що web-застосунок робиться для широкої аудиторії та кількість користувачів може бути великою, за формою взаємодії було обрано метод непрямої взаємодії. Для цього застосунок має надавати доступ кожному користувачу до курсів з навчальними матеріалами та закріпленням матеріалу у вигляді практичних занять, що дає можливість кожній людині опанувати матеріал автономно.

Щодо взаємодії між вчителями та учнями, то додаток повинен мати систему коментарів та чату, можливість обговорювати кожну частину навчального матеріалу, що дозволяє взаємодіяти учасникам навчального процесу між собою. Завдяки цьому виконується один із принципів інтерактивного навчання і підвищується ефективність отримання знань.

За використанням технологій було обрано напрям з використанням технологій. Для цього було вирішено зробити проєкт у вигляді веб-застосунку.

Веб-застосунки для навчання мають наступні переваги:

1. Доступність: дозволяють отримати доступ до навчального матеріалу з будь-якого місця та у будь-який час.
2. Гнучкість: надають можливість навчатися власним темпом і вибирати підходящий режим навчання.
3. Різноманітність ресурсів: забезпечують доступ до широкого спектра навчальних матеріалів, включаючи відеоуроки, інтерактивні вправи, тести тощо.
4. Інтерактивність: сприяють активній взаємодії з матеріалом через використання різноманітних інтерактивних елементів.
5. Можливості співпраці: дозволяють спілкуватися та співпрацювати з іншими учасниками навчання, навіть якщо вони знаходяться далеко від вас.
6. Оновлення та адаптація: легко оновлюються та адаптуються до змін у навчальних програмах та потребах користувачів.
7. Відстеження прогресу: надають можливість відстежувати прогрес навчання, виконання завдань та результати тестів.
8. Ефективність вивчення: завдяки інтерактивності та доступності навчального матеріалу сприяють покращенню ефективності навчання.

Зазначені переваги роблять формат веб-застосунку як один із найкращих варіантів для втілення будь якого проєкту, пов’язаного з навчальним процесом, і вивчення будови автомобілів не є виключеням.

1.2 Аналіз існуючих web-додатків для інтерактивного вивчення будови автомобілів

Цільовою аудиторією веб-додатку є люди, які хочуть набути знань з будови автомобілів. У першу чергу, веб-додаток орієнтується на новачків, які хочуть вивчати будову автомобілів з нуля, але додатком можуть користуватись і люди, які вже мають певні знання. На даний момент вже існують рішення, якими користуються люди для вирішення проблеми. Розглянемо найбільш поширені.

1.2.1 How A Car Works

“How A Car Works” (howacarworks.com) - це онлайн-ресурс, присвячений навчанню про автомобілі та їхню будову. Цей сайт пропонує широкий спектр інформації про те, як працюють різні системи автомобілів, їхні основні компоненти та принципи роботи.

На сайті можна знайти ілюстровані посібники, які пояснюють роботу різних частин автомобіля, таких як двигун, трансмісія, підвіска, гальмівна система та інші. Кожна стаття або розділ зазвичай супроводжується діаграмами, фотографіями та пояснювальними текстами, які розкривають принцип роботи кожної системи.

"How A Car Works" також надає інформацію про технічне обслуговування та ремонт автомобілів, що може бути корисним для власників автомобілів, а також для тих, хто цікавиться автомобільною технікою.

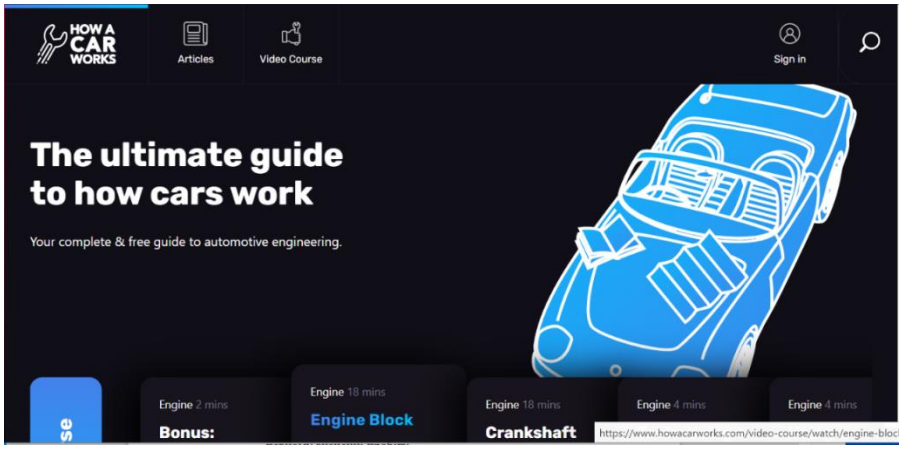


Рис. 1.1 Інтерфейс веб-сайту HowACarWorks

Сайт має два напрямки - курс зі статтями та відеокурс. Частина курсів є безкоштовною, але для отримання доступу к усім матеріалам потрібно внести оплату. Під час оплати, проходить реєстрація акаунта для користувача по електроній пошті, після чого він може проходити увесь курс увійшовши до акаунту.

Головні недоліки даного ресурсу:

1. Відсутність перекладів курсу. Усі матеріали написані лише англійською мовою, що обмежує використання людьми, які не знають англійської, зокрема, українськими користувачами. Мовний бар'єр погано впливає на процес навчання.

2. Доступ до багатьох навчальних матеріалів є платним. Обмеження у вигляді придбання доступу до курсу зменшує потенційну аудиторію, особливо це стосується неплатоспроможного населення.

3. На веб-сайті відсутня будь яка система коментування та іншої взаємодії користувачів з матеріалом та іншими учнями. Це залишає учня один на один з матеріалом без можливості обговорення курсу та пов'язаних з ним питань.

4. Відсутність практичної частини. Користувач ресурсу може лише ознайомитись зі статтями та відеоресурсами без подальшого закріплення нових знань. Без можливості застосування теоретичних знань на практиці, учні можуть мати обмежене розуміння матеріалу. Практичні вправи та завдання допомагають закріпити теоретичні концепції та забезпечують краще засвоєння знань. Важливо, щоб навчальні курси мали практичну складову, яка допомагає учням застосовувати отримані знання, розвивати навички та підготовлюватися до реальних ситуацій.

Висновок. Веб-сайт **How A Car Works** є непоганим ресурсом для вивчення будови автомобільної техніки, але має такі значні недоліки як відсутність перекладу, системи взаємодії користувачів, практичних вправ та платний доступ до матеріалів.

1.2.2 Green-Way

“Green-Way” (green-way.com.ua) - український веб-сайт для початківців в автомобільній тематиці. Сайт зосереджений на навчанні. У першу чергу, головним напрямком курсу Green-Way є навчання майбутніх водіїв правилам дорожнього руху. Також у курсі виділено розділ під навчення будови та роботи авто (рис. 1.2).

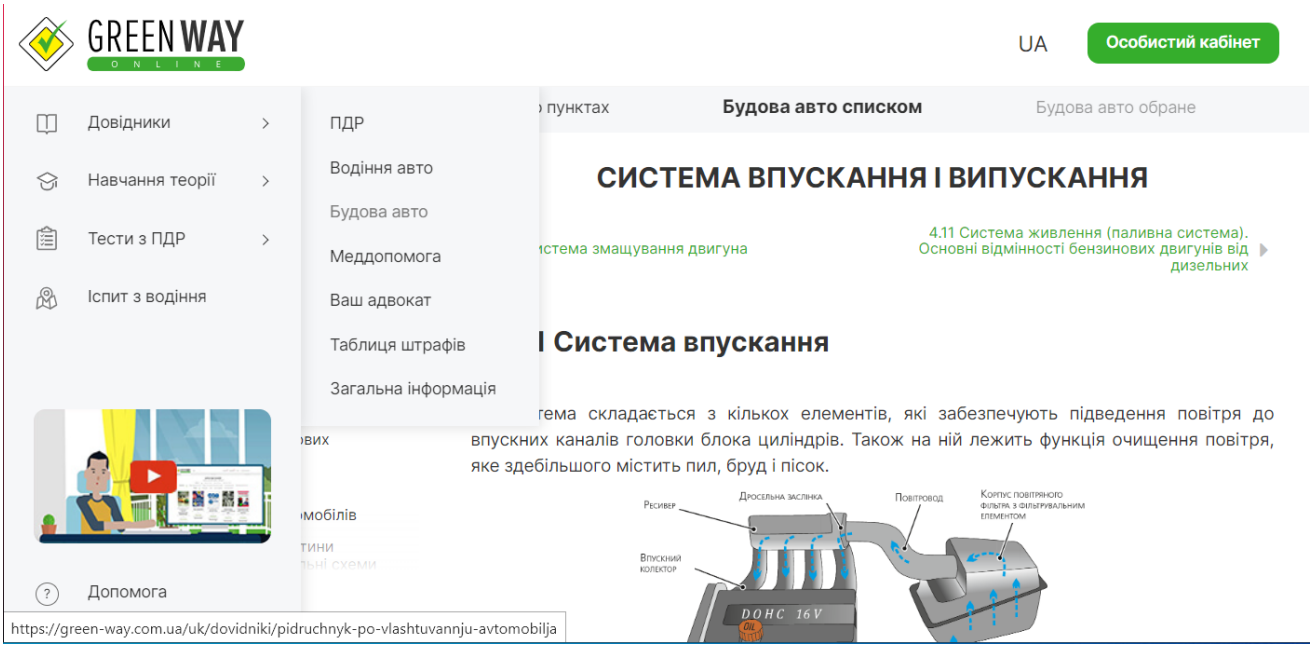


Рис. 1.2 Інтерфейс веб-сайту Green-Way

Сайт пропонує довідники з різних напрямів, які повинен знати майбутній водій, серед них є «Будова авто». Довідник має список розділів зі статтями, які мають текстову та графічну інформацію.

Користувач може створити акаунт на сайті та отримати доступ до курсів. Більшість курсів є платними, серед них можна знайти «Автомобільну будову». Курси, на відміну від довідників, мають систему прогресу та практичні частини у кожному уроці. Практика містить тестування з теми урока, виконання якого обмежено в часі. (рис. 1.3)

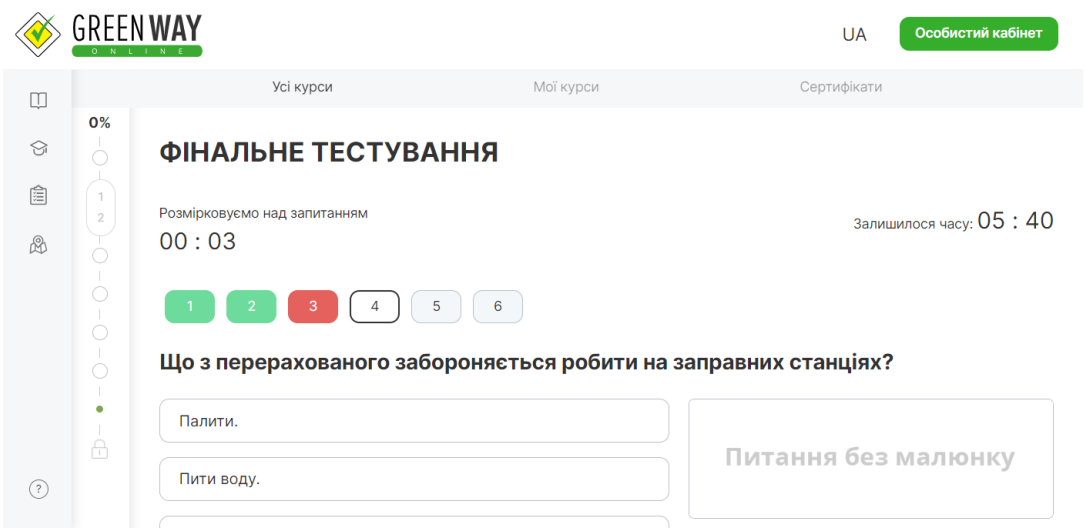


Рис. 1.3 Приклад тестового завдання

Авторизовані користувачі можуть відслідковувати свій прогрес на курсі, коментувати навчальні матеріали та спілкуватись з авторами курсів та іншими учнями, що розширює можливості у навчанні.

Сайт Green-Way має наступні недоліки.

1. Інтерфейс. Через занадто велику кількість функцій (у тому числі не пов’язаних з тематикою будівлі автомобілів) інтерфейс є перевантаженим, що може заплутати користувача під час пошуку потрібної теми.

2. Доступ до курсів з будівлі авто є платним. Хоча сайт і має довідник з безкоштовними матеріалами, для доступу до повноцінного курсу потрібно платити. Також курси Green-Way мають обмежений час доступу, що може заважати людям, які не мають постійної можливості займатись навчанням.

3. Основною тематикою сайту є правила дорожнього руху. Більшість матеріалів та вся практична частина (окрім деяких курсів) сфокусована саме на ПДР, що є незручним для людини, яка хоче лише вивчати роботу та будову авто. Також обмеження цього напряму у вигляді окремого курсу заважає подальшому розвитку. Наприклад, практична частина курсів сайту Green-Way підтримує лише тестування, а додавання нових засобів для практики є складним.

Висновок. Веб-сайт **Green-Way** є багатофункціональним рішенням для вирішення проблеми, але зазначені вище недоліки дають простір для поліпшення розроблюваного проєкту.

1.2.3 Авто Перевірка

Авто Перевірка (avto.proverka.com.ua) - це портал, який містить статті з автомобільної тематики.

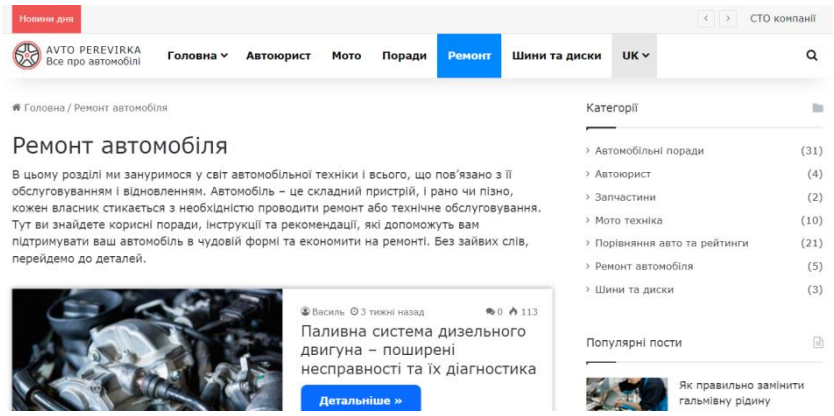


Рис. 1.4 Приклад навчальної статті

На “Авто Перевірка” викладаються публікації, які несуть в собі корисні знання з автомобільної тематики: будова авто, ремонт, поради та інші суміжні теми. Кожна стаття входить в окрему категорію, завдяки чому людина може шукати потрібну їй статтю у потрібній галузі (рис. 1.4).

Весь контент на сайті є безкоштовним та не потребує реєстрації та авторизації.

Головні проблеми розглянутого сайту:

1. Відсутність структуризації матеріалів. Усі матеріали викладаються у вигляді окремих статей, які не об’єднані між собою. Це вимагає користувача самостійно шукати та переходити до певних статей та визначати порядок вивчення. Це найбільш на новачків, які не знають з чого починати та з якими статтями варто ознайомлюватись.

2. Сайт не має практичних завдань. Відсутність вправ не дає закріпити отримані знання, що погіршує процес вивчення.

3. Відсутність системи акаунтів, коментарів. Через це усі публікації залишаються без обговорень, а учні не можуть взаємодіяти один з одним. Також неможливо зберігати прогрес по вивченим статтям, що може заплутати новачка.

Висновок. Портал **“Авто Перевірка”** хоча і може надавати корисну для навчання інформацію, він має занадто базову функціональність, яка не може забезпечити повноцінний процес навчання, особливо інтерактивного.

Розглянуті аналоги було зведено в порівняльну таблицю 1.1.

Таблиця 1.1

Зведені результати аналізу характеристик програмних засобів для вивчення будови автомобілів

Показник	HowACarWorks.com	Green-Way.com.ua	Avto.Proverka.com.ua
Зручний інтерфейс	+	-	+
Пошук статей	+	+	+
Система облікових записів	Тільки для доступу к курсу	Повноцінна система	Відсутня
Взаємодія користувачів один з одним	-	+	-

Продовження таблиці 1.1

Зведені результати аналізу характеристик програмних засобів для вивчення будови автомобілів

Показник	HowACarWorks.com	Green-Way.com.ua	Avto.Proverka.com.ua
Система прогресу	-	+	-
Закріплення матеріалу	-	+	-
Доступ до курсу	Платний	Платний	Безкоштовно
Мова	Англійська	Українська	Українська

Аналіз альтернативних засобів виявив наступні недоліки: складність інтерфейсу, відсутність системи облікових записів, проблеми зі взіємодією користувачів, відсутність системи прогресу і закріплення пройденого матеріала та проблеми з перекладом на українську мову. Врахування цих недоліків дозволяє сформулювати первинні вимоги та подалі розробити продукт який буде мати переваги перед аналогами.

1.3 Тренінг як одна із технологій інтерактивного навчання у процесі вивчення будови автомобіля

Як було розглянуто у підрозділі 1.1, інтерактивне навчання може мати різні класифікації та форми втілення. Одним із варіантів реалізації інтерактивного навчання є така система як тренінг.

Тренінг (англ. training від train – навчати, виховувати, готувати, тренувати) - це спеціально організована форма навчання, яка спрямована на підвищення знань, навичок та компетенцій у певній області [4]. У випадку навчання будові автомобілів, тренінг можна втілити такими способами:

1.Відеоуроки та відеоінструкції. Надання користувачам доступ до відеоуроків, які показують процеси демонтажу, зборки, діагностики та ремонту різних частин автомобіля. Відеоінструкції можуть бути використані для пояснення складних процесів та демонстрації крок за кроком.

2.Інтерактивні вправи та тести. Створення інтерактивних вправ та тестів, які допоможуть користувачам закріпити отримані знання. Це може включати в себе вибір правильної запчастини на основі її зображення, визначення назви частини за описом, розпізнавання проблем на основі звуків або зображень тощо.

3.Інтерактивні анімації та діаграми. Використання анімації та інтерактивних діаграм, щоб показати, як працюють різні системи автомобіля. Це може допомогти користувачам краще зрозуміти складні механізми та процеси.

4.Спільнота. Створення відповідних умов, завдяки яким користувачі зможуть обговорювати проблеми, ділитися досвідом та задавати питання. Це створить можливість для взаємодії та обміну знаннями між учасниками навчання.

5.Статті з порадами та інструкціями: Надання користувачам доступу розділу зі статтями, де вони зможуть знайти матеріали з корисними порадами, інструкціями та новинами з автомобільної тематики.

Усі названі елементи можна реалізувати у вигляді навчального курсу, який допоможе учням зрозуміти та освоїти матеріал про будову автомобілів. Навчальний курс може мати структуровану програму, що включає в себе різні розділи та теми, які будуть вивчатися поетапно.

Наприклад, перший розділ може бути присвячений загальним поняттям та принципам будови автомобіля, другий - розглядати різні системи автомобіля (двигун, трансмісія, підвіска тощо), третій - методам діагностики та ремонту. Кожен розділ може включати в себе відеоуроки, інтерактивні вправи, тести, демонстраційні анімації, коментарі для обговорення, набір корисних статтей тощо.

Організація курсу у вигляді онлайн-курсу дозволить учням вивчати матеріал у зручний для них час та темп, а також мати можливість повертатися до

матеріалів для повторення або глибшого вивчення. Такий підхід дозволить зробити навчання більш доступним та ефективним для широкого кола людей.

Висновок. У першому розділі дипломної роботи було проаналізовано актуальність веб-додатка для інтерактивного вивчення будови автомобілів та їх компонентів, розглянуто основні теоретичні поняття, виявлено аналоги та їх переваги і недоліки та обрано напрям розвитку продукту. Завдяки отриманим даним можна сформулювати вимоги до проєкту і почати проєктування та розробку.

2 ВИБІР ТЕХНОЛОГІЙ І СЕРЕДОВИЩА РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРАКТИВНОГО ВИВЧЕННЯ БУДОВИ АВТОМОБІЛІВ

2.1 Основні підходи до розробки тренінгу

Розробка тренінгу - це процес, що вимагає глибокого розуміння потреб аудиторії, поставлених перед тренінгом завдань та мети навчання. Перед плануванням та розробкою тренінга потрібно виконати наступне.

- Аналіз цільової аудиторії. Перед початком розробки тренінгу важливо ретельно проаналізувати цільову аудиторію. Цільовою аудиторією є люди, які бажають вивчати будову та роботу автомобільної техніки.
- Формулювання цілей. Основний крок у розробці тренінгу - це формулювання конкретних цілей, які потрібно досягти. Ці цілі повинні бути конкретними, вимірюваними, досяжними, релевантними та часово обмеженими. Ціллю тренінгу є надання людині достатніх знань для розуміння роботи автомобілів.
- Розробка змісту. На основі аналізу потреб та визначених цілей розробляється зміст тренінгу. Це може включати план занять, матеріали для навчання, вправи та завдання.
- Вибір методів навчання. Вибір методів навчання залежить від мети тренінгу, аудиторії та доступних ресурсів. В цьому випадку методом навчання є набір лекцій, вправ, демонстрації, дискусій, які оформлені у вигляді онлайн-курсу.
- Розробка матеріалів. На основі змісту та обраного методу навчання розробляються відповідні навчальні матеріали, такі як презентації, відео, інтерактивні завдання тощо.
- Тестування та оцінка. Після розробки тренінгу важливо провести тестування матеріалів та оцінку їх ефективності. Це може включати попередні тестування з аудиторією, виправлення помилок та вдосконалення матеріалів.

- Оцінка результатів. Здійснення оцінки знань та навичок учасників після завершення тренінгу для визначення ефективності навчання та виявлення можливих областей для поліпшення.

2.2 Розробка плану проведення тренінгу із вивчення будови автомобілів та їх компонентів

У попередніх розділах було обрано курс як варіант втілення тренінгу у розроблюваному веб-застосунку. Кожен курс повинен мати певну програму, яку учень може проходити у лінійному порядку.

Весь курс складається з розділів, в кожному з яких є статті з навчальними матеріалами та практична частина з вправами. Проходження «Розділи та уроки» у них йдуть по порядку. Після проходження вправ одного розділу учень переходить до наступного.

Наприклад, першим розділом може бути такий розділ як “Вступ”, який має наступні уроки: Історія, Види автомобілів, Види палива і т.д.

Як може виглядати план курсу:

Розділ 1. Вступ

Урок 1.1 - Історія автомобілів

Урок 1.2. - Класифікація авто

Розділ 2. Двигуни

Урок 2.1 - Принцип роботи двигунів

Урок 2.2 - Види ДВЗ

Урок 2.3 - Види палива

Розділ 3. Будівля 4-тактного двигуна

Урок 3.1 - Блок циліндрів

Урок 3.2 - Поршень

Розділ 4. Підвіска

Урок 4.1 - Амортизатори

Урок 4.2 - Діференціал

Розділ 5. Трансмісія

Урок 5.1 - КПП

Урок 5.2 - Передаточні числа

Кожний урок складається з текстової частини та/або інших навчальних матеріалів такі як малюнки, відео, зовнішні посилання, аудіо. Якщо матеріал статті базується на інших джерела, то наприкінці статті вони повинні бути зазначені для запобігання порушення авторських прав. Також до кожного підпункту в статті може бути створена область коментарів для того, щоб інші учні могли коментувати її.

Після основного блоку статті з інформацією йде практична частина, яка містить вправи по тематиці уроку. Результати пройденого тесту зберігаються у особовому кабінеті.

Окрім основного курсу, сайт може містити додаткові інструменти для допомоги у вивчені будови автомобілів та їх компонентів. Наприклад, таким інструментом може бути інтерактивна схема автомобіля, візуально зображені деталі якого користувач може переглянути та перейти до статей пов’язаних з нею, що може спростити користувачу пошук необхідної інформації, якщо він не знає назви деталі, але знає як вона виглядає. Також можлива реалізація функції для виявлення певних проблем в роботі автомобіля, якщо людина зіштовхнулась з поламкою реального авто, що також буде корисно для новачків і зможе навчити їх ремонту.

2.3 Огляд засобів реалізації веб-застосунку та технологій

Продукт має вигляд веб-застосунку. Веб-застосунки - це програми, які запускаються на веб-сервері та доступні користувачам через веб-браузер [5]. Вони складаються з різних компонентів, які взаємодіють між собою для надання певного функціоналу.

Основними компонентами веб-додатків є:

1. Клієнтська частина (Фронтенд) - це частина веб-додатка, яка відповідає за те, як інформація відображається та взаємодіє з користувачем у браузері. Це та частина, з якою користувач безпосередньо взаємодіє.
2. Серверна частина (Бекенд) - це частина веб-додатка, яка знаходиться на сервері і відповідає за обробку запитів користувачів, взаємодію з базою даних та генерацію відповідей, які надсилаються на фронтенд (клієнтську сторону) для відображення користувачам. Бекенд не є видимим для кінцевого користувача, але він забезпечує всі необхідні функції, які дозволяють веб-додатку працювати.
3. База даних (Зовнішня) - використовується для збереження та організації даних, які використовуються в веб-додатку. Хоча і може бути частиною бекенду, але більшості випадків бази даних робляться у вигляді окремих серверів.

Розглянувши вимоги та обраний формат проєкту було обрано набір інструментів та технологій для розробки додатку.

2.3.1 Інструменти

Для розробки усіх необхідних частин додатку використовуються наступні інструменти:

Visual Studio - це інструмент для розробки програмного забезпечення, створений компанією Microsoft. Дане середовище розробки дозволяє створювати різноманітні програми: десктопні, мобільні, веб-додатки, ігри тощо. Розроблені за допомогою Visual Studio програми можуть бути призначені для платформ, які підтримують Windows, .NET, Xbox та інші.

Основною перевагою Visual Studio є створена екосистема з технологій Microsoft, яка спрощує підключення та управління ними під час розробки.

Visual Studio має Community версію, яка є безкоштовною та доступною для завантаження з офіційного сайту Microsoft. Для розробки дипломної роботи використовується саме ця версія 2022 року. Для розробки серверної частини веб-додатку було завантажено відповідний пакет.

Visual Studio Code (VS Code) - це легкий і потужний редактор коду, розроблений компанією Microsoft. Він призначений для редагування початкових текстів та роботи з файлами проєктів у різних мовах програмування. Однією з ключових особливостей VS Code є його відкрите джерело, що дозволяє розширювати та налаштовувати його за допомогою різноманітних розширень.

Visual Studio Code є популярним інструментом серед розробників завдяки своїй легкості використання, потужному функціоналу та широкій підтримці розширень.

У ході дипломної роботи, VS Code використовується для написання веб-сторінок, які є фронтендом.

Git - це розподілена система керування версіями (VCS), яка використовується для ведення версій коду проєкту та спільної роботи над ним за допомогою командної розробки. Вона дозволяє розробникам зберігати історію змін коду, вносити зміни в код відокремлено від головної версії проєкту та об'єднувати їх з іншими версіями. Git є однією з найпопулярніших систем керування версіями та використовується мільйонами розробників по всьому світу.

2.3.2 Фронтенд

До фронтенду веб-додатка входять веб-сторінки, з якими взаємодіє користувач через браузер. Для написання сторінок використовуються наступні мови:

HTML (Hypertext Markup Language) - це стандартна мова розмітки, яка використовується для створення веб-сторінок. Вона визначає структуру та вміст веб-сторінок за допомогою різних рівнів заголовків, текстових блоків, зображень, відео, посилань та інших елементів. HTML складається з набору тегів, кожен з яких вказує браузеру, як відображати певну частину вмісту.

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для оформлення веб-сторінок. Вона дозволяє задавати зовнішній вигляд елементів HTML, таких як кольори, шрифти, розміри, відступи, рамки та інші стилістичні

властивості. CSS використовується для розділення структури та вмісту веб-сторінок від їх вигляду, що спрощує підтримку та редагування дизайну.

JavaScript - це мова програмування, яка використовується для створення інтерактивних веб-сайтів. Вона дозволяє додавати різноманітну функціональність до веб-сторінок, таку як анімація, динамічне оновлення вмісту, обробка подій користувача, валідація форм, робота з кукісами та багато іншого. JavaScript використовується як для розробки клієнтської частини веб-сайтів (при виконанні коду в браузері користувача), так і для створення серверних додатків (за допомогою платформи Node.js).

Обрані мови дають можливість розробляти повноцінні сторінки з дизайном та функціональністю.

2.3.3 Бекенд

Мова програмування C# (Сі-Шарп) - це об'єктно-орієнтована мова програмування, розроблена корпорацією Microsoft. Вона призначена для розробки різноманітних програм, від десктопних до веб-додатків, ігор, мобільних додатків та багатьох інших. C# є частиною платформи .NET і використовується разом із середовищем розробки Visual Studio для створення програмного забезпечення під різні платформи, включаючи Windows, macOS і Linux. Ця мова є популярним вибором серед розробників завдяки своїй простоті вивчення, потужному функціоналу та широким можливостям інтеграції з іншими технологіями.

ASP.NET - це фреймворк для веб-розробки, розроблений корпорацією Microsoft. Він надає розробникам набір інструментів, бібліотек і механізмів для створення динамічних веб-додатків та веб-сайтів. ASP.NET використовує мови програмування, такі як C# або Visual Basic, для створення серверної частини веб-додатків.

ASP.NET включає в себе різні компоненти, такі як:

- ASP.NET Web Forms: це модель програмування, яка дозволяє розробникам будувати веб-додатки, використовуючи події та контроли, які схожі на класичні віконні додатки.
- ASP.NET MVC (Model-View-Controller): це архітектурний шаблон, який розділяє веб-додаток на модель (дані), представлення (відображення) і контролер (логіку обробки запитів). Він сприяє створенню більш розділених, легко тестових і підтримуваних додатків.
- ASP.NET Web API: це фреймворк для створення RESTful веб-служб, які можна використовувати для взаємодії з веб-додатками за допомогою HTTP-протоколу.
- ASP.NET Core: це перехідна платформа, яка спрощує розробку веб-додатків та розширює їх можливості до різних платформ, включаючи Windows, macOS і Linux.

ASP.NET надає розробникам широкий спектр інструментів і можливостей для створення сучасних та ефективних веб-додатків і веб-сайтів.

Для розробки веб-додатку був обраний шаблон MVC, що зробить код проєкту більш організованим та структурованим.

Microsoft SQL Server (MSSQL Server) - це реляційна система керування базами даних, розроблена корпорацією Microsoft. Вона надає потужні можливості для зберігання, керування та обробки даних у великих корпоративних та підприємницьких середовищах. MSSQL Server підтримує різні мови запитів, включаючи Transact-SQL (T-SQL), яка є розширеним діалектом SQL.

Усі технології бекенду є частиною екосистеми Microsoft, що спрощує розробку веб-додатка.

Висновок. У другому розділі дипломної роботи було розглянуто підходи до створення тренінгів та обрано інструменти та технології для подальшої розробки веб-застосунку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ ІНТЕРАКТИВНОГО ВИВЧЕННЯ БУДОВИ АВТОМОБІЛІВ

3.1 Аналіз вимог

Грунтуючись на даних, отриманих у попередніх розділах, можна сформувати функціональні та нефункціональні вимоги до продукту.

Функціональні вимоги:

- демонстрація матеріалу наступних форматів: текст, малюнки, відео, аудіо;
- пошук матеріалу у додатку за допомогою пошукової строки;
- наявність перевірки після вивчення матеріалу у вигляді тестів (тести включають в себе такі види як однозначний вибір, вірно/невірно, заповнення пропусків, послідовність, зіставлення; до кожного тестового питання повинна бути можливість додати малюнок/аудіо/відео-файл при необхідності);
- система реєстрації/авторизації;
- система коментарів;
- збереження прогресу проходження курсу в акаунті;
- інтерактивні схеми компонентів авто;
- функція пошуку причини та рішення автомобільних несправностей.

Нефункціональні вимоги:

- зручність інтерфейсу;
- мова навчальних матеріалів – українська;
- статті з матеріалами повинні бути структуровані з можливістю перегляду їх по логічному порядку;
- статті повинні мати достатньо графічних матеріалів для більшої наглядності.

3.2 Проєктування веб-застосунку

Після формування вимог проходить процес проєктування програмного забезпечення. Найбільш зручним засобом проєктування програмного забезпечення є створення діаграм мовою UML.

UML (Unified Model Language) – це стандартизована мова, яка використовується для опису та проєктування програмних систем, в першу чергу тих, які використовують об’єктно-орієнтовані парадигми. UML надає набір графічних символів та правил, завдяки яким проєктуються різні аспекти систем, такі як структура, поведінка, взаємодія компонентів та бізнес-процеси.

Завдяки діаграмам можна оформити багаті на деталі та зрозумілі графічні представлення щодо різних аспектів роботи програми, а використання мови UML забезпечує відповідність діаграм загальноприйнятим стандартам.

3.2.1 Варіанти використання

Сформовані функціональні вимоги можна зобразити у вигляді діаграми варіантів використання, яка буде наглядно демонструвати, як користувач може взаємодіяти з веб-застосунком.

Діаграма варіантів використання (Use-Case Diagram) – інструмент у моделюванні системи, який допомагає описати функціональність системи, з якою можуть взаємодіяти актори та які дії вони можуть виконувати.

Рис. 3.1 UML діаграма використання

На зображеній діаграмі (рис. 3.1) позначені основні функції застосунку та зв’язки між ними, наприклад:

1. Асоціація – є зв’язками між акторами та варіантами використання. Наприклад, користувач може знайти статтю через функцію «Пошук», що є варіантом використання.

2. Узагальнення – відношення, яке відмічається пустою стрілкою та використовується для поєднання загального та часткового варіантів використання. На діаграмі такий зв’язок створено між входом в акаунт та авторизацією та реєстрацією. Користувач бажає увійти в акаунт і у нього є вибір – авторизуватись або зареєструватись, що є частковими варіантами.

3. Розширення (Extend) – це відношення включення, елементи якого розширюють варіант використання, але не є обов’язковими при виконанні. В цьому випадку таке відношення є між функціями перегляду уроків та коментуванням. Користувач може переглядати уроки, що також дає можливість йому залишати коментар, але це не є обов’язковою дією. Теж саме стосується перегляду інтерактивних схем та переходу до статей. Учень може переглядати схеми і лише за бажанням переходити до статті, яка його зацікавить.

Створена use-case діаграма дає чітке бачення щодо основного функціоналу, що дозволяє перейти до подальшого проектування програмного забезпечення.

3.2.2 Мапа сайту

Знаючи основні функції веб-застосунку можна спроєктувати мапу сайту.

Мапа сайту – це структурна діаграма, яка представляє архітектуру веб-сайту або веб-застосунку, усі її сторінки та їх ієрархію. Завдяки мапі сайту полегшується розуміння структури платформи і її контенту. Зазвичай, мапи сайтів роблять у вигляді діаграм.

Рис. 3.2 Мапа сайту

На мапі сайтів (рис. 3.2) зображено структуру сторінок. Головною сторінкою є «Головна сторінка», від якої користувач може потрапити на сторінки з пошуком статей, вправами, інтерактивними схемами, інспектором несправностей та входом в акаунт (авторизація/реєстрація). Після пошуку статей людина може перейти до статті. Також після входу в акаунт, користувач може перейти до сторінки Особистий кабінет.

Блок «Стаття/вправа» є репрезентацією серії сторінок, яких може бути багато, але зв’язки з головною сторінкою та пошуком статей є однаковими.

Створена мапа дає уявлення того, як повинна бути розроблена фронтенд частина веб-застосунку на рівні сторінок.

3.2.3 Діаграми класів та баз даних

Далі йде проєктування серверної частини програмного забезпечення.

Через те, що при розробці веб-застосунку буде використовуватись парадигма ООП, і отже, код буде побудований на класах, буде доречним розробити діаграму класів.

Діаграма класів – один із ключових інструментів у мові моделювання UML, який використовується для візуалізації структури класів та взаємозв'язків між ними. Діаграми класів складаються з таких елементів як класи, інтерфейси і різні види зв'язків між ними.

Кожен клас ділиться на такі секції як ім'я, поля класу та його методи (рис.3.3)

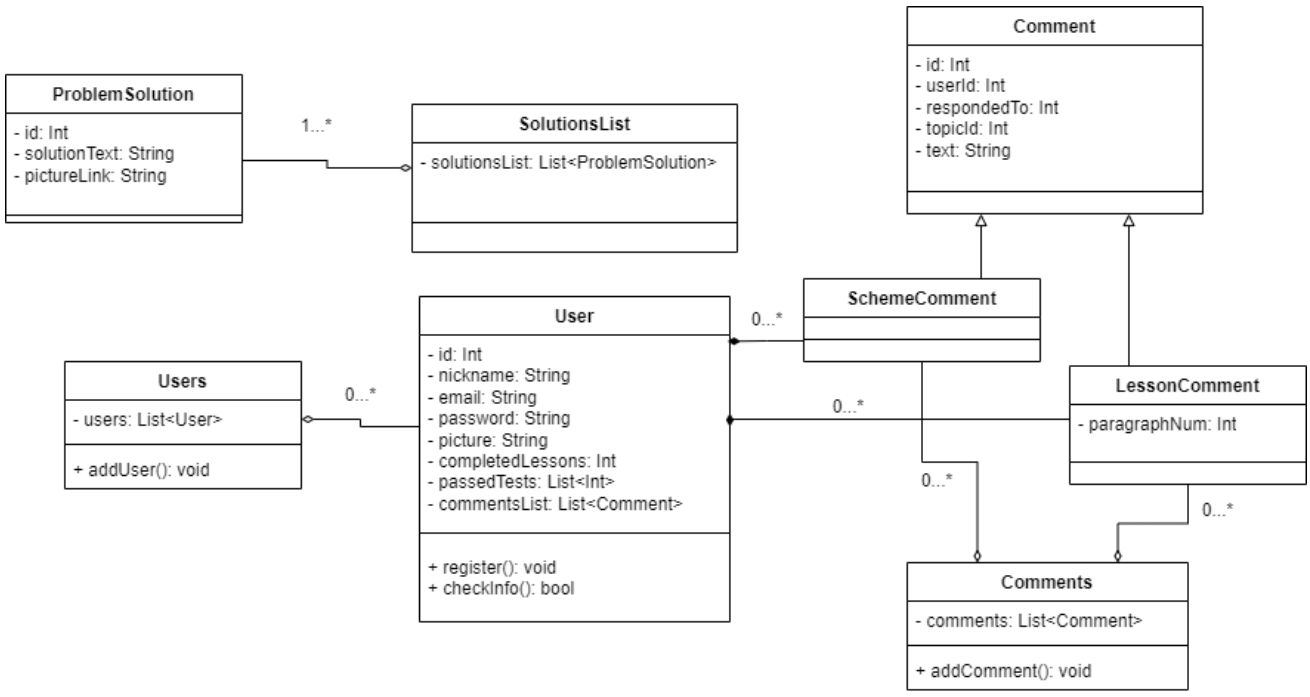


Рис. 3.3 Діаграма класів

Створена діаграма класів має наступні класи:

- *User* – клас, об'єкти якого є акаунтами користувачів. Містить поля відповідні за дані акаунтів такі як nickname (нікнейм), email (електрона пошта), picture (посилання на картинку, якщо користувач має аватар) і т.д. Також є методи для реєстрації і перевірки даних.

- *Users* – клас містить список існуючих користувачів. Акаунти у вигляді об’єктів додаються до списку users List<User>, звідки потім зберігаються в базі даних.
- *Comment* – батьківський клас, від якого наслідуються такі класи як *SchemeComment* та *LessonComment*. Має поля пов’язані з інформацією щодо кожного коментаря та їх вмістом: userId – айді відправника, respondedTo – айді користувача, до якого була створена відповідь (є опціональною), topicId – до якої статті було залишено коментар, text – сам текст коментаря.
- *SchemeComment* – клас, який є нащадком Comment. Відповідає за коментарі залишені до інтерактивних схем. Наслідує усі атрибути батьківського класу.
- *LessonComment* – також є нащадком *Comment*. Є коментарями до уроків та їх частин. Окрім атрибутів батька має власний: paragraphNum – номер частини статті, до якої був залишений коментар.
- *Comments* – клас містить список усіх коментарів. Коментарі потім зберігаються в базі даних.
- *ProblemSolution* – об’єкти класу містять інформацію щодо рішення проблем, які надсилаються клієнту після проходження тесту у Інспекторі несправностей. Має поле для тексту – solutionText та посилання для малюнку – pictureLink.
- *Solutions* – клас, який має список з об’єктами класу *ProblemSolution*.

У кожного атрибуту класу є спеціальні позначки, які показують рівень доступу до кожного поля, а саме:

1) «+» - public – до поля можна отримати доступ поза межами класу;

2) «-» - private – поле доступне лише у межах класу.

Завдяки цім рівням проходить захист даних від непередбаченого доступу до них.

Також між класами є зв’язки наступних видів:

- Агрегація – зв’язок, який показує, що об’єкт одного класу входить до іншого. Наприклад, об’єкти класу *User* входять до об’єкту (його списку) класу *Users*. Кількість об’єктів зазначена як від 0 до нескінченності.
- Композиція – більш строгий варіант агрегації. Об’єкти одного класу не можуть існувати без іншого. Такий зв’язок є між класами *User* та *SchemeComment*. Якщо видалиться користувач (об’єкт класу *User*), то і коментар буде видалений разом з ним.
- Наслідування – один клас є нащадком, а другий – батьківським, що відповідає такому принципу ООП як унаслідування. На діаграмі можна побачити такий зв’язок між батьківським класом *Comment* та класами-нащадками *SchemeComment* і *LessonComment*.

Під час створення діаграми класів було використано такі принципи ООП як:

- Інкапсуляція: обмеження доступу до даних об’єктів для захисту від їх некоректного використання. Було досягнуто за рахунок використання рівнів доступу до даних та створення шляху до даних через методи відповідних класів.
- Успадкування: створення класу на основі вже існуючого, який отримує властивості та методи батьківського класу. Від класу Comment успадковуються класи *SchemeComment* та *LessonComment*.
- Поліморфізм: використання об’єктів нащадків через їх батьківські класи з однаковим інтерфейсом не зважаючи на реальну реалізацію. Цей принцип був використаний для збереження об’єктів класів *SchemeComment* та *LessonComment* у списку класу *User*. Для цього ці об’єкти перетворюються в об’єкти класу *Comment* що дозволяє їх зберігати разом незважаючи на різні класи-нащадки.

Окрім основної програми з класами також буде використовуватись база даних, тому треба зробити діаграму БД (рис. 3.4)

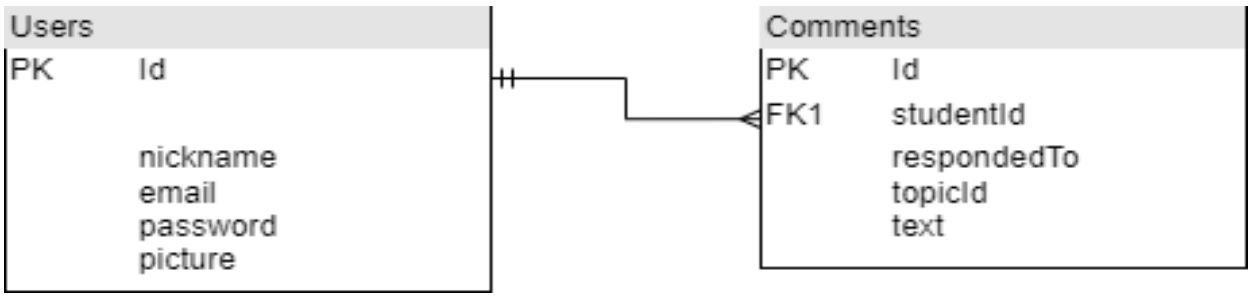


Рис. 3.4 Схема бази даних

Через те, що у базі даних будуть зберігатись акаунти та коментарі, база даних містить дві відповідних таблиці (рис. 3.4). Між таблицею Users і Comments іде зв’язок один і тільки один к багатьом. Це значить, що один User може мати безліч прив’язаних до нього Comment, а Comment може бути прив’язаний тільки до одного User.

Розроблені діаграми були створені для подальшої розробки веб-застосунку. Наступним кроком є програмна реалізація продукту.

3.3 Програмна реалізація

3.3.1 Підготовка середовища

Для розробки веб-застосунку будуть використовуватись інструменти, які було розглянуті у попередніх розділах, а саме: Visual Studio, Visual Studio Code, Git. Також для роботи з базами даних буде встановлено MSSQL Server та MSSQL Management Studio. Усі обрані інструменти є безкоштовними.

При встановленні Visual Studio важливо обрати та завантажити пакет ASP.NET, без якого розробка веб-застосунку на фреймворках цього сімейства не буде можливою.

Після встановлення створюється проєкт. Обираємо назву та шаблон ASP.NET Core MVC.

Для роботи з базами даних завантажуємо пакет для роботи з базами даних через NuGet під назвою *System.Data.SqlClient* (рис. 3.5). Пізніше буде розглянуто роботу з цим пакетом.

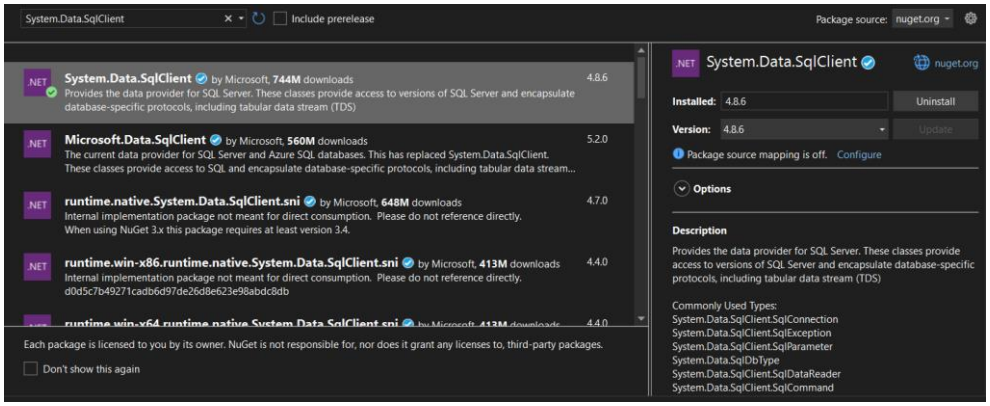


Рис. 3.5 Пакет System.Data.SqlClient

Розглянемо структуру створеного проєкту. Через те, що замість базової версії ASP.NET Core була обрана версія з патерном MVC, проєкт має свої особливості (рис. 3.6).

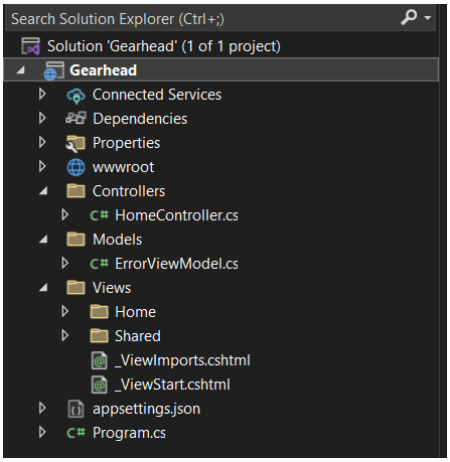


Рис. 3.6 Структура створеного проєкту

MVC – це схема розділення даних застосунку, яка дає можливість розділити застосунок на моделі (дані), представлення (відображення) та контролери (логіку), що робить код проєкту більш структурованим.

Model – описує дані, які використовуються в програмі та логіку, пов’язану з ними.

View – відповідає за візуальну частину та інтерфейс користувача.

Controller – оброблює запити користувача до застосунку та пов’язує моделі з представленнями.

Проєкт містить вже створені папки Models, Views та Controllers для збереження відповідних класів в них.

Окрім цих папок є інші елементи. Розглянемо найбільш важливі:

Program.cs – є вхідною точкою в застосунок;

appsettings.json – конфігурація застосунка;

wwwroot – папка зі статичними файлами, такі як сторінки, скрипти JavaScript, файли CSS, зображення і т.д.

Розглянувши структуру проєкту можна переходити до розробки.

3.3.2 Розробка клієнтської частини

Оформлена раніше мапа сайту дає інформацію щодо того, які сторінки мають бути розроблені. Дизайн та більша частина елементів сторінки є однаковими, тому має сенс розробити шаблон, на основі якого будуть створюватись інші сторінки.

Для верстки сторінок використовується текстовий редактор Visual Studio Code.

```
<script src="script.js"></script>
</head>
<body>
    <nav id="topNavBar">
        <ul>
            <li><img src="images/gearheadLogo.png" id="logo"></li>
            <li><a href="">Головна</a></li>
            <li><input type="text" id="search" name="search" required minlength="2" maxlength="20"/></li>
            <li style="float: right"><a href="">Вхід</a></li>
        </ul>
    </nav>
    <nav id="sideNavBar">
        <ul>
            <li><a href="">Назва вправи</a></li>
        </ul>
    </nav>
    <main>
        <section class="lesson">
            <a name="1"></a>
            <h2>Двигуни. Частина 1. Принцип роботи</h2>
            <img src="images\engine.jpg" alt="engine" title="Двигун!" id="cover" class="postImage">
            <p>Двигуни є...</p>
        </section>
        <div class="comments">
            <div>
                <img src="images/default.png" id="">
                <p class="date"></p>
                <p class="commentText">Comment</p>
            </div>
        </div>
        <div class="test"></div>
    </main>
    <footer>
        <p>2024, Gearhead</p>
        <p>Контакти: admin@gearhead.ua</p>
```

Рис. 3.7 Основа для сторінок

За допомогою мови HTML було створено основу для верстки сторінок (рис. 3.7).

Створена сторінка має наступну структуру:

1)<nav id=”topNavBar”> - блок, який формує верхню навігаційну панель, що містить логотип сайту, перехід на головну сторінку, пошукову строку та кнопку Вхід.

2)<nav id=”sideNavBar”> - блок, який створює стовпчик по ліву сторону сторінки. Цей стовпчик містить елементи з посиланнями на уроки та такі функції як Інтерактивні схеми та Інспектор несправностей.

3)<main> - головною особливістю цієї частини є те, що у кожній сторінці, контент цього розділу буде відрізнятись. Наприклад, на вище продемонстрованому малюнку цей елемент містить секції відповідні за частини статей, коментарі та тестові вправи, у сторінках з інтерактивними схемами ця сторінка буде містити саме схеми і т.д.

4)<footer> - завжди знаходиться внизу і демонструє інформацію щодо сайту.

```
<link rel="stylesheet" href="style.css">
<script src="script.js"></script>
```

Рис. 3.8 Посилання на зовнішні файли

Рис. 3.8 – посилання на файли CSS та JavaScript

Окрім елементів структури сторінки також було додано посилання на файли CSS та JavaScript (рис. 3.8).. У першому розписуються стилі сторінки (рис. 3.9), у другому – скріпти.

```
#logo{
    display: block;
    height: 100px;
}
.active {
    background-color: #c10000;
}
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    position: fixed;
    top: 0;
    width: 100%;
    height: 100px;
    overflow: hidden;
    background-color: #333;
  }
li {
    display: inline;
    height: 100px;
    float: left;
}
li a {
    display: block;
    color: white;
    height: 100px;
    font-size: 35px;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
```

Рис. 3.9 Дизайн верхньої навігаційної панелі

Бокова панель є навчальним курсом зі структурованим списком уроків. Кожен елемент є посиланням, натиснувши на яке, користувач потрапляє на окрему сторінку зі статтею (рис. 3.10).

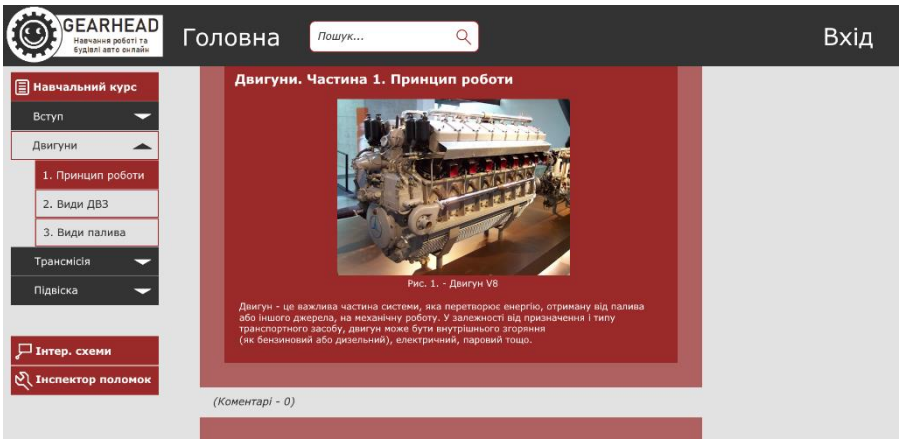


Рис. 3.10 Приклад статті

Кожна стаття має набір навчальних матеріалів різних форматів – текст, зображення, відео, аудіо. Для цього використовуються наступні конструкції в HTML:

<img src=”посилання”> - зображення.

<video src=”посилання”> - відео.

Для більшої зручності, відеоматеріали було завантажено на відео сервіс YouTube та створено посилання через цей елемент. Це не впливає на доступ до відео, користувач все ще може його дивитись через сторінку курсу.

<audio src=”посилання”> - звуковий контент.

Обрані формати навчальних матеріалів доповнюють статті і роблять процес навчання більш ефективним.

Для авторизації та реєстрації було створено окрему сторінку, для переходу до якої треба натиснути на кнопку Вхід на верхній навігаційній панелі (рис. 3.11)

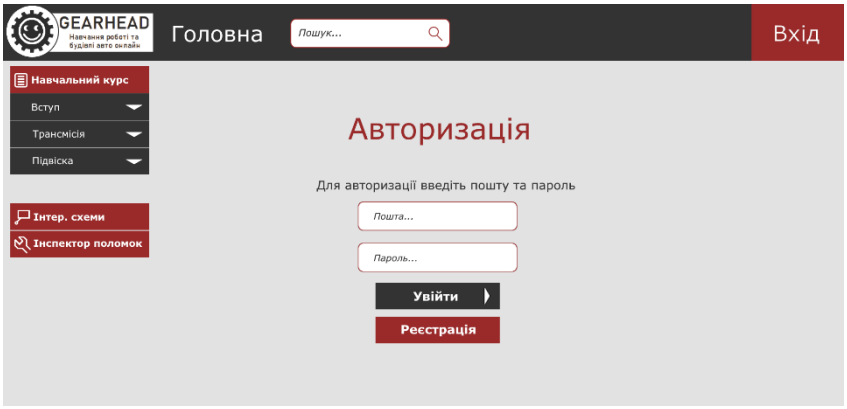


Рис. 3.11 Дизайн форми для авторизації

Сторінка авторизації та реєстрації містить форму, яка складається з полів для пошти і паролю та кнопки для підтвердження даних (рис. 3.12). Форма для реєстрації ще має поле для вводу нікнейму. Також окремою кнопкою було реалізовано зміну форм авторизація/реєстрація.

```
<form method="post">
    <h3>Авторизація</h3>
    <p>Для авторизації введіть пошту та пароль</p><br>
    <input type="text" id="email" name="email"><br>
    <input type="text" id="password" name="password">
    <input type="submit" value="Увійти">
</form>
```

Рис. 3.12 Форма у коді HTML

Після введення даних в ці поля та натискання кнопки підтвердження дані надсилаються на сервер у вигляді запиту. Якщо запит пройшов необхідні перевірки (наприклад, пошта і пароль є вірними), сервер повертає відповідь клієнту і вхід в акаунт проходить успішно. Для того, щоб було зрозуміло, що форма відправляє дані на сервер, до форми було додано атрибут method зі значенням «post».

По схожому алгоритму працює пошук статей. Поле в навігаційній панелі є текстовою формою. Якщо користувач ввів текст в рядок та натиснув enter, сервер отримує текст та шукає ті статті, в назвах яких є введений текст. Сервер повертає назви статей та посилання на них, які вже формуються у список на окремій веб-сторінці. Кожна знайдена стаття перетворюється у елемент списку, який додається у список завдяки таким функціям JavaScript як *.createElement()* та *.append()*.

Також інтерактивні форми є у системі коментарів. Авторизований користувач вводить текст коментаря до статті/іншого коментаря, і дані щодо коментаря відправляються на сервер та формуються у вигляді елемента структури сторінки завдяки вище зазначеними функціями JavaScript.

Ще однією функцією, для якої була створена окрема сторінка є Інтерактивні схеми (рис. 3.13).

Рис. 3.13 Інтерактивна схема

Інтерактивна схема складається з основного зображення, посилань у вигляді кнопок до статті, пов’язаної з обраною деталлю, і зображень-накладок, які роблять окантовку при наведенні на назву деталі.

Усі зображення мають властивість CSS “position: absolute” для того, щоб зображення залишались в одному місці, а не йшли по порядку. Малюнок зі схемою є найнижчим шаром, поверх якого накладаються інші зображення з контурами деталей, які є прозорими, але при наведені на назву деталі стають видимими.

Натиснувши на назву деталі у вигляді кнопки, користувач потрапляє до пов’язаної статті по прямому посиланню.

Після функції Інтерактивні схеми іде функція Інспектор поламок (несправностей), яка також має окрему сторінку (рис. 3.14).

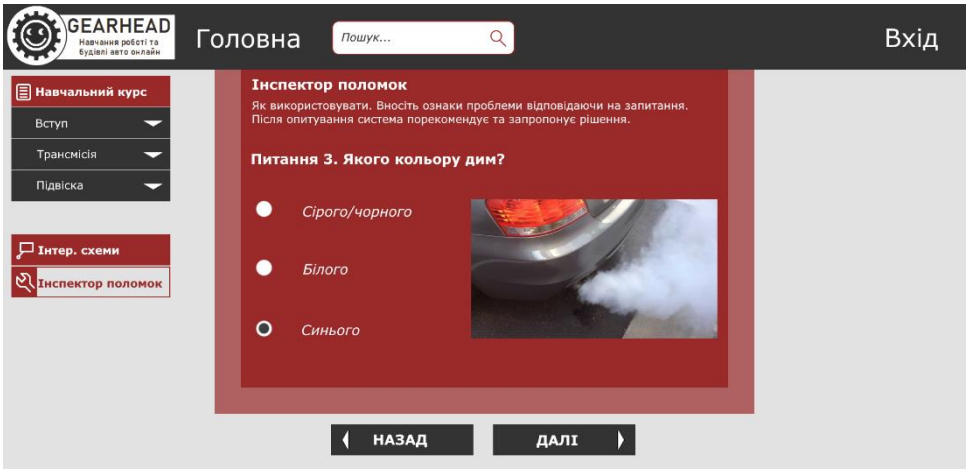


Рис. 3.14 Інспектор несправностей

Інспектор несправностей було вирішено реалізувати у вигляді тестування, під час якого система збирає відповіді і на їх основі дає пораду щодо проблеми. Усі питання є прописаними заздалегідь і всі варіанти відповіді можна уявити у вигляді дерева, у якому з кожним питанням з’являється більше маршрутів, на кінці яких вже є готове рішення.

На початку тестування користувач має змінну зі значенням 0. Кожна відповідь додає певне число, і від отриманої суми залежить наступне

питання/рішення. Якщо людина пройшла всю серію питань, отримана сума надсилається на сервер і звідти отримується відповідь з текстом рішення, яке оформлюється у вигляді сторінки. Також фінальна сторінка може мати посилання на статтю з курсу.

Вправи після статей у курсі також мають у вигляді тестування, але мають більше варіантів взаємодії у формах і, як наслідок, різні типи завдань, наприклад:

Однозначний вибір/заповнення пропусків – вибір одного вірного варіанту. Вибір реалізується за допомогою радіокнопок.

Декілька відповідей - відповідь зараховується, якщо всі вірні варіанти були обрані. Реалізується у вигляді чекбоксів.

Вірно/невірно – потрібно визначити чи вірне твердження чи ні.

Зіставлення – учню потрібно зіставити одні варіанти з іншими. Реалізується за допомогою спливаючих списків.

Порядок дій – користувач повинен обрати дії в правильному порядку.

Секція з вправами є вбудованою в сторінку і завантажується разом з нею без необхідності до окремого звернення до сервера.

Тест вважається виконаним, якщо на всі питання були дані вірні відповіді. Пройдений тест зберігається в статистику користувача і відмічається в назві уроку в списку курсу.

Кількість пройдених вправ авторизований користувач може переглянути в особистому кабінеті (рис. 3.15).

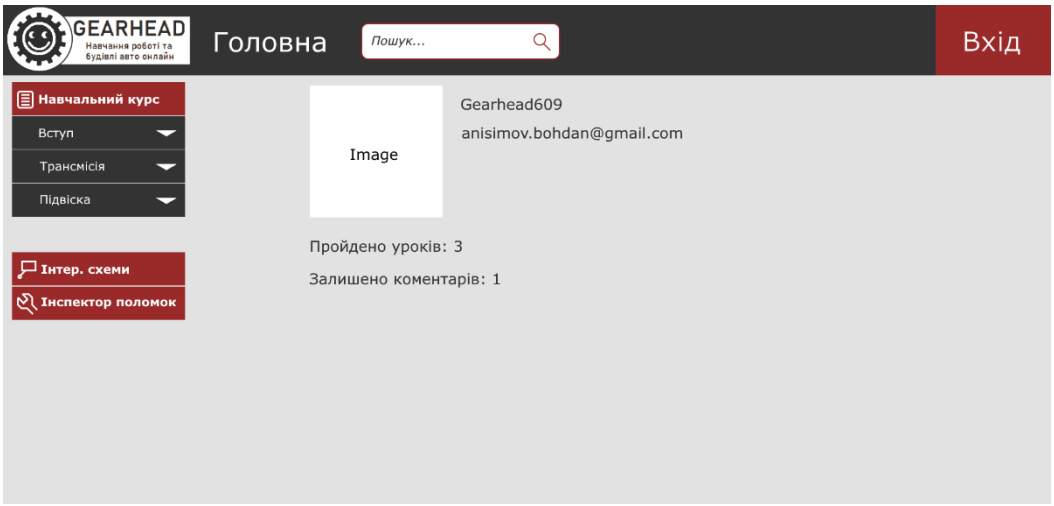


Рис. 3.15 Особистий кабінет

Потрапити в особистий кабінет можна натиснувши на Вхід, що перенесе користувача на сторінку зі своїми даними. Серед даних є нікнейм, пошта, кількість пройдених уроків та кількість залишених коментарів. В подальшому планується додати функцію задання аватару акаунту. При натисканні на кнопку входу посилається запит на сервер, у якому в готову сторінку користувача вставляються його дані і повертаються клієнту.

Усі розроблені веб-сторінки і пов’язані з ними функції потребують реалізації серверної частини додатку для повноцінної роботи, що буде наступним кроком у розробці.

3.3.3 Розробка серверної частини

Раніше було розглянуто структуру проекту ASP.NET Core MVC, яка складається з таких елементів як моделі, структури та представлення. Розглянемо використання цих елементів на практиці.

В каталозі *Views* є папка Home, в якій є файл *home.cshtml*, що є головною сторінкою. В цьому файлі нічого немає, тому внесемо туди код, який був розроблений раніше для головної сторінки сайту (рис 3.16).

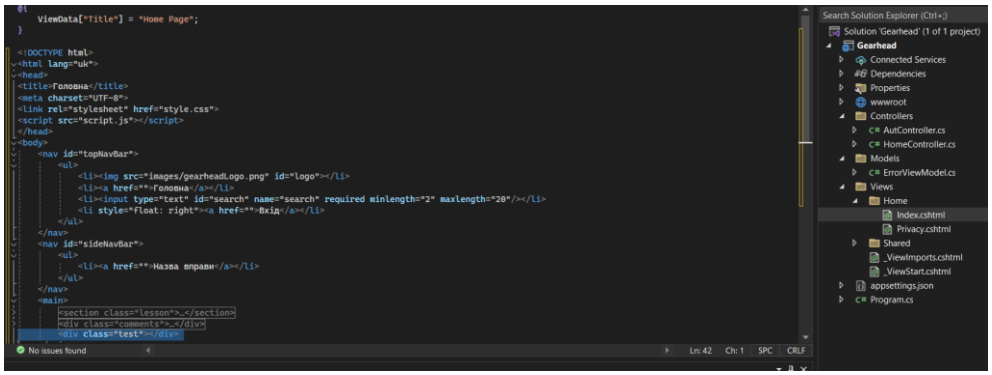


Рис. 3.16 Створення файлу головної сторінки

Також слід додати файли CSS та JavaScript в папку *wwwroot* для того, щоб ці файли були статичними і завантажувались разом зі основною сторінкою. Для кожного з цих типів файлів вже зроблені окремі папки (рис. 3.17).

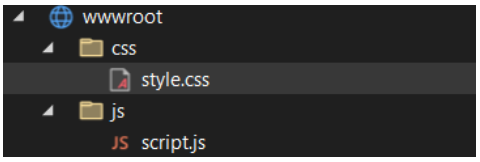


Рис. 3.17 Структура статичних файлів

Для підтримки представлень, в класі *Program* потрібно підключити відповідні сервіси (рис. 3.18):

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();
var app = builder.Build();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}");

app.Run();
```

Рис. 3.18 Підключення сервісів представлень

Останнім кроком є створення контролера, який буде повертати користувачу необхідну сторінку по запиту (рис. 3.19):

```
using Microsoft.AspNetCore.Mvc;

namespace Gearhead.Controllers
{
    0 references
    public class AutController : Controller
    {
        0 references
        public IActionResult LogIn()
        {
            return View("~/Views/Account/Account.cshtml");
        }
    }
}
```

Рис. 3.19 AutController

Для того, щоб був зроблений запит, треба присвоїти кнопці адресу *Home/Index*. У результаті при натисканні на кнопку Вхід у користувача в браузері

відкривається сторінка з формою для аутентифікації. Цей метод використовується і для інших сторінок. Звернення до серверу і до конкретного контролеру через адресну строку є маршрутизацією.

Для створення класів використовується створена діаграма класів. На її основі будуть створені необхідні класи та зв’язки між ними.

Усі класи створюються в папці *Models* так як вони є моделями (рис. 3.20). Для кожного класу створюється окремий файл для кращої структурованості.

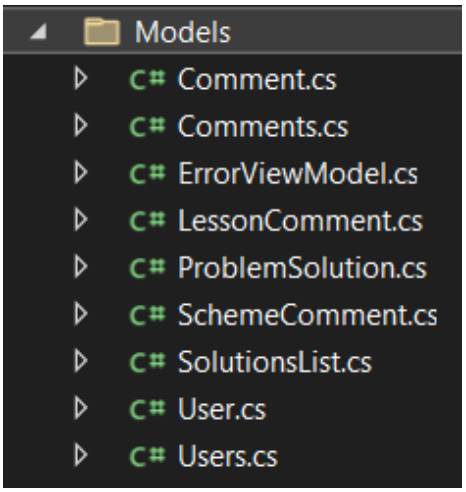


Рис. 3.20 Створені моделі

Через те, що під час проєктування застосунку та розробки архітектури використовується Об’єктно-Орієнтоване Проєктування, між класами було також реалізовано зв’язки.

Наприклад, клас *LessonComment* є нащадком класу *Comment* (рис. 3.21).

```
namespace Gearhead.Models
{
    4 references
    public class Comment
    {
        private int id;
        private int userId;
        private int respondedTo;
        private int topicId;
        private string text;
    }
}
```

```
namespace Gearhead.Models
{
    0 references
    public class LessonComment : Comment
    {
        private int paragraphNum;
    }
}
```

Рис. 3.21 Реалізація успадкування у коді

Деякі класи були зроблені статичними для того, щоб не створювати зайвих об’єктів. Це класи-контейнери, до полів яких треба звертатись для збереження даних у список. Такими класами є *Users, Comments, SolutionsList* (рис. 3.22).

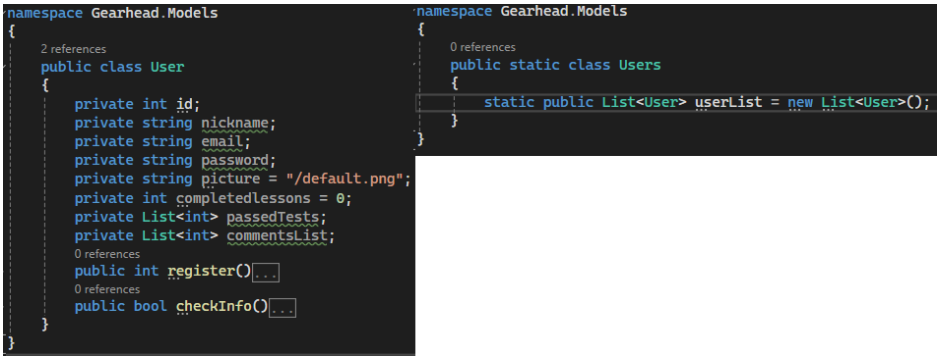


Рис. 3.22 Приклад статичних класів та взаємозв’язку Агрегація

Розглянемо процес обробки даних на прикладі форми для авторизації (рис. 3.12). Після заповнення та підтвердження форми, дані з неї йдуть до серверу. Для цього форма повинна мати атрибут method=”post”. Кожне поле має їм’я, яке буде співпадати з параметрами метода в класі бекенду.

```
using Microsoft.AspNetCore.Mvc;
namespace Gearhead.Models

namespace Gearhead.Controllers
{
    0 references
    public class SigninController : Controller
    {
        [HttpGet]
        0 references
        public IActionResult SignIn()
        {
            return View("~/Views/Account/signin.cshtml");

        }
        [HttpPost]
        0 references
        public IActionResult SignIn(string nickname, string email, string password)
        {
            var newUser = new User(nickname, email, password);
            Users.Add(newUser);
        }
    }
}
```

Рис. 3.23 Створення об’єкту User

На рисунку 3.23 зображено алгоритм створення акаунту на основі даних, отриманих з форми.

Це є прикладом роботи з даними отриманими з форм. Для більшої якості також додається перевірка даних, наприклад, для того, щоб не було створено 2 юзери з однаковими нікнеймами.

Отримані з веб-сайту дані та моделі потрібно зберігати в базі даних. У розроблюваному веб-застосунку у базі даних зберігаються дані акаунтів користувачів та коментарі. Це було зображено у вигляді діаграми баз даних, яка містить 2 таблиці.

Першим кроком для роботи з БД є самостворення бази. Для цього потрібно відкрити раніше встановлений MSSQL Management Studio (рис. 3.24).

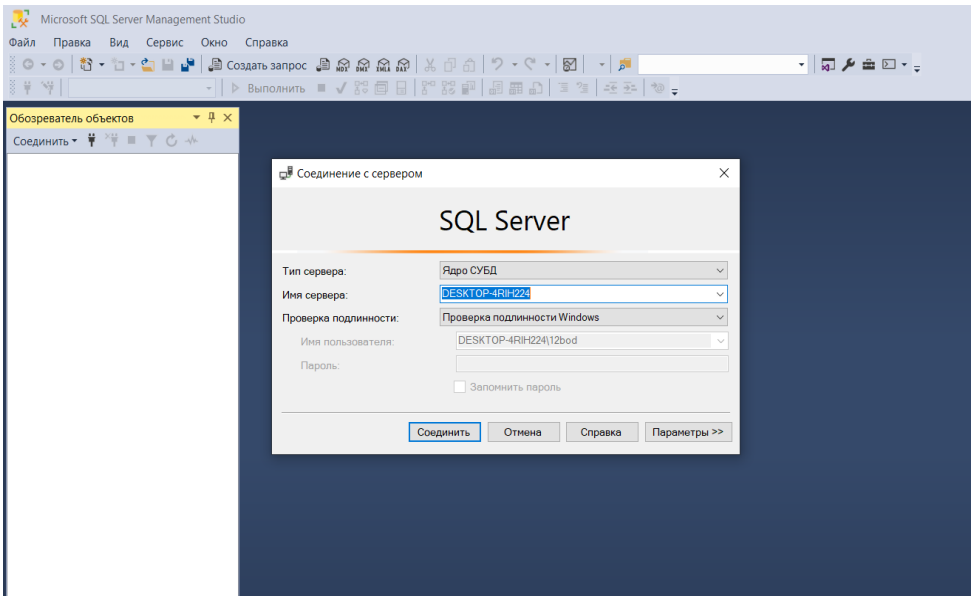
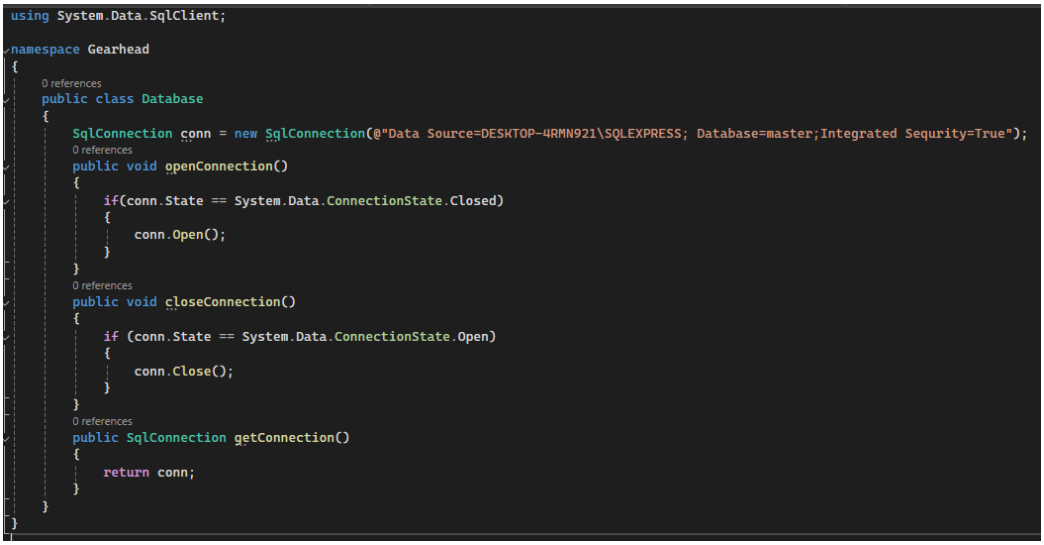


Рис. 3.24 MSSQL Management Studio

Після відкриття SSMS, розробнику пропонується створити БД підключившись до сервера за замовчуванням. Створений сервер та БД будуть локальними.

Введену назву бази даних треба скопіювати. Через неї буде підключено застосунок до серверу БД з коду.

Завантажений пакет *System.Data.SqlClient* потрібно підключити інструкцією *using*. Також створюємо спеціальний клас для бази даних (рис. 3.25).



```
using System.Data.SqlClient;

namespace Gearhead
{
    0 references
    public class Database
    {
        SqlConnection conn = new SqlConnection(@"Data Source=DESKTOP-4RMN921\SQLEXPRESS; Database=master;Integrated Sequrity=True");
        0 references
        public void openConnection()
        {
            if(conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }
        }
        0 references
        public void closeConnection()
        {
            if (conn.State == System.Data.ConnectionState.Open)
            {
                conn.Close();
            }
        }
        0 references
        public SqlConnection getConnection()
        {
            return conn;
        }
    }
}
```

Рис. 3.25 Клас Database

У цьому класі було створено об’єкт класу *SqlConnection* під назвою conn. У його аргументи було вписано назву серверу, назву БД та наявність інтегрованого захисту.

Також було додано наступні методи:

- *openConnection()* – метод, який відкриває з’єднання з БД;
- *closeConnection()* – метод, який закриває з’єднання;
- *getConnection()* – повернення рядка.

Для подальшої роботи з базою даних треба зробити об’єкт *Database* з назвою *db*, через який будуть проходити всі дії з базою.

Спочатку треба підключитизь до бази даних. Для цього є створена раніше команда *db.openConnection()*, після чого можна створити базу даних наступними інструкціями:

db.command.CommandText = "CREATE DATABASE GearheadDB"; //надсилання SQL-виразу для створення бази даних з назвою GearheadDB;

db.command.Connection = db.getConnection(); //визначає підключення.

Щоб виконати команду, використовуємо метод *ExecuteNonQueryAsync():*

await command.ExecuteNonQueryAsync().

Створимо таблицю *User* для збереження акаунтів.

```
Database db = new Database();
db.openConnection();
db.command.CommandText = $"CREATE TABLE User (" +
    "Id INT PRIMARY KEY, " +
    "Nickname STRING NOT NULL, " +
    "Email STRING, NOT NULL, " +
    "Password STRING, NOT NULL, " +
    "Image STRING)";
db.command.Connection = db.getConnection();
app.Run();
```

Рис. 3.26 Створення таблиці для класу User

Таблиця також формується завдяки SQL-виразами (рис. 3.26). Створена таблиця має такі ж характеристики, як і на діаграмі.

Ще SQL-вирази можна використовувати для додавання, редагування, видалення об’єктів таблиць. Наприклад, додавання об’єкта в таблицю:

db.command.CommandText = $“INSERT INTO StudyTask (Id, Nickname, Email, Password, Image) VALUES ({user1.id},

{ user1.nickame }, { user1.email}, { user1.password}, { user1.image})”;

Спробуємо зчитати дані з БД. Для цього є об’єкт SqlDataReader та методи *ExecuteReader()* і *ExecuteReaderAsync() (рис. 3.27)*.

```
db.openConnection();
db.command.CommandText = "SELECT * FROM User";
db.command.Connection = db.getConnection();
SqlDataReader sqlDataReader = await.db.command.ExecuteReaderAsync();
if (sqlDataReader.HasRows())
{
    string str = "";
    while (await sqlDataReader.ReadAsync())
    {
        var id = sqlDataReader.GetValue(0);
        var nickname = sqlDataReader.GetValue(1);
        var email = sqlDataReader.GetValue(2);
        string Concat(str, Convert.ToString(id), " ", Convert.ToString(nickname), " ",Convert.ToString(email), "\n");
    }
}
await sqlDataReader.CloseAsync();
```

Рис. 3.27 Зчитування елементів таблиці

Написаний код є прикладом зчитування з таблиці User. Отримані таким засобом дані можна перетворювати в об’єкти та/або надсилати дані до фронтенду.

Інструкція «SELECT * FROM [Назва таблиці]» вибирає усі об’єкти цієї таблиці.

Функція sqlDataReader.GetValue() повертає значення поля таблиці за порядковим номером.

Механізми, які були розглянуті у контексті роботи з базами даних SQL, становлять основу для обробки та зберігання даних у проекті.

3.4 Тестування

Розроблений веб-застосунок потрібно перевірити на якість та відповідність зазначеним вимогам. Для забезпечення якості програмного забезпечення існує такий напрям як тестування.

Тестування – це перевірка програмного забезпечення на відповідність вимогам. Є багато різних методів тестування, серед яких було обрано ручне тестування з використанням тест-кейсів. Головною причиною вибору є те, що веб-застосунок не має багато функцій, і їх можна перевірити вручну.

Тест-кейс - це сукупність вхідних даних, умов перед виконанням, очікуваних результатів і умов після виконання, розроблена з метою тестування певного аспекту програми або для перевірки відповідності вимогам.

Усі тест-кейси були оформлені у вигляді таблиці 3.1:

Таблиця 3.1

Тест-кейси

Номер тест кейса	**Дії**	**Очікувана поведінка**	**Отриманий результат**
1	1.Запустити сервер	Запуск та робота серверу без помилок.	Сервер запустився та працює без помилок.
2	1.Перейти до головної сторінки за посиланням	Відкривається головна сторінка з якої може переходити до інших. Скрипти та стилі також завантажуються.	Головна сторінка була відкрита. Скрипти та стилі були завантажені та використовуються сторінкою.
3	1.Перейти до сторінки з аутентифікацією	Відкривається сторінка авторизації з формою для даних.	Відкрилась сторінка з формою для даних.
4	1.Перейти до сторінки з аутентифікацією 2.Натиснути «Реєстрація»	Відкривається сторінка реєстрації з формою для даних.	Відкрилась сторінка з формою для даних.
5	1.Перейти до сторінки з аутентифікацією 2.Ввести некоректні дані	Система повідомляє щодо некоректних даних.	Система повідомила щодо некоректних даних.

Продовження таблиці 3.1

Тест-кейси

Номер тест кейса	**Дії**	**Очікувана поведінка**	**Отриманий результат**
6	1.Перейти до сторінки з аутентифікацією 2.Натиснути «Реєстрація» 3.Ввести некоректні дані	Система повідомляє щодо некоректних даних.	Система повідомила щодо некоректних даних.
7	1.Перейти до сторінки з аутентифікацією 2.Ввести коректні дані	Успішний вхід в акаунт.	Успішний вхід в акаунт.
8	1.Перейти до сторінки з аутентифікацією 2.Натиснути «Реєстрація» 3.Ввести коректні дані	Успішне створення акаунту.	Успішне створення акаунту.
9	1.Перейти до сторінки з аутентифікацією 2.Ввести коректні дані 3.Перейти до особистого кабінету	Відкривається сторінка особистого кабінету яка містить інформацію щодо користувача.	Відкрилась сторінка особистого кабінету яка містить інформацію щодо користувача.
10	1.Ввести 1 символ в пошукову строку	Пошук повідомляє, що потрібен запит більше ніж 1 символ.	Пошук повідомив, що потрібен запит більше ніж 1 символ.
11	1.Ввести коректний запит в пошукову строку	Відкривається сторінка з результатами пошуку.	Відкривається сторінка з результатами пошуку.
12	1.Відкрити вправу з курсу	Відкривається сторінка зі статтею, коментарями та практичною частиною.	Відкрилась сторінка зі статтею, коментарями та практичною частиною.
13	1.Пройти практичну частину без акаунту	Система попереджає що результати не буде збережено.	Система попередила що результати не буде збережено.
14	1.Пройти практичну частину з акаунтом допустивши хочаб 1 помилку	Система не зараховує тест.	Система не зарахувала тест.
15	1. Пройти практичну частину з акаунтом без помилок	Система зараховує виконаний тест в статистику юзера. Вправа помічається як Виконана у курсі.	Система зарахувала виконаний тест в статистику юзера. Вправа помітилась як Виконана у курсі.
16	1.Залишити коментар без акаунта	Система не дозволяє залишити коментар та пропонує увійти в акаунт.	Система не дозволила залишити коментар та запропонувала увійти в акаунт.
17	1.Залишити пустий коментар	Система не дозволяє залишити коментар та повідомляє, що коментар має містити хочаб 1 символ.	Система не дозволила залишити коментар та повідомила, що коментар має містити хочаб 1 символ.
18	1.Залишити коментар	Коментар зберігається у системі і з’являється на сторінці. Коментар додається до статистики користувача.	Коментар був збережений у системі і з’явився на сторінці. Коментар додався до статистики користувача.
19	1.Відкрити Інтерактивні схеми	Відкривається сторінка з інтерактивними схемами.	Відкрилась сторінка з інтерактивними схемами.
20	1.Натиснути на позначення деталі на інтерактивній схемі	Відкривається сторінка з курсу з пов’язаної теми.	Відкрилась сторінка з курсу з пов’язаної теми.
21	1.Відкрити Інспектор несправностей	Відкривається сторінка з інспектором несправностей.	Відкрилась сторінка з інспектором несправностей.
22	1.Пройти тестування в Інспекторі несправностей.	Відкривається сторінка з рішенням проблеми та посиланням на статтю з курсу.	Відкрилась сторінка з рішенням проблеми та посиланням на статтю з курсу.
23	1.Натиснути на Головна на навігаційній панелі.	Відкривається головна сторінка.	Відкрилась головна сторінка.

Всього було проведено 23 тест-кейси, з яких 23 пройшли з очікуваним результатом, що демонструє успішність тестування і відповідність вимогам програмного забезпечення.

Висновок. У третьому розділі дипломної роботи було спроєктовано, розроблено з використанням обраних технологій та протестовано веб-застосунок для вивчення будови автомобілів. Створений застосунок відповідає усім зазначеним вимогам і виконує поставлені перед ним задачі.

ВИСНОВКИ

Під час роботи над дипломним проєктом було виконані усі поставлені задачі:

1. Проаналізовано процес вивчення будови та роботи автомобілів і їх компонентів, потенційну аудиторію застосунку та його перспективи.

2. Розглянуто існуючі програмні засоби, знайдено їх переваги та недоліки. Було розглянуто наступні аналоги: HowACarWorks, Green-Way та АвтоПеревірка.

3. Розроблено функціональні та нефункціональні вимоги до розробки веб-застосунку для інтерактивного вивчення будови автомобілів.

4. Досліджено та обгрунтовано інструменти і технології для розробки веб-застосунку та спроєктовано сам застосунок. Були застосовані наступні технології: мова програмування C#, фреймворк AS.NET Core, база даних MSSQL, інструменти Visual Studio, VS Code, Git та набір мов для написання сайтів - HTML, CSS, JavaScript.

5. Розроблено веб-застосунок на основі обраних технологій і з урахуванням зазначених вимог. Застосунок має вигляд сайту з навчальними матеріалами у вигляді курсу та іншими формами інтерактивних матеріалів.

6. Проведено тестування веб-застосунку. Застосунок відповідає зазначеним вимогам.

Створений програмний продукт має спростити процес вивчення будови автомобілів, що може позитивно вплинути на автомобільну грамотність серед населення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Олішевська В. Є. Автомобільний транспорт в умовах переходу від автомобілів з двигуном внутрішнього згорання до електромобілів / В. Є. Олішевська, Г. С. Олішевський, 2022. – С. 68.

2. Ситнік Т. Місце й роль інтерактивного навчання у системі інноваційних технологій в закладах вищої освіти. Вісник Черкаського національного університету імені Богдана Хмельницького. Серія:" Педагогічні науки", 2021. – С. 11-18.

3. Староста В. І. Технології інтерактивного навчання: сутність, класифікація. Науковий вісник Миколаївського національного університету імені В.О.Сухомлинського. Педагогічні науки. 2019. – С. 230-237.

4. Лукянчук Н. Класифікація видів тренінгів. Навчання і виховання обдарованої дитини: теорія та практика, 2013. – С. 272-279.

5. Що таке веб-додаток? [Електроний ресурс] - Режим доступу до ресурсу: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/

6. А.М. Білан, В.Г. Гетта. Методика навчання будови автомобіля. Навчальний посібник, Чернігів, 2012. – С. 7-23.

7. A tour of the C# language [Електронний ресурс] // Microsoft Corporation. – 2023. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

8. Жовтяк І.В., Добришин Ю.Є., Гаркуша В.В. Методичний посібник для проведення навчальної практики частина 1 «Розробка WEB-застосувань ASP.NET MVC на платформі .NET Core» для студентів спеціальності 122 «Комп’ютерні науки» та для спеціальності 121 «Інженерія програмного забезпечення» - К.: Університет економіки та права "КРОК". 2019. – С. 37

9. Гудзовата, О. О., Костирко, В. І., Артищук, І. В. Використання уніфікованої мови візуального моделювання UML (Unified Modeling Language) як інструменту підтримки проектування інформаційниї систем. Підприємництво і торгівля, 2019. – С. 109-111.

10. Застосування UML. Діаграма класів [Електронний ресурс] // Державний університет інформаційно-комунікаційних технологій. – 2020. – Режим доступу до ресурсу: https://duikt.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy?lang=ua&act=view&page=1&category=626&id=8002&sys_link=zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy.

11. C# SQL Server Connection [Електронний ресурс] // Net-informations.com. – 2023. – Режим доступу до ресурсу: http://csharp.net-informations.com/data-providers/csharp-sql-server-connection.htm

12. Коноваленко І.В., Марущак П.О. "Платформа. NET та мова програмування C# 8.0.", 2020.С. – С. 5-6.

13. Крепич С. Я., Співак І. Я. «Якість програмного забезпечення та тестування: базовий курс.», 2020р.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка веб-застосунку для інтерактивного вивчення будови автомобілів та їх компонентів мовою C#

Виконав студент 4 курсу
групи ПД-41
Анісімов Богдан Андрійович
Керівник роботи
К.п.н., доц, доцент кафедри ІПЗ Шевченко Світлана Миколаївна

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - спрощення процесу вивчення будови автомобілів за допомогою веб-застосунку розробленого мовою C#.
- **Об'єкт дослідження** - процес інтерактивного вивчення будови автомобілів.
- **Предмет дослідження** - веб-застосунок для інтерактивного вивчення будови автомобілів та їх компонентів.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати процес вивчення будови та роботи автомобілів і їх компонентів.
2. Розглянути існуючі програмні засоби для вивчення будови та роботи автомобілів та їх компонентів, знайти їх переваги та недоліки.
3. Розробити функціональні та нефункціональні вимоги.
4. Дослідити інструменти і технології для розробки веб-застосунку та спроєктувати застосунок.
5. Розробити веб-застосунок на основі обраних технологій і з урахуванням зазначених вимог.
6. Провести тестування веб-застосунку.

3

АНАЛІЗ АНАЛОГІВ

Показник	HowACarWorks	Green-Way	Avto.Perevirka	Gearhead
Зручний інтерфейс	+	-	+	+
Пошук статей	+	-	+	+
Система облікових записів	Тільки для доступу к курсу	Повноцінна система	Відсутня	Повноцінна система
Система коментарів	-	+	-	+
Система прогресу	-	+	-	+
Закріплення матеріалу	-	+	-	+
Інтерактивні схеми	-	-	-	+
Інспектор несправностей	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Демонстрація статей.
2. Пошук статей.
3. Система тестування.
4. Реєстрація/авторизація.
5. Система коментарів.
6. Система прогресу пройдених уроків.
7. Система інтерактивних схем.
8. Інспектор несправностей авто.

5

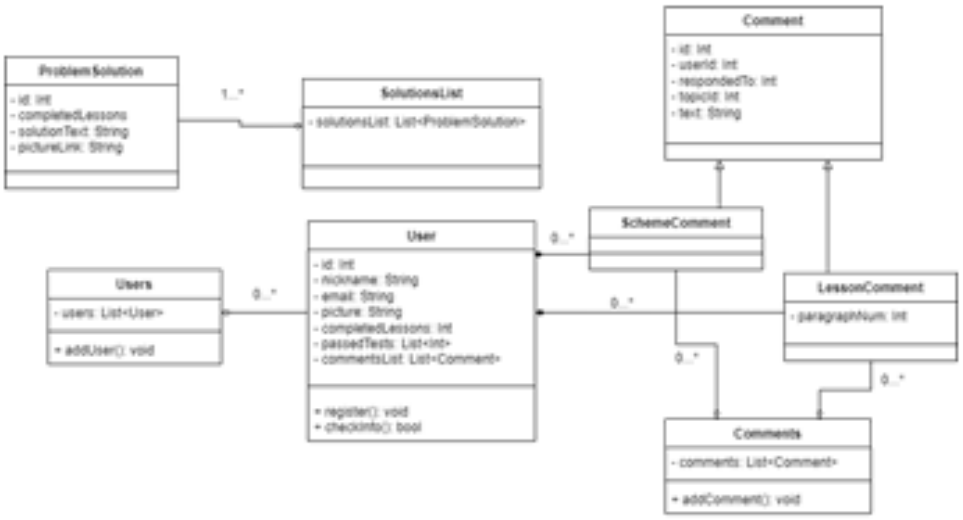
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

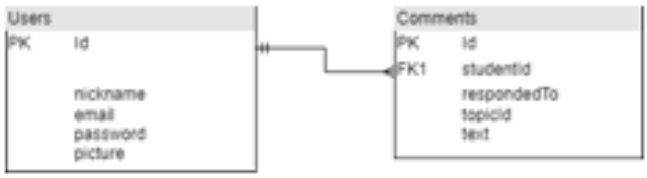
7

ДІАГРАМА КЛАСІВ



8

СХЕМА БАЗИ ДАНИХ



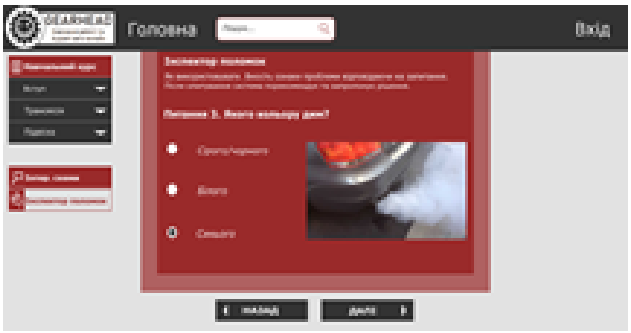
9

МАПА САЙТУ

10

ЕКРАННІ ФОРМИ

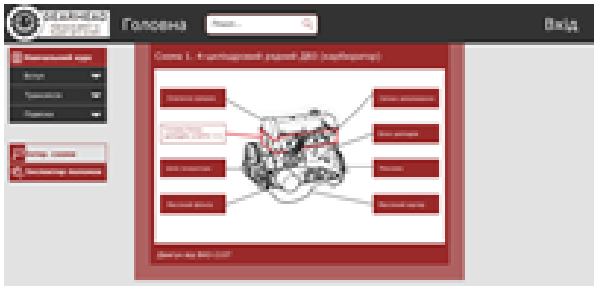
Проходження навчального курсу



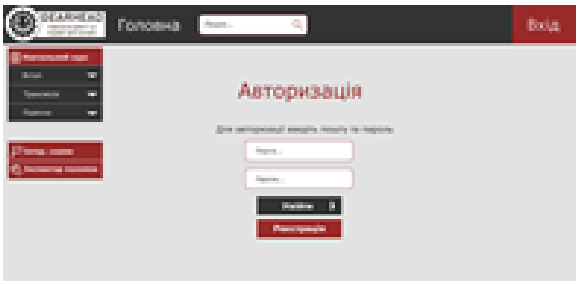
Інспектор несправностей

11

ЕКРАННІ ФОРМИ



Інтерактивна схема



Вхід в акаунт

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Анісімов Б.А., Шевченко С.М. Розробка WEB-застосунку для інтерактивного вивчення будови автомобілів та їх компонентів мовою C#. *Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»*. 24 квітня 2024р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 22 - 23

13

ВИСНОВКИ

1. Проаналізовано процес вивчення будови та роботи автомобілів і їх компонентів.
2. Розглянуто існуючі програмні засоби, знайдено їх переваги та недоліки. Було розглянуто наступні аналоги: HowACarWorks, Green-Way та АвтоПеревірка.
3. Розроблено функціональні та нефункціональні вимоги.
4. Досліджено інструменти і технології для розробки веб-застосунку та спроєктовано сам застосунок. Розглянуто наступні технології: мова програмування C#, фреймворк AS.NET Core, база даних MSSQL, інструменти Visual Studio, VS Code, Git та набір мов для написання сайтів - HTML, CSS, JavaScript.
5. Розроблено веб-застосунок на основі обраних технологій і з урахуванням зазначених вимог.
6. Проведено тестування веб-застосунку. Застосунок відповідає зазначеним вимогам.

14