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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для подорожей Україною "Discovering UA" з використанням HTML, CSS, JS та PHP»

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

_____ Максим ГОРІЛКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-44
Максим ГОРІЛКО

Керівник: Олена НЕГОДЕНКО
к.т.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

Горілку Максиму Владиславовичу

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для подорожей Україною "Discovering UA" з використанням HTML, CSS, JS та PHP»
керівник кваліфікаційної роботи к.т.н., доцент Олена НЕГОДЕНКО,
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:
   3.1. Науково-технічна література.
   3.2. Офіційна документація Bootstrap.
   3.3. Мова PHP.
   3.4. Мова HTML.
   3.5. Мова CSS.
   3.6. Мова JS.
   3.7. База даний MySQL.

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

1. Аналіз існуючих аналогів, методів та технологій розробки веб-застосунків для подорожей Україною.
2. Проектування веб-застосунку для подорожей Україною з використанням HTML, CSS, JS ТА PHP.
3. Програмна реалізація та опис функціонування веб-застосунку.
4. Тестування веб-застосунку для подорожей Україною за допомогою платформи Wrike.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до додатку.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма послідовності.
6. Діаграма активності.
7. Екранні форми.
8. 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 існуючих аналогів для подорожей Україною	14.03-21.03.2024	
4	Проектування застосунку для подорожей Україною	22.03-31.03.2024	
5	Програмна реалізація застосунку	01.04-19.04.2024	
6	Проведення UAT тестування на платформі Wrike	20.04-25.04.2024	
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 _____ Максим ГОРІЛКО
(підпис)

Керівник
кваліфікаційної роботи _____ Олена НЕГОДЕНКО
(підпис)

`

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 54 стор., 1 табл., 26 рис., 15 джерел.

Мета роботи – підвищення популярності та привабливості подорожей Україною за рахунок застосування веб-застосунку створеного мовами HTML, CSS, JavaScript та PHP.

Об'єкт дослідження – процес планування та організації подорожей Україною.

Предмет дослідження – веб-застосунок для планування та організацією подорожей Україною.

Короткий зміст роботи: В роботі проведений аналіз потреб кінцевих користувачів у сфері подорожей, визначено функціональні та не функціональні вимоги до веб-застосунку. Проведено тестування функціонала застосунку, перевірено роботу користувацького інтерфейсу. Для розробки було обрано мови програмування HTML, CSS, JavaScript та PHP, фреймворки Bootstrap та SwiperJS, які представляють с себе прості інструменти для розробки сучасних веб-застосунків, а також використання бази даних MySQL для безпечного та надійного збереження даних кінцевих користувачів.

Ключовими перевагами веб-застосунку для подорожей Україною перед існуючими аналогами є можливість додати подорож в список обраних, наявність системи відгуків та рейтингу, наявність каталогу віртуальних турів для подорожі онлайн, а також реалізації зручної та функціональної CRUD системи для адміністрації.

Сферою використання застосунку є сфера туризму та подорожей.

КЛЮЧОВІ СЛОВА: HTML, CSS, JS, PHP, MySQL, Bootstrap, SwiperJS, веб-додаток, веб-застосунок, веб-сайт.

`

ЗМІСТ

ВСТУП

У наші дні суспільство виявляє все більше зацікавленості у використанні онлайн-ресурсів для забезпечення виконання різноманітних задач. Разом із зростанням популярності таких сервісів з'являється потреба в постійному розвитку нових та оригінальних інструментів для задоволення потреб користувачів.

Одним із ключових напрямків, де дана потреба особливо відчутна, - спрощення процесу вибору та оплати різноманітних подорожей й турів. Традиційні методи, вимагають особистої присутності та виконання складних процедур для оформлення подорожі, часто є малоефективними та часово затратними.

У зв'язку з цим розробка веб-додатку для подорожей має на меті забезпечити зручний інтерфейс для користувачів, автоматизувати процес вибору та оплати подорожей, а також надати доступ до необхідної інформації про різноманітні туристичні місця та послуги.

Об'єкт дослідження – процес планування та організації подорожей Україною.

Предмет дослідження – веб-застосунок для планування та організацією подорожей Україною.

Мета роботи – підвищення популярності та привабливості подорожей Україною за рахунок застосування веб-застосунку створеного мовами HTML, CSS, JavaScript та PHP.

Методи дослідження включають аналіз потреб кінцевих користувачів у сфері подорожей, визначення функціональних вимог до веб-додатку, розробка та тестування програмного забезпечення, а також забезпечення безпеки та захисту інформації користувачів.

Опираючись на поставлену мету, були визначені наступні задачі бакалаврської кваліфікаційної роботи, а саме:

- провести аналіз наявних рішень, що забезпечують планування та

`

організацію подорожей Україною;

- провести аналіз наявних засобів та середовищ розробки веб-застосунку для подорожей Україною;
- розробити функціональні й нефункціональні вимоги до програмного забезпечення для подорожей Україною та спроєктувати модель архітектури;
- розробити веб-застосунок для подорожей Україною, використовуючи мови програмування HTML, CSS, JavaScript та PHP, відповідно до визначених вимог;
- провести User Acceptance Testing веб-застосунку з використанням платформи Wrike для перевірки відповідності розробленого програмного забезпечення вимогам та очікуванням користувачів.

Зважаючи на усі вищезазначені аспекти, виникла потреба у створенні веб-додатка, що забезпечить туристам та мандрівникам повноцінне занурення у захопливий світ відкриттів та неперевершених вражень. Веб-додаток під назвою "DISCOVERING UA" призначений для тих, хто високо цінує подорожі та прагне провести свій відпочинок максимально комфортно та незабутньо. Даний веб-інструмент не лише дозволяє користувачам легко обирати тури, але також гарантує безпечну та ефективну онлайн оплату обраних маршрутів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальні особливості туристичної діяльності в Україні

Туризм в Україні є важливою частиною національної економіки та сприяє розвитку соціально-культурної сфери держави. Загальні особливості туристичної діяльності в Україні визначаються різноманітністю природничих ресурсів, історією та традиціями національної гостинності.

Однією з ключових привабливостей України є її природне багатство. Від Карпат та Кримських гір до прекрасних лісів на Поліссі – країна пропонує різноманітні кліматичні та природні умови для відпочинку. Курорти Чорного та Азовського узбережжя, які пропонують сонячний та морський відпочинок, також здобувають популярність навіть серед іноземців.

Україна славиться своєю багатошаровою історією, яка відображена в архітектурі міст, палаців та церков. Київ, Львів, Одеса, Яремче та інші міста вражають туристів архітектурним різноманіттям та унікальністю пам'яток минулого. Культурні свята та фестивалі, такі як "ГОГОЛЬFEST" чи "КИЇВСЬКЕ ЛІТО", збагачують туристичний досвід та дозволяють глибше відчути дух української культури.

Неповторна гостинність місцевого населення відзначається теплом та щирістю. Традиційні українські страви, рушники та пісні стають частиною туристичного досвіду, надаючи йому унікальний колорит та смак.

Українська туристична інфраструктура постійно розвивається, надаючи різноманітні послуги для різних категорій туристів. Готелі, ресторани, транспортні засоби та екскурсійні агентства працюють на ринку, пропонуючи комфорт та якісний сервіс.

Проте, варто враховувати, що туризм в Україні також стикається з викликами, такими як нестабільність в політичній та економічній сферах, недостатня кількість спеціалістів в області туризму.

1.2 Порівняльний аналіз веб-застосунків для подорожей Україною

В сучасному світі технологій та глобалізації, подорожі стали частиною нашого повсякденного життя. Веб-застосунки для подорожей стали надійними супутниками для мандрівників, допомагаючи їм вибрати напрямок, знайти найкращі готелі, ресторани та екскурсії. У контексті українського туризму, де розмаїття природи, культурні скарби та гостинність створюють неповторний туристичний продукт, аналіз веб-застосунків для подорожей стає актуальним завданням. Цей аналіз дозволить визначити та порівняти сучасні інструменти, які допомагають в подорожах Україною, засвідчуючи, як вони відповідають потребам та очікуванням сучасного туриста. Розглядаючи функціонал, зручність використання та рівень інтеграції з місцевими особливостями, можливо розкрити перспективи та виклики, які стоять перед веб-застосунками у сфері подорожей та туризму в Україні.

1.2.1 Веб-застосунок: KRAINA-UA

KRAINA-UA [1] – веб-застосунок, який присвячений унікальному досвіду туризму в Україні. Заснований у 2007 році, він відзначається великим досвідом у сфері організації турів та подорожей. Основна мета KRAINA-UA – відкривати для своїх відвідувачів та клієнтів невідомі й незвідані перлини України, створюючи неперевершені враження.

Особливості веб-застосунка KRAINA-UA:

- перегляд каталогу турів: надає можливість користувачам ознайомлюватися з різноманітними турами, що пропонуються туристичним агентством;
- детальна інформація про подорожі: забезпечує повний опис кожного туру, включаючи маршрут, призначення та послуги;
- особистий кабінет користувача: надає можливість створити особистий кабінет користувача зі збереженням введених даних.

Недоліки веб-застосунка KRAINA-UA:

- відображення культурної різноманітності: відсутність широкого охоплення різних аспектів культури та природи України у подорожах;
- інтерактивність: потребує покращення взаємодії з відвідувачами через соціальні мережі та інші інтерактивні засоби.

На рисунку 1.1 відображено веб-застосунок KRAINA-UA, який можна вважати прикладом віртуозної інтеграції туристичних послуг в онлайн середовищі.

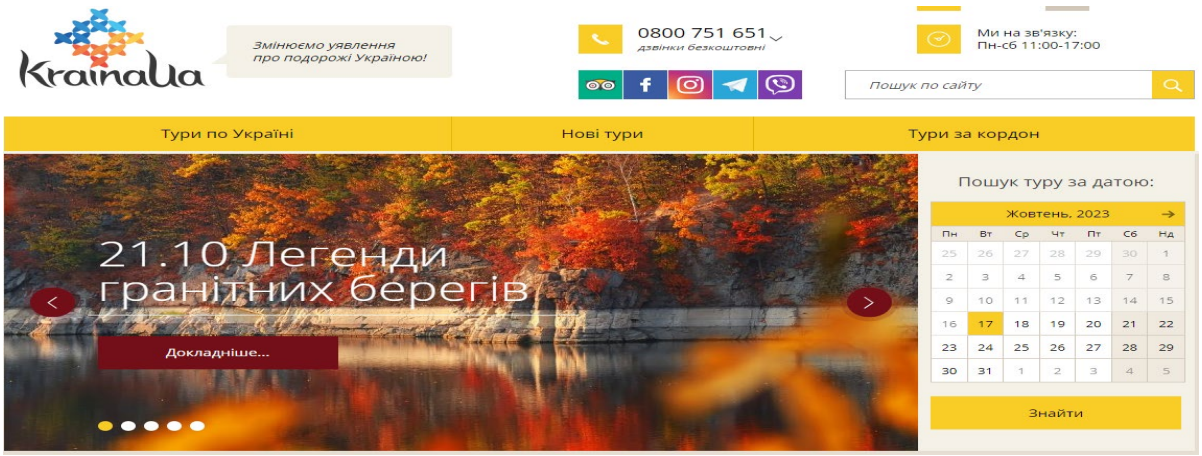


Рис. 1.1 Веб-застосунок KRAINA-UA

Важливо зазначити, що KRAINA-UA може покращити свій інтерфейс, особливо щодо розділу новин та акцій. Здається, що цей розділ має нагромаджений характер, оскільки він включає загальні новини, такі як інформація про продукцію, таку як Apple Watch. Дана практика може вивести користувачів з фокуса та відвести їх увагу від ключових новин та оновлень, пов'язаних з турами та послугами самої компанії.

На рисунку 1.2 відображена проблема веб-застосунка KRAINA-UA.

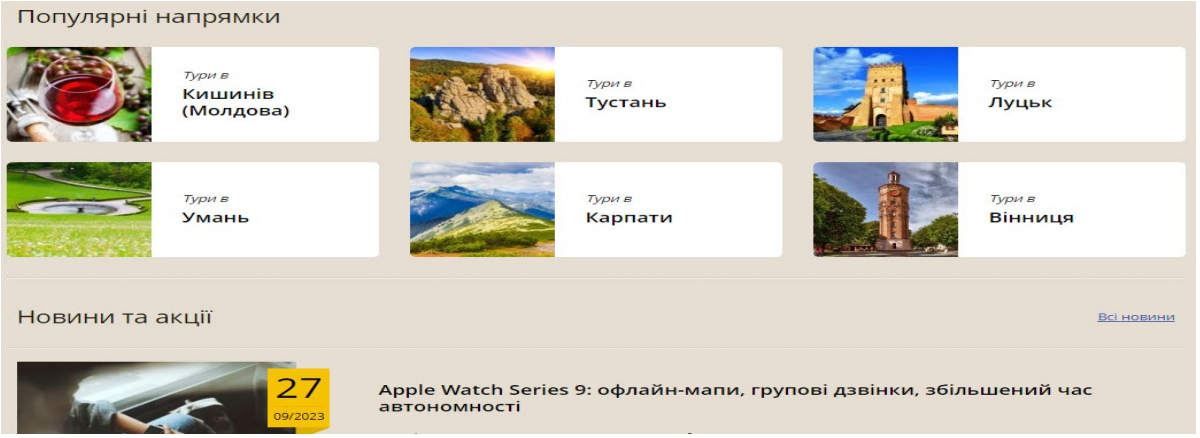


Рис. 1.2 Проблематика веб-застосунка KRAINA-UA

1.2.2 Веб-застосунок: ETNOSVIT

ETNOSVIT [2] – веб-застосунок, що запрошує своїх відвідувачів на виняткові та дивовижно пророблені подорожі по Україні. Заснований у 2014 році, він відзначається своєю великою пристрастю до вивчення та пропаганди багатошарової культури країни. Основна мета ETNOSVIT – розкривати відвідувачам невідомі й незвідані перлини української культури та природи.

Особливості веб-застосунка ETNOSVIT:

- система відгуків: користувачі можуть ділитися своїми враженнями від подорожей, залишаючи відгуки;
- мобільна адаптація: забезпечує приємний користувацький досвід на мобільних пристроях через ефективну адаптацію інтерфейсу.

Недоліки веб-застосунка ETNOSVIT:

- неоднакова доступність інформації: деякі аспекти турів можуть мати нерівномірно заповнену інформацію, що тільки відлякує користувачів;

На рисунку 1.3 відображено веб-застосунок ETNOSVIT, який можна вважати прикладом віртуозної інтеграції туристичних послуг в онлайн середовищі.

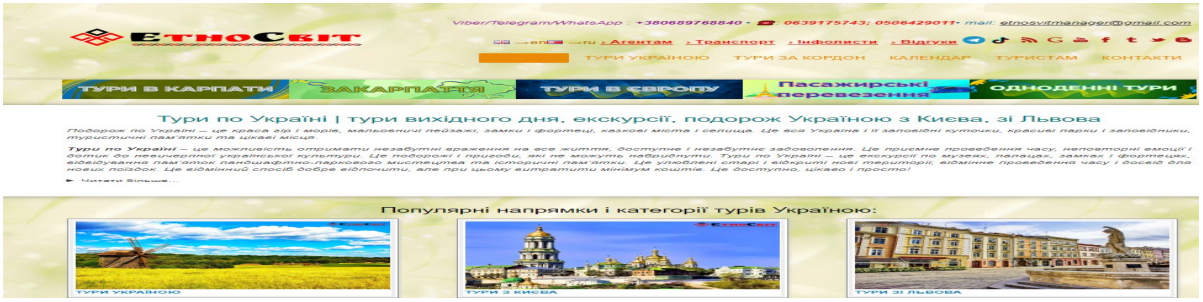


Рис. 1.3 Веб-застосунок ETNOSVIT

Важливо зазначити, що ETNOSVIT має уже застарілий інтерфейс. Здається, що цей інтерфейс має нагромаджений характер, оскільки він включає на головні сторінці: загальні новини, доступні тури, відеоматеріали, контактну інформацію, в деяких моментах власну цільову рекламу і т. д. Дана практика може вивести користувачів з фокуса та відвести їх увагу від різноманіття доступних турів, а власна цільова реклама може здаватися, як непрофесійність, через те, що увага іде не на тури, а на цільову рекламу.

1.2.3 Веб-застосунок: МОЛОДІЖНИЙ ТУРИЗМ

МОЛОДІЖНИЙ ТУРИЗМ [3] – веб-застосунок, який покликаний забезпечити унікальний та дивовижний досвід подорожей. Заснований у 2009 році, МОЛОДІЖНИЙ ТУРИЗМ володіє багатим досвідом у сфері туризму та відзначається високою якістю надання послуг та добре проробленим зворотним зв'язком з користувачами.

Особливості веб-застосунка МОЛОДІЖНИЙ ТУРИЗМ:

- перегляд каталогу турів: користувачі можуть легко переглядати різноманітні тури, які пропонуються туристичним агентством;
- зручний інтерфейс: має привабливий та зручний дизайн, що робить використання веб-застосунка приємним та легким для кінцевих користувачів.

Недоліки веб-застосунка МОЛОДІЖНИЙ ТУРИЗМ:

- надмірна кількість рекламних банерів: перенасичення сторінки рекламними банерами, що впливає на швидкість завантаження веб-застосунку.

На рисунку 1.4 відображено веб-застосунок МОЛОДІЖНИЙ ТУРИЗМ, який можна вважати прикладом віртуозної інтеграції туристичних послуг в онлайн середовищі.

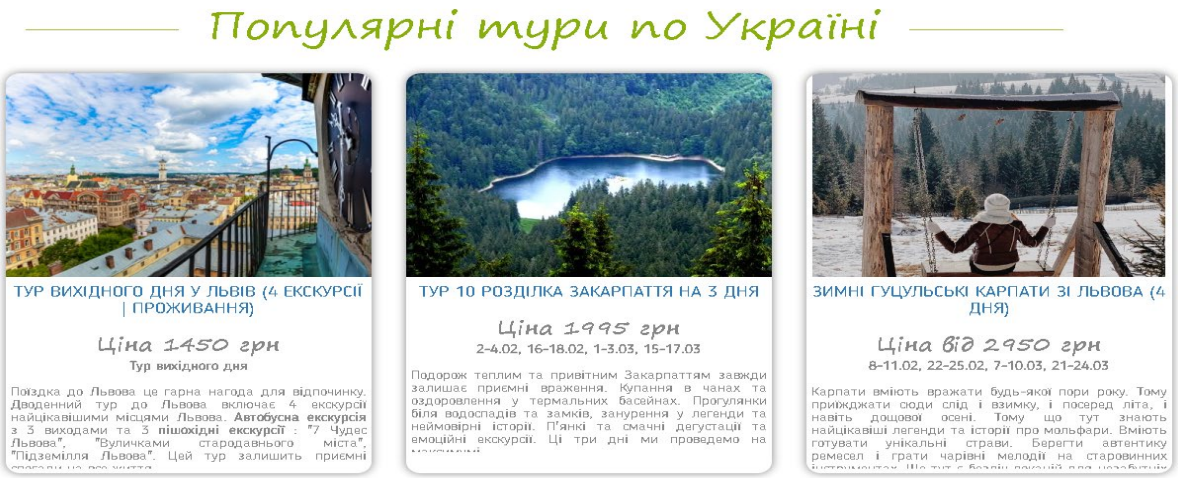


Рис. 1.4 Веб-застосунок МОЛОДІЖНИЙ ТУРИЗМ

Важливо зазначити, що МОЛОДІЖНИЙ ТУРИЗМ має проблему з мовним контентом, а саме з перекладом. Користувачі можуть сприймати неякісний переклад як ознаку непрофесійності та відсутності уваги до деталей.

Таблиця 1.1

Порівняння існуючих програм-аналогів

Характеристика	KRAINA-UA	ETNOSVIT	МОЛОДІЖНИЙ ТУРИЗМ	DISCOVERING.UA
Фільтраційна та пошукова системи	Так	Ні	Ні	Так
Додавання подорожі до списку обраних	Ні	Ні	Ні	Так

Продовження таблиці 1.1

Порівняння існуючих програм-аналогів

Характеристика	KRAINA-UA	ETNOSVIT	МОЛОДІЖНИЙ ТУРИЗМ	DISCOVERING.UA
Наявність системи збереження транзакцій	Ні	Ні	Ні	Так
Особистий кабінет	Так	Ні	Ні	Так
Загальна швидкість роботи застосунку	0.15 s.	0.50 s.	0.10 s.	0.05 s.
Середній час відповіді застосунку	0.46 s.	0.91 s.	0.59 s.	0.21 s.

1.3 Огляд використаних програмних засобів для реалізації

1.3.1 Мова програмування HTML

HTML (або Hyper Text Markup Language) був створений як мова для опису наукових документів, але його загальна структура дозволила з часом адаптувати його для опису різноманітних типів документів і навіть програм [4].

Унікальність HTML полягає в його простоті та легкості вивчення, що робить його ідеальним вибором для початківців. Кожен елемент у HTML визначає свою роль на сторінці, атрибути ж допомагають керувати їхнім зовнішнім виглядом та поведінкою.

HTML як такою не є програмною мовою, але вона використовується для створення структури веб-застосунків, надаючи засоби для впорядкування та представлення інформації. Завдяки своїй універсальності та розширюваності,

HTML забезпечує основу для інших технологій розробки, таких як CSS для стилізації та JavaScript для взаємодії та динаміки.

HTML також постійно розвивається, з'являються нові версії специфікацій, які охоплюють нові можливості та покращення для розробників. Наприклад, HTML5, остання версія стандарту, додає багато нових тегів для реалізації різноманітних функцій на веб-сторінках [5]. Крім того, HTML є основою для багатьох інших технологій та стандартів у розробці, таких як веб-компоненти, що сприяють модульності та повторному використанню коду.

1.3.2 Мова програмування CSS

CSS (або СТИЛІЗОВАНІ КАСКАДНІ ТАБЛИЦІ) [7] можна включити в HTML-документи, щоб контролювати презентаційні аспекти кожного елемента на сторінці. Використання CSS замінило всі функції HTML, які раніше стосувалися представлення. Наприклад, тег <font> застарів, оскільки сімейство шрифтів тепер визначаються правилами таблиці стилів, а саме CSS.

CSS знаходиться на так званому перехресті дизайну та програмування, надаючи зручний спосіб організації стилів, які можна застосувати до одного чи кількох елементів, або навіть до цілих сторінок. Заснована на основі концепції каскаду, CSS дозволяє встановлювати пріоритети стилів та застосовувати їх відповідно до ієрархії елементів.

Одним з ключових переваг використання CSS є можливість розділення змісту та представлення, що полегшує редагування та підтримку веб-застосунків. Завдяки цьому підходу, розробники можуть легко змінювати вигляд сторінок без зміни їх вмісту. Крім того, CSS дозволяє створювати адаптивні та респонсивні дизайни, які підлаштовуються під різні пристрої та розміри екранів.

Ще однією важливою можливістю CSS є можливість використовувати різні селектори для точного визначення, які елементи сторінки будуть застосовувати певні стилі. Це дозволяє розробникам точно контролювати вигляд і поведінку елементів на сторінці та спрощує управління стилями великих проєктів.

CSS продовжує залишатися однією з найважливіших технологій у розробці.

Однією з головних переваг CSS є можливість застосування стилів до різних медіа-типів, що дозволяє створювати адаптивний та доступний контент для широкого спектра пристроїв, включаючи мобільні телефони, планшети, настільні комп'ютери [8].

Важливою частиною розвитку CSS є підтримка нових функцій та модулів. Нові можливості, такі як CSS Custom Properties (змінні), фільтри, блоки властивостей, полігони та багато іншого, розширюють можливості стилізації та дозволяють розробникам забезпечувати ще більшу креативність та індивідуальність у своїх проєктах.

1.3.3 Мова програмування JS

JavaScript [9] є об’єктно орієнтованою мовою програмування (ООП), що є досить важливим фактом і має багато наслідків щодо того, як цю мову можна використовувати. В мові програмування JS не визначаються класи, а створюються об’єкти як екземпляри цих класів. Однією з ключових особливостей JavaScript є його можливість виконувати код безпосередньо в браузері, що дозволяє реалізувати складні функції без необхідності використання серверної взаємодії.

JavaScript також використовується на серверному рівні за допомогою платформи node.JS, що розширює його можливості та робить його важливою частиною повноцінного стека розробки.

JavaScript використовується для створення веб-застосунків, додавання інтерактивності до веб-сторінок, анімації, перевірки даних та вирішення інших завдань. Завдяки широкій підтримці у браузерах, ця мова програмування стала однією з найпопулярніших для розробки в усьому світі.

Останнім часом JavaScript також отримав широке застосування у сфері розробки блокчейн-застосунків. Фреймворки, такі як Ethereum та Web3.js, дозволяють розробникам створювати децентралізовані застосунки з використанням JavaScript, що робить його ключовим інструментом у цьому швидко розвиваючому сегменті технологій.

JavaScript відомий також своєю можливістю взаємодії з HTML та CSS, що

дозволяє динамічно змінювати вміст та стилізацію веб-сторінок під час взаємодії користувача. Ця можливість робить JavaScript важливим інструментом для розробки інтерактивних і відповідних до контексту веб-застосунків.

Крім того, JavaScript використовується для розробки веб-ігор, оскільки він надає широкий спектр можливостей для створення складних графічних ефектів та обробки користувацьких дій. Багато ігор, які працюють у браузері, базуються на JavaScript і використовують різні бібліотеки та фреймворки, такі як Phaser або Three.js.

1.3.4 Мова програмування PHP

PHP (або Hypertext Preprocessor) [10] є серверною мовою сценаріїв, розроблена для мережі Інтернет. PHP використовується для вбудовування динамічного вмісту в HTML, формуючи веб-сторінки буквально на льоту. Він підтримує широкий спектр протоколів, включаючи HTTP та FTP.

Однією з сильних сторін PHP є його адаптація з різними базами даних, такими як MySQL, PostgreSQL, та іншими. Це надає розробникам широкі можливості для обробки та зберігання даних у веб-застосунках.

PHP також використовується для створення динамічних веб-додатків, таких як електронні комерційні платформи, соціальні мережі та форуми. Багато популярних систем управління контентом, таких як WordPress, Joomla та Drupal, базуються на PHP, що робить його невід'ємною складовою веб-розробки.

Розробка на PHP має свої переваги, зокрема, велика активна спільнота та багато готових рішень, бібліотек і фреймворків. Фреймворки, такі як Laravel, Symfony та CodeIgniter, пропонують структурований підхід до розробки, що дозволяє швидко створювати та підтримувати веб-застосунки будь-якої складності.

Крім того, PHP постійно оновлюється та розвивається, додавши нові функції та покращуючи продуктивність. Наприклад, введення PHP 7 призвело до значного покращення швидкодії виконання коду та зменшення споживання пам'яті, що зробило його ще більш привабливим для веб-розробників.

1.3.5 Середовище розробки Visual Code

Microsoft Visual Code [11] - інструмент, призначений для полегшення створення та редагування веб-проєктів і не лише. Він підтримує будь-яку мову програмування завдяки вбудованим розширенням.

Однією з ключових переваг Microsoft Visual Code є його спрощений інтерфейс, який пропонує ефективну робочу область та інтуїтивно зрозумілі інструменти. Розширена підтримка мов програмування, вбудована система контролю версій та інструменти для налагодження роблять його універсальним для роботи.

VS Code активно підтримує розробку на різних платформах і надає можливість гнучкого вибору розширень. Інтеграція з популярними сервісами, такими як Git, а також різними хмарними платформами, робить його ідеальним для командної розробки та сприяє ефективній співпраці.

VS Code має широкий набір інструментів для підвищення продуктивності розробника, таких як автозавершення коду, рефакторинг, пошук та заміна, інтегровані термінали та багато іншого. Крім цього, він має розширений функціонал для роботи з контейнерами та хмарними сервісами, що спрощує налаштування та управління веб-проєктами у будь-якому середовищі.

Однією з важливих особливостей VS Code є його спільнота розробників, яка активно внесла внесок у розвиток різноманітних розширень та плагінів. Це дозволяє користувачам налаштовувати середовище розробки відповідно до їхніх потреб та переваг.

Крім того, VS Code постійно оновлюється та покращується, додаючи нові функції та вдосконалюючи наявні. Інтеграція з широким спектром інструментів розробки та технологій, таких як Docker, Kubernetes, Azure та інші, робить його одним з найбільш універсальних інструментів для веб-розробки.

1.3.6 Фреймворк BOOTSTRAP

BOOTSTRAP [12] найпопулярніший інтерфейсний фреймворк, створений для розробки елегантних й потужних інтерфейсів (UI) для веб-сторінок професійного рівня.

Фреймворк володіє розширеним набором сконфігурованих елементів, що сприяє швидкому та зручному створенню консистентних та привабливих інтерфейсів. Завдяки своїй гнучкості та функціональності, Bootstrap є основною складовою для будь-якого проєкту, спрощуючи завдання веб-розробки та забезпечуючи високий рівень професійного вигляду та функціональності.

Bootstrap також пропонує багато компонентів та модулів для створення різноманітних елементів інтерфейсу, включаючи кнопки, навігаційні панелі, форми, каруселі та багато іншого. Це робить його відмінним інструментом для швидкої розробки прототипів та побудови складних веб-інтерфейсів з мінімальними зусиллями.

Bootstrap також відомий своєю адаптивністю, що робить його ідеальним вибором для розробки мобільних та планшетних версій веб-сторінок. Завдяки вбудованим компонентам, які легко адаптуються до різних розмірів екрана, Bootstrap дозволяє створювати інтерфейси, які оптимізовані для будь-якого пристрою. Ще однією перевагою Bootstrap є активна спільнота зацікавлених розробників, яка надає безліч корисних ресурсів, документацію та допомогу для новачків та досвідчених розробників. Це робить Bootstrap не лише потужним інструментом для швидкої розробки, але і важливим ресурсом для навчання та вдосконалення навичок веб-розробки.

Однією з визначних рис Bootstrap є його можливість налаштовуватися та розширюватися за допомогою власних стилів та компонентів. Розширення та теми, створені спільнотою або індивідуальними розробниками, дозволяють створювати унікальні та індивідуальні веб-інтерфейси, які відповідають потребам проєкту.

1.3.7 Система управління базами даних MySQL

MySQL [13] популярна система керування базами даних SQL з відкритим вихідним кодом, розроблена корпорацією ORACLE. MySQL керує структурованою колекцією даних, а також допомагає додавати та отримувати доступ до даних, що зберігаються в базі даних.

За допомогою MySQL можна вести обробку великих обсягів даних і забезпечити швидкий доступ до них навіть в умовах великого навантаження. Вона підтримує різні типи даних, включаючи текстові, числові, дати та багато інших, що робить її універсальним інструментом для зберігання різноманітних даних.

MySQL також відома своєю високою надійністю і масштабованістю. Вона забезпечує можливість резервного копіювання даних, реплікації для забезпечення високої доступності та кластеризації для розподілення навантаження між серверами. Це дозволяє використовувати MySQL у великих проєктах з високими вимогами до доступності та швидкодії.

MySQL також має широкий спектр інструментів моніторингу баз даних, що дозволяє забезпечити їх ефективне управління та оптимізацію продуктивності. Інструменти, такі як MySQL Workbench, надають GUI для створення та керування базами даних, виконання SQL-запитів та моніторингу їх продуктивності.

Крім того, MySQL активно розвивається та оновлюється, додаючи нові покращення для підтримки сучасних вимог до обробки та зберігання даних. Нові версії MySQL регулярно випускаються з покращеною продуктивністю, безпекою та функціональністю, що робить MySQL однією з найбільш сучасних та потужних систем керування базами даних на ринку.

1.3.8 Локальний сервер XAMPP

XAMPP [14] невеликий і легкий дистрибутив Apache, що містить найпоширеніші пакети для розробки веб-застосунків в одній програмі. Його вміст, невеликий розмір та портативність роблять його ідеальним інструментом для веб-розробки та тестування.

XAMPP надає надійне та безпечне середовище розробки локального веб-сервера для веб-додатків на основі технології PERL та мови програмування PHP. Крім того, він пропонує для використання, такі системи управління базами даних (СУБД), як MariaDB та MySQL.

XAMPP також містить в собі інші корисні інструменти, такі як phpMyAdmin, який дозволяє легко керувати DB, і OpenSSL для шифрування комунікації через HTTPS. Це дозволяє розробникам швидко налаштовувати локальні сервери для розробки веб-застосунків і додатків без складнощів, пов'язаних з окремим встановленням та налаштуванням кожного компонента окремо.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ЗАСТОСУНКА

2.1 Проєктування архітектури веб-застосунка

Створення архітектури веб-застосунка є важливою передумовою для успішної розробки, оскільки саме на цьому етапі визначається план розробки майбутнього застосунку. В цій фазі вирішуються ключові питання щодо взаємозв'язків між компонентами системи, вибору оптимальних фреймворків і бібліотек для реалізації функціональності та забезпечення високої продуктивності та масштабованості. Веб-застосунок зазвичай складається з трьох компонентів: клієнтської частини, яка відповідає за взаємодію з користувачем, серверної частини, що обробляє запити користувачів і взаємодіє з базою даних, а також бази даних, де зберігається інформація, необхідна для роботи застосунку. Крім вибору технологій та визначення архітектури, ще одним важливим етапом проєктування є використання мови моделювання UML (Unified Modeling Language). UML надає стандартний набір графічних інструментів для моделювання складних систем, таких як веб-застосунки, що дозволяє командам розробників чітко визначати структуру та поведінку системи.

Використання UML дозволяє знизити ризик помилок на етапі проєктування, сприяє кращому розумінню системи всіма учасниками проєкту та полегшує подальшу розробку та підтримку програмного забезпечення. Такий підхід допомагає забезпечити якість продукту та його відповідність вимогам користувачів.

Діаграма варіантів використання (або USE CASE DIAGRAM) – діаграма варіантів використання є важливим інструментом для моделювання та розуміння взаємодії між користувачами та системою. Її простота та зрозумілість робить її ідеальним засобом для представлення взаємодії та потреб користувачів. Зазвичай діаграма варіантів використання супроводжується текстовим описом кожного варіанту використання, що допомагає розширити розуміння. Крім того, вона часто використовується разом з іншими типами діаграм, такими як діаграми класів або

діаграми послідовності, для надання більш повного образу системи [15].

Діаграма варіантів використання допомагає виявити основні функціональні можливості системи та їх зв'язок з користувачами. Вона забезпечує візуальне уявлення про те, як користувачі взаємодіють з системою у різних сценаріях використання. Кожен use case описується у вигляді окремої послідовності дій, які здійснюються системою та користувачем для досягнення конкретної мети. Комбінування діаграми варіантів використання з іншими типами діаграм дозволяє здійснити більш ефективний аналіз і проєктування системи, забезпечуючи розуміння як функціонального, так і структурного аспектів програмного забезпечення. Даний підхід допомагає уникнути узагальнень або пропусків у специфікації вимог, що можуть призвести до непорозуміння та помилок у розробці. Такий інтегрований підхід сприяє створенню більш якісного та надійного програмного продукту.

Правильне визначення акторів та варіантів використання є ключовим етапом у розробці програмного забезпечення, оскільки вони визначають область впливу системи та її функціональні можливості. Актори можуть мати різні ролі та взаємодіяти з системою у різних контекстах, тому їхня ідентифікація має бути повною та чіткою.

Варіанти використання визначаються для відображення конкретних сценаріїв взаємодії між акторами та системою. Кожен варіант використання описується послідовністю кроків, які потрібно виконати для досягнення певної мети. Вони можуть бути простими, такими як "АВТОРИЗУВАТИСЯ", або складними, як "ВИКОНАТИ ПЛАТІЖ".

Завдяки використанню діаграм варіантів використання та інших інструментів моделювання, розробники можуть краще розуміти потреби користувачів та ефективно втілювати їх у програмне забезпечення. Це сприяє покращенню якості та коректності розроблюваної системи. На рисунку 2.1 представлена діаграма варіантів використання.

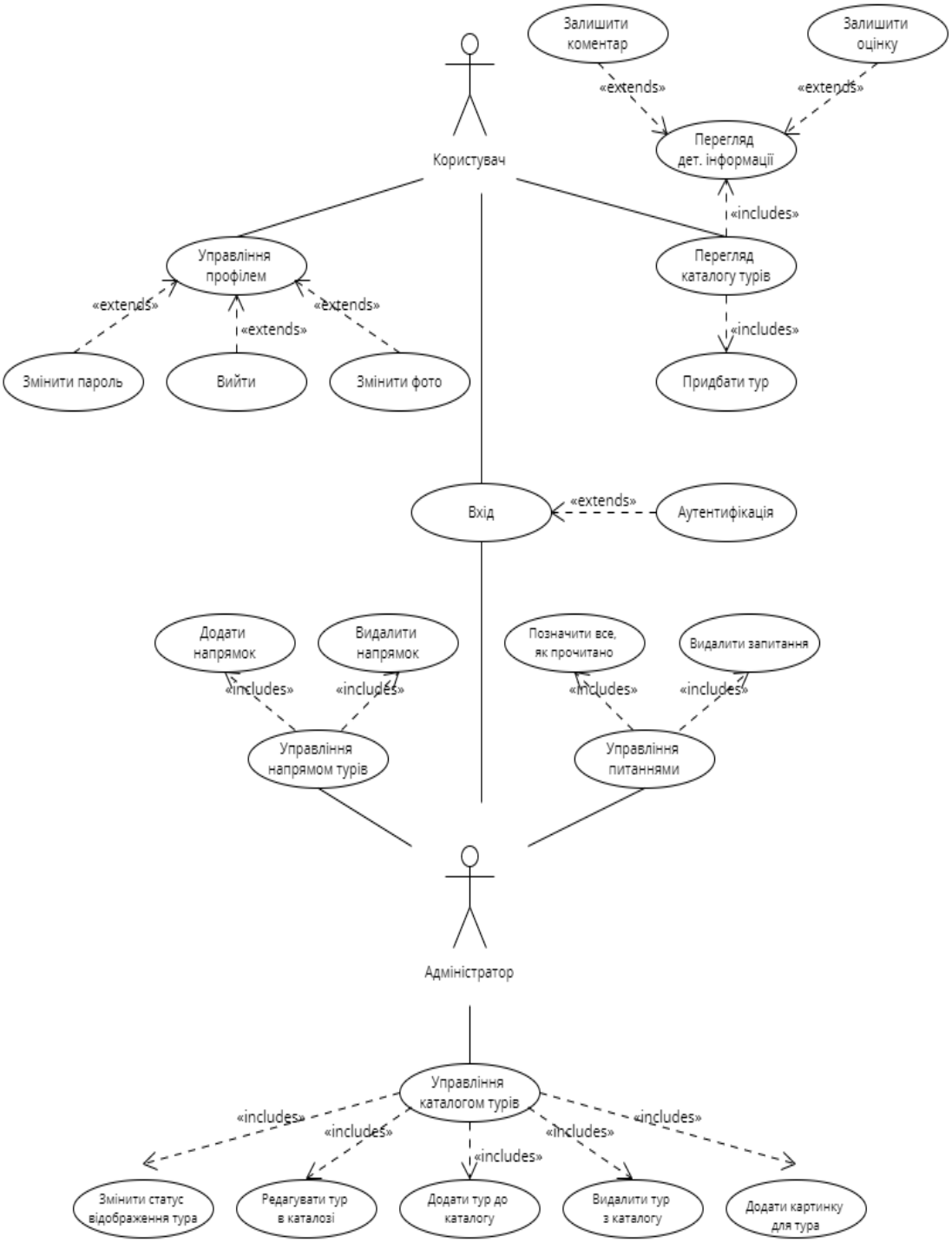


Рис. 2.1 USE CASE DIAGRAM

Діаграма діяльності (або ACTIVITY DIAGRAM) – графічний інструмент, який використовується для моделювання послідовності дій, або процесів у системі. Вона дозволяє візуалізувати різні активності, дії та переходи між ними, що допомагає у розумінні логіки та взаємозв'язків у системі.

Початковий стан (INITIAL STATE) – початковий стан перед початком діяльності зображується за допомогою початкового стану. Процес може мати в основному лише один початковий стан, якщо не зображуються вкладені дії. Загалом використовують чорне зафарбоване коло, щоб зобразити початковий стан системи. Для об'єктів це стан, коли вони створені. Початковий стан із діаграми активності (INITIAL STATE) позначає точку входу та початковий стан активності.

Дія, або стан діяльності (ACTIVITY STATE) – діяльність являє собою виконання дії над об'єктами, або об'єктом. Загалом представляється діяльність за допомогою прямокутника із закругленими кутами. В основному будь-яка дія, чи подія, що відбувається, представлена за допомогою діяльності.

Перехід (TRANSITION) – перехід вказує на зміну стану у системі. Він показує переміщення від однієї дії до іншої та може мати умови, які потрібно задовольнити для здійснення переходу. Зазвичай переходи позначаються стрілками, які з'єднують стани діяльності.

Умова (GUARD CONDITION) – умова визначає, чи може бути здійснений певний перехід. Вона вказує на умову, яку потрібно перевірити перед тим, як здійсниться перехід. Умова може бути вказана біля стрілки, яка показує перехід.

Кінцевий стан (FINAL STATE) – кінцевий стан вказує на завершення активності, або процесу. Він використовується для позначення кінця виконання діаграми діяльності або певної послідовності активностей. Кінцевий стан зображується за допомогою кружечка з внутрішнім кружечком або заливкою.

Використання діаграм діяльності допомагає не лише у візуалізації послідовності дій, а й у виявленні можливих помилок, оптимізації процесів та уточненні вимог до системи. Вона є важливим інструментом для аналізу, проєктування та вдосконалення систем різного роду.

На рисунку 2.2 представлена ACTIVITY DIAGRAM для розроблювальної кваліфікаційної бакалаврської роботи.

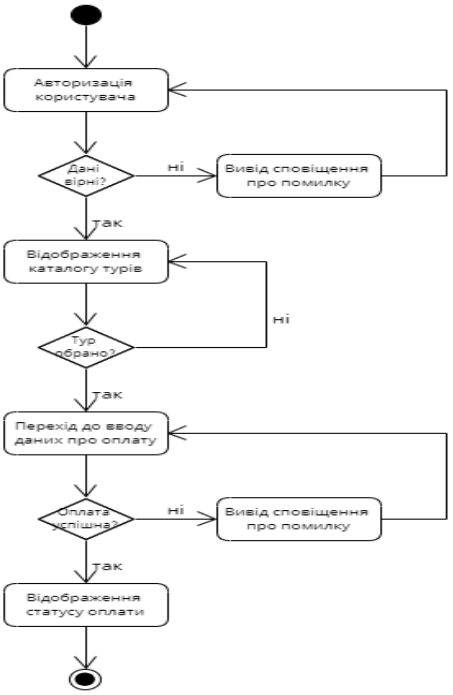


Рис. 2.2 ACTIVITY DIAGRAM

Діаграма послідовності (або SEQUENCE DIAGRAM) є потужним інструментом, що дозволяє візуалізувати взаємодію між об'єктами в послідовному порядку, особливо корисно для розробників програмного забезпечення, які використовують її для моделювання та аналізу виконання програмного коду. Однак, її застосування може виходити за межі сфери розробки програмного забезпечення. Бізнес-персонал також може використовувати діаграми послідовності для ілюстрації та розуміння того, як взаємодіють різні бізнес-процеси та об'єкти в організації. Це допомагає у виявленні можливих шляхів оптимізації та удосконалення бізнес-процесів. Крім того, діаграми послідовності на рівні бізнесу можуть служити як важливий документ для формулювання вимог до майбутньої системи. Діаграма послідовності також може бути ефективним інструментом для навчання та комунікації. Вона дозволяє візуалізувати складні процеси та взаємодії у простий і зрозумілий спосіб, що робить її корисною для навчання новачків у певній галузі, або для пояснення складних концепцій.

Діаграми послідовності можуть також бути використані для моделювання та аналізу систем різного характеру, від інформаційних систем до виробничих процесів. Це дозволяє інженерам та аналітикам отримувати глибше розуміння

робочих процесів та виявляти можливості для їх оптимізації.

Застосування діаграм послідовності може також розширюватися на область управління проєктами. Вони можуть використовуватися для візуалізації послідовності завдань та взаємодії між різними учасниками проєкту, що сприяє кращому розумінню та ефективній координації робочих процесів.

Діаграми послідовності також можуть бути використані для тестування програмного забезпечення. Шляхом моделювання послідовності взаємодії між об'єктами можна створити тести, які перевіряють правильність роботи системи в різних умовах та з різними вхідними даними. Це допомагає забезпечити високу якість програмного забезпечення та підвищити його надійність.

На рисунку 2.3 представлена sequence diagram на якій відображено процес придбання користувачем подорожі.

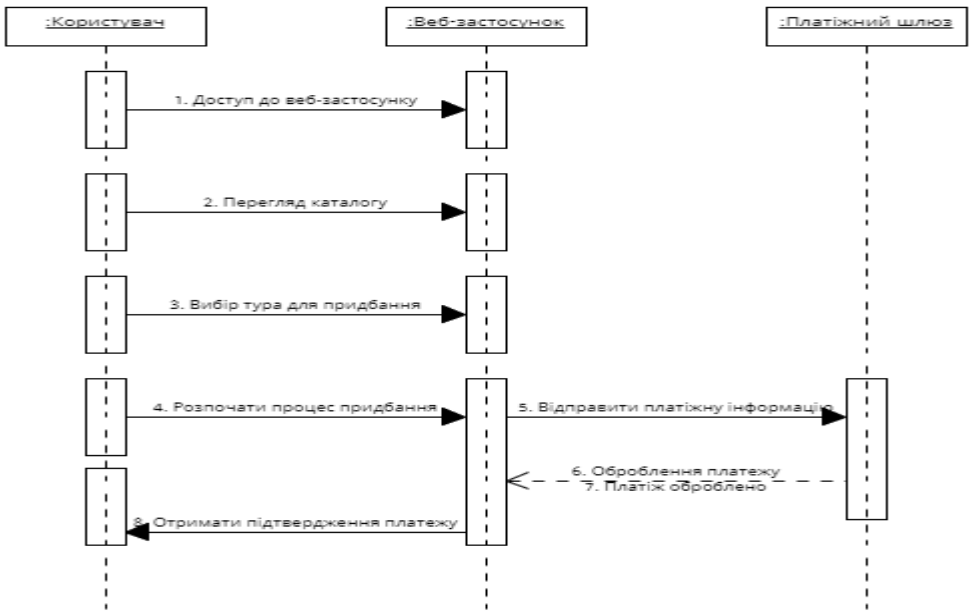


Рис. 2.3 SEQUENCE DIAGRAM

Архітектурне проєктування визначає, як різні елементи програмного застосунку взаємодіють між собою, їх структуру та розподіл обов'язків з метою забезпечення ефективності та зручності розширення у майбутньому. Одним із найпоширеніших підходів у веб-архітектурі є концепція клієнт-серверна. Даний підхід передбачає, що програмний продукт складається з двох основних складових:

клієнтської та серверної частин. Клієнт, зазвичай у вигляді веб-браузера, відповідає за відображення інтерфейсу користувача та збір введених даних, тоді як сервер обробляє запити, виконує бізнес-логіку та надає необхідні дані.

Архітектура клієнт-сервер дозволяє розділити функціональність застосунка на дві частини, що сприяє зменшенню складності системи та полегшує її підтримку та розвиток. Крім того, такий підхід забезпечує більшу масштабованість, оскільки можливе незалежне масштабування клієнтської та серверної частин.

Клієнтська частина зазвичай відповідає за відображення даних та інтерфейсу користувача, а також за взаємодію з користувачем через різноманітні елементи управління, такі як кнопки та форми. Вона також може виконувати певні операції на стороні клієнта для зменшення навантаження на сервер та полегшення роботи з застосунком.

Серверна частина відповідає за обробку запитів від клієнта, виконання бізнес-логіки застосунку та доступ до даних. Вона може взаємодіяти з базами даних, зовнішніми сервісами та іншими серверами для надання необхідної функціональності клієнту. Архітектура клієнт-сервер є основою для багатьох сучасних веб-застосунків та забезпечує їхню ефективну роботу та розширюваність.

Для полегшення підтримки та розширення платформи, вона повинна бути розроблена на основі MULTIEIER ARCHITECTURE, або клієнт → сервер → база даних. В цьому випадку трирівнева клієнт-серверна архітектура — архітектурний підхід, що використовується у розробці програмного забезпечення для розділення функцій програми на три логічні рівні: клієнтський рівень, рівень бізнес-логіки та серверний рівень. Кожен рівень має власну функціональність і взаємодіє з іншими рівнями за допомогою визначених інтерфейсів. На клієнтському рівні знаходиться інтерфейс користувача, який взаємодіє з користувачем програми. Даний рівень може бути представлений веб-браузером, мобільним додатком або іншим клієнтським програмним забезпеченням. Основна функція клієнтського рівня - збирати введені користувачем дані та передавати їх на сервер для обробки.

Рівень бізнес-логіки (також відомий, як середній рівень) містить логіку програми, яка відповідає за обробку даних, виконання операцій та прийняття

рішень. Цей рівень зазвичай містить в собі різноманітні компоненти, такі як сервіси, контролери, моделі даних тощо. Він відповідає за бізнес-логіку програми та виконання операцій, необхідних для взаємодії з базою даних.

Серверний рівень є місцем, де знаходиться база даних та інші ресурси, які використовуються програмою. Він відповідає за зберігання, оновлення та доступ до даних. Серверний рівень може містить в собі бази даних, сервери додатків та інші серверні компоненти, необхідні для забезпечення функціональності програми.

Трирівнева клієнт-серверна архітектура забезпечує розділення обов'язків між різними компонентами програмного забезпечення, що сприяє зручності розробки, супроводу та масштабуванню програм. Даний підхід дозволяє забезпечити високу модульність, розширюваність та ефективність системи. На рисунку 2.4 показано архітектуру веб-застосунка для розроблювальної кваліфікаційної роботи.

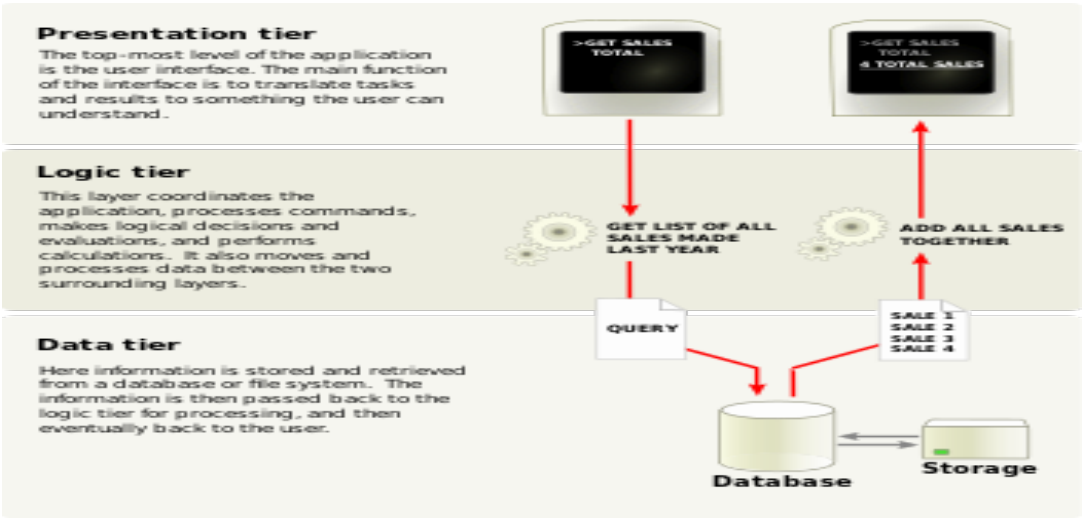


Рис. 2.4 Клієнт-серверна архітектура веб-застосунку

Для зберігання даних було вирішено використовувати систему управління базами даних (СУБД) – MySQL, як рівень доступу до даних користувачів. База даних має забезпечувати централізоване зберігання всіх даних з повною гарантією їх цілісності та узгодженості. Реалізація таких операцій, як: отримання даних їх контроль та узгодженість має здійснюватися за допомогою відповідних процедур, або тригерів.

Бізнес-логіка платформи повинна бути реалізована на сервері та включати такі елементи, як: авторизація кінцевих користувачів, перевірка прав доступу (користувач, або адміністратор) та обробку даних в цілому. Інтерфейс майбутньої системи має бути реалізований на засадах сучасних стандартів, а саме використання мов програмування HTML, CSS, JS та PHP, а також фреймворку Bootsrap.

Для розробки системи використовується інтегроване середовище розробки (IDE) Microsoft Visual Code та система управління базами даних (СУБД) MySQL, а також локальний сервер XAMPP. Для коректного відображення платформи необхідна наявність браузера із підтримкою веб-стандартів, наприклад Google Chrome, або OperaGX.

Дане розроблене програмне забезпечення може виконуватися на абсолютно будь-якій операційній системі, за умови що в даній системі є дротовий доступ до мережі Інтернет та наявність будь-якого сучасного браузера, який підтримує технології HTML5, наприклад Google Chrome.

2.2 Визначення вимог до веб-застосунка

Розробка будь-якої системи, зокрема веб-застосунка, вимагає чіткого визначення вимог. Вимоги можна класифікувати за різними критеріями, одними з найбільш поширених є функціональні, нефункціональні та системні вимоги. Функціональні вимоги визначають, які конкретні функції, або послуги має надавати система, що є основою для взаємодії з користувачами, або іншими системами. Нефункціональні вимоги визначають якості, або обмеження, які повинна мати система, такі як продуктивність, безпека, надійність тощо. Дані вимоги стосуються якості та характеристик системи, а не конкретних функцій. Системні вимоги визначають технічні аспекти інфраструктури, необхідні для реалізації та ефективної роботи системи. Дані вимоги описують обмеження щодо апаратного та програмного забезпечення, мережевої архітектури, сервісів підтримки тощо.

2.2.1 Функціональні вимоги

Функціональні вимоги визначають те, що система повинна робити. Це конкретні функції або послуги, які мають бути доступні користувачам. Вони описують взаємодію між системою та її користувачами, або іншими системами. Наприклад, у даному веб-застосунку функціональні вимоги можуть включати можливість реєстрації користувачів, додавання турів в список обраних, надавати коментар та оцінку туру, можливість переглядати віртуальні тури тощо.

Функціональні вимоги веб-застосунку:

- реєстрація користувачів: можливість створення облікового запису для користувачів з унікальним ідентифікатором та паролем;
- авторизація користувачів: забезпечення доступу до особистого кабінету лише авторизованим користувачам за допомогою логіну та пароля;
- перегляд каталогу турів: користувачі повинні мати можливість переглядати різноманітні тури, які доступні в каталогу подорожей Україною.
- взаємодія з турами: можливість користувачів залишати коментар до туру, а також виставляти рейтинг.

2.2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якості, або обмеження, які повинна мати система. Вони описують якість, продуктивність, безпеку, надійність тощо. Дані вимоги не описують конкретну функціональність, але встановлюють вимоги до того, як система має працювати. Наприклад, у даному веб-застосунку нефункціональні вимоги можуть включати швидкодію системи, час доступності системи, безпеку даних, масштабованість тощо.

Нефункціональні вимоги веб-застосунку:

- продуктивність: застосунок повинен мати можливість обробляти до 100 запитів на хвилину та забезпечувати швидку відповідь користувачам у межах 5 секунд;
- надійність: веб-застосунок повинен мати мінімальний час недоступності (downtime) не більше ніж 99.9% в місяць;

- безпека: забезпечення захищеності персональних даних користувачів, використання шифрування для передачі конфіденційної інформації;
- масштабованість: застосунок повинен бути готовим до масштабування, здатний обробляти збільшене навантаження без значного погіршення продуктивності;
- сумісність: веб-застосунок має підтримувати різні браузери та пристрої, забезпечуючи однаковий функціонал для всіх користувачів.

2.2.3 Системні вимоги

Системні вимоги визначають технічні аспекти інфраструктури, необхідні для реалізації та ефективної роботи системи. Вони вказують на обмеження щодо апаратного та програмного забезпечення, мережевої архітектури, сервісів підтримки тощо. Наприклад, у даному веб-застосунку системні вимоги можуть включати використання певної бази даних, хостинг на певній платформі, використання мережевих протоколів тощо.

Системні вимоги :

- хостинг: веб-застосунок має бути розгорнутий на надійному веб-хостингу з можливістю масштабування ресурсів;
- база даних: використання реляційної бази даних, наприклад PostgreSQL або MySQL, для збереження даних користувачів та інших системних даних.
- контроль версій: використання системи контролю версій, такої як Git, для збереження кодової бази та спільної роботи над розробкою.

2.3 Проектування та розробка структури веб-застосунка

Структура застосунку повинна визначати, з яких саме компонентів вона буде складатися, та як компоненти будуть взаємодіяти між собою. У проєктуванні програмного забезпечення використовуються різні підходи та компоненти, які розробляються для виконання певних конкретних задач, або функціональностей.

Визначення структури та архітектури програмного забезпечення без

перебільшення є важливим етапом у розробці майбутнього програмного продукту. Даний процес містить в собі визначення ключових аспектів платформи, взаємодію між ними, структуру даних та логіку платформи в цілому. Ефективно розроблена архітектура допомагає своєю чергою забезпечити гнучкість, масштабованість та легкість в майбутньому обслуговувані програмного забезпечення.

Під час визначення структури застосунку, розробники також розглядають питання розподілу функціональностей між різними компонентами, їхню взаємодію та залежності. Важливо враховувати не лише поточні потреби проєкту, а й потенційні зміни та розширення у майбутньому.

Структура застосунку може бути організована за різними принципами, такими як модульна, або шарова архітектура, залежно від потреб проєкту та його масштабів. Кожен компонент може мати свою відповідальність та функціональність, яка відповідає за конкретний аспект системи.

Правильно спроєктована архітектура допомагає уникнути перевантаження будь-яких окремих компонентів та забезпечити баланс між їхнім функціоналом та продуктивністю. Вона також полегшує процес розвитку, тестування та підтримки.

Один з ключових аспектів при визначенні структури застосунку - вибір підходу до організації компонентів. Різні методології, такі як монолітна архітектура, мікросервісна архітектура або серверні та клієнтські застосунки, мають свої переваги та недоліки. Важливо обрати той, який найкращим чином відповідає потребам проєкту та можливостям команди розробників.

Після визначення загального підходу до архітектури, розробники розглядають різні компоненти, які будуть складовою частиною системи. Це можуть бути модулі, бібліотеки, сервіси, або інші вузли системи, які виконують певні функції. Правильний вибір компонентів дозволяє створити добре організовану та ефективну систему.

Крім того, важливо визначити зв'язки та взаємодії між компонентами. Це містить в собі визначення інтерфейсів, протоколів комунікації та механізмів обміну даними між різними частинами системи. Ефективне керування взаємозв'язками дозволяє створити гнучку та легко розширювану систему, яка легко адаптується до

змін у вимогах та умовах експлуатації.

Наприклад, на рисунку 2.5 наведено структуру розробленого веб-застосунка. Нижче буде наведено призначення деяких основних елементів застосунку:

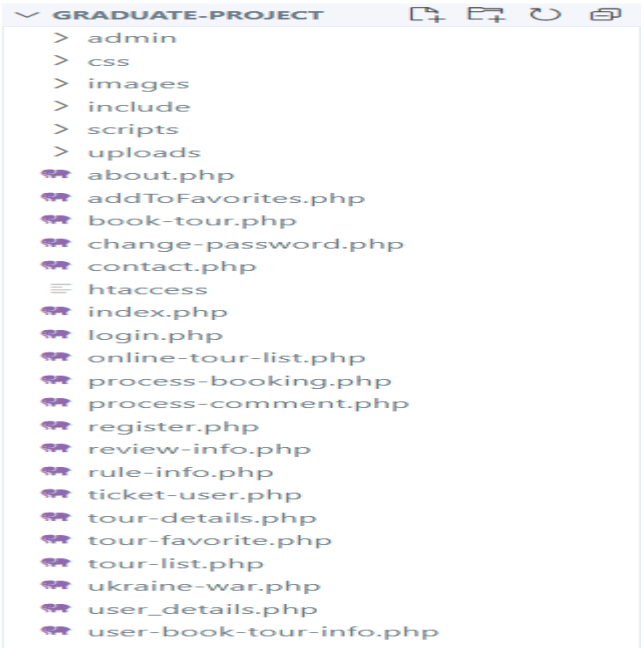


Рис. 2.5 Вигляд структури веб-застосунка DISCOVERING.UA

тека: admin – тека, в якій знаходяться файли та скрипти, які необхідні для роботи з власно розробленою aPanel;

тека: css – тека, в якій знаходиться файл конфігуратор стилів, а саме: style.css;

тека: images – тека, в якій знаходяться зображення для всіх турів, а також за допомогою PHP, в даній теці зберігаються нові зображення для турів;

тека: include – тека, в якій знаходяться всі необхідні файли, для роботи веб-сайту, наприклад: файл header.php, який викликається в index.php для генерації header на веб-сайті в файлі index.php;

тека: scripts – тека, в якій знаходяться всі необхідні скрипти для роботи, наприклад файл: greating-admin, який при заході в панель адміна, виводить адміністратору повідомлення: ДОБРОГО ДНЯ, АБО РАНКУ залежно від часу;

тека: uploads – тека, в якій зберігаються завантажені користувачами аватари. Наприклад, якщо користувач змінює собі аватар, то зображення, яке він завантажує для відображення зберігається саме в даній теці;

файл: about.php – файл веб-сайту, який являє собою відображення загальної інформації про компанію;

файл: addToFavorites.php – файл веб-сайту, який являє собою логіку, яка відповідає за процес додавання користувачем туру в список обраних;

файл: book-tour.php – файл веб-сайту, який являє собою логіку, яка відповідає за процес оформлення користувачем туру;

файл: change-password.php – файл веб-сайту, який являє собою логіку, яка відповідає за процес зміни пароля через особистий кабінет користувача;

файл: contact.php – файл веб-сайту, який являю собою відображення контактної інформації компанії;

файл: index.php – файл веб-сайту, який являє собою головну сторінку платформи (або домашню сторінку, homepage);

файл: login.php – файл веб-сайту, який являє собою логіку, яка відповідає за процес авторизації користувача;

файл: online-tour-list.php – файл веб-сайту, який являє собою відображення каталогу віртуальних турів для перегляду користувачем;

файл: process_comment.php – файл веб-сайту, який являє собою логіку, яка відповідає за процес залишення відгуку та сформування загального рейтингу туру;

файл: process-booking.php – файл веб-сайту, який являє собою логіку, яка відповідає за процес оформлення користувачем туру;

файл: register.php – файл веб-сайту, який являє собою логіку процесу реєстрації користувача на веб-сайті;

файл: review-info.php – файл веб-сайту, який являє собою відображення сторінки, де користувачі можуть щось запитати в сл. підтримки веб-сайту;

файл: ticket-user.php – файл веб-сайту, який являє собою відображення інформації про проведені транзакції на веб-сайті, які робив користувач;

файл: tour-details.php – файл веб-сайту, який являє собою відображення детальної інформації про той чи інший тур користувачу;

файл: tour-favorite.php – файл веб-сайту, який являє собою відображення каталогу турів, які користувач додав як обрані;

файл: tour-list.php – файл веб-сайту, який являє собою відображення всього доступного каталогу турів для користувача;

файл: ukraine-war.php – файл веб-сайту, який являє собою відображення віртуальної подорожі, де описується війна в Україні;

файл: user_details.php – файл веб-сайту, який являє собою відображення детальної інформації про користувача (його особистий кабінет);

2.4 Проєктування та розробка бази даних веб-застосунка

Проєктування БД є важливим етапом для забезпечення ефективного зберігання, організації та обробки великого обсягу даних. Розробка бази даних – складний та відповідальний процес, який передбачає не лише створення структури для збереження даних, але і врахування вимог до їх подальшого ефективного використання. Процес розробки та підняття бази даних охоплює процес проєктування самої структури, вибір оптимального методу, який буде використовуватися для зберігання та обробки даних. Під час проєктування бази даних, розробники повинні уважно аналізувати потреби бізнесу та вимоги до даних, щоб визначити оптимальну структуру та відносини між таблицями. Важливо ретельно вивчити взаємозв'язки між різними сутностями та визначити нормалізацію для забезпечення ефективності та цілісності бази даних.

Після визначення структури бази даних, важливо вибрати оптимальний метод зберігання та обробки даних. Це може містити в собі вибір між реляційною та нереляційною базами даних, використання спеціальних технологій для обробки великих обсягів даних або використання різних типів баз даних для різних потреб.

Не менш важливою є і вибір оптимальної моделі бази даних, чи то реляційної, NoSQL або гібридної, залежно від конкретних потреб та характеристик проєкту. Кожен тип бази даних має свої переваги та обмеження, тому вибір потрібно робити з урахуванням специфіки завдань, які вона повинна вирішувати.

Особлива увага також приділяється забезпеченню безпеки та захисту даних, оскільки збереження конфіденційності та цілісності інформації є критично

важливим для будь-якої організації. Це містить в собі не лише визначення прав доступу до даних, а й ретельний моніторинг та аудит дій користувачів для запобігання несанкціонованому доступу.

Додатково, важливим аспектом є резервне копіювання та відновлення даних, щоб уникнути втрати інформації в разі аварійних ситуацій. Такі процедури дозволяють швидко відновити роботу системи після будь-яких негативних подій.

Крім того, застосування різних методів шифрування гарантує, що конфіденційна інформація залишається захищеною від несанкціонованого доступу. Використання сучасних шифрувальних алгоритмів дозволяє забезпечити високий рівень безпеки, навіть у випадку злому системи. Завершальним етапом є тестування та оптимізація бази даних, щоб забезпечити її ефективну роботу та високу продуктивність під час операцій. На рисунку 2.6 представлено схему бази.

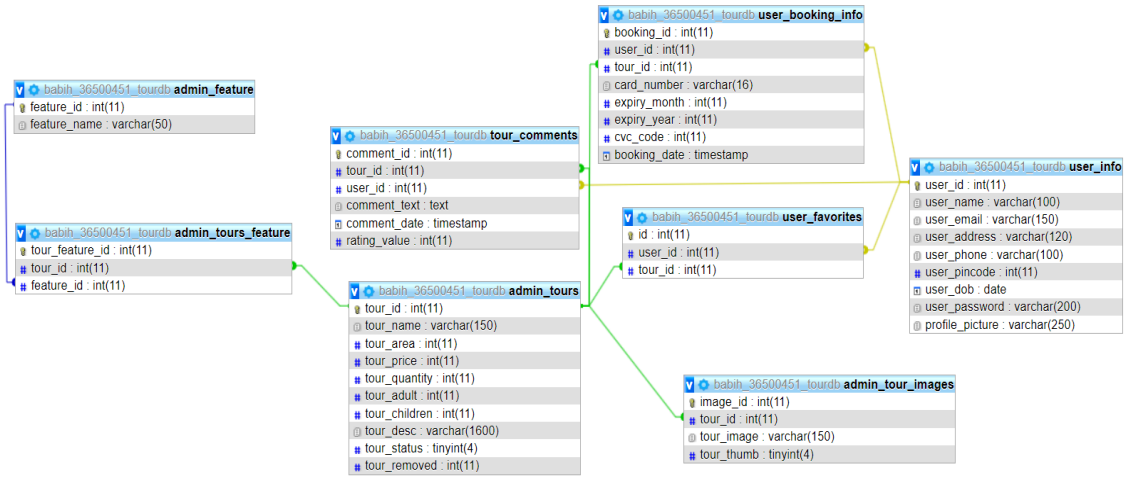


Рис. 2.6 Схема БД застосунка DISCOVERING.UA

Наприклад, таблиця user_info зберігає інформацію про кінцевих користувачів платформи, які пройшли успішно процес реєстрації. Вона містить в собі наступні поля, а саме:

- user_name: ім'я кінцевого користувача, яке він вказав в процесі проходження етапу реєстрації;

- user_email: ел. адреса кінцевого користувача, яку він вказав в процесі проходження етапу реєстрації;

- user_address: місце проживання кінцевого користувача, яке він вказав в процесі проходження етапу реєстрації;

- user_phone: номер телефону кінцевого користувача, який він вказав в процесі проходження реєстрації;

- user_pincode: поштовий індекс кінцевого користувача, який він вказав в процесі проходження етапу реєстрації;

- user_dob: дата народження кінцевого користувача, яку він вказав в процесі проходження етапу реєстрації;

- user_password: пароль кінцевого користувача, який він вказав в процесі проходження етапу реєстрації;

- profile_picture: аватар кінцевого користувача, який він обрав в процесі проходження етапу реєстрації.

Наприклад, таблиця admin_tours зберігає інформацію про додані тури адміністратором платформи. Вона містить в собі наступні поля, а саме:

- tour_name: назва туру, яку вказав адмін платформи під час процесу додавання туру в каталог;

- tour_gid: кількість екскурсоводів, яку вказав адмін платформи під час процесу додавання туру в каталог;

- tour_price: ціна туру, яку вказав адмін платформи під час процесу додавання туру в каталог;

- tour_quantity: кількість доступних білетів для оформлення, яку вказав адміністратор платформи під час процесу додавання туру в каталог;

- tour_adult: доступні місця для дорослих, яку вказав адміністратор платформи під час процесу додавання туру в каталог;

- tour_children: доступні місця для дітей, яку вказав адміністратор, під час процесу додавання туру в каталог;

- tour_status: статус туру, який вказує адміністратор під час процесу редагування туру в каталозі (наприклад, можливо сховати тур з каталогу);

2.5 Проєктування користувацького інтерфейсу веб-застосунка

Проведено проєктування користувацького інтерфейсу для веб-застосунку. Основною метою розробки інтерфейсу було створення зручного та ефективного засобу взаємодії з подорожами. Під час проєктування було проведено аналіз потреб кінцевих користувачів. Були здійснені дослідження для визначення функціональних можливостей, які повинен містити готовий програмний продукт, а також для ідентифікації найбільш доцільних елементів інтерфейсу для використання.

У процесі проєктування велика увага була приділена досягненню максимальної простоти та легкості використання інтерфейсу для всіх користувачів. Ключовим аспектом також було створення послідовної структури, зручної навігації та інтуїтивно зрозумілих елементів управління. Користувачам мала надаватися можливість швидко зрозуміти функціональність програмного забезпечення та виконувати необхідні дії з мінімальними зусиллями. Також було враховано можливість адаптації інтерфейсу для мобільних пристроїв. Дизайн мав бути зручним і функціональним як на ПК, так і на мобільних пристроях. Всі аспекти розробки інтерфейсу ґрунтувалися на сучасних принципах дизайну та враховували потреби кінцевих користувачів.

Також, було важливо забезпечити високий рівень доступності та досяжність для користувачів з різним рівнем навичок у використанні комп'ютерів та Інтернету. Для цього використовувалися яскраві та зрозумілі піктограми, зроблено акцент на інтуїтивному розміщенні елементів керування та використання стандартних конвенцій дизайну веб-інтерфейсів.

Не менш важливим було забезпечити відповідність інтерфейсу принципам безпеки та конфіденційності даних, щоб користувачі відчували себе захищеними під час використання програмного забезпечення.

Крім того, важливим аспектом було створення інтуїтивного інтерфейсу, щоб навіть нові користувачі могли швидко зрозуміти, як використовувати програмне забезпечення без необхідності в додатковій інструкції. Для досягнення цієї мети

було проведено тестування користувацької взаємодії з макетами і прототипами, а також враховано найкращі практики дизайну інтерфейсу.

Нарешті, проєктування інтерфейсу також охоплювало в собі розробку адаптивного дизайну, щоб забезпечити оптимальний вигляд та функціональність на різних пристроях та розмірах екранів, починаючи від настільних комп'ютерів до смартфонів та планшетів, щоб максимально задовольняв потреби користувачів у веб-застосунку "DISCOVERING UA" для подорожей Україною. Перша сторінка веб-інтерфейсу, яку бачить кінцевий користувач відображена на рисунку 2.7.

Рис. 2.7 Головна сторінка веб-застосунка DISCOVERING.UA

У випадку якщо кінцевий користувач відкриває на головній сторінці модальне вікно для реєстрації, яке зображене на рисунку 2.8, сторінка безпосередньо відображає модальну форму вводу всіх необхідних даних для проходження процесу реєстрації.

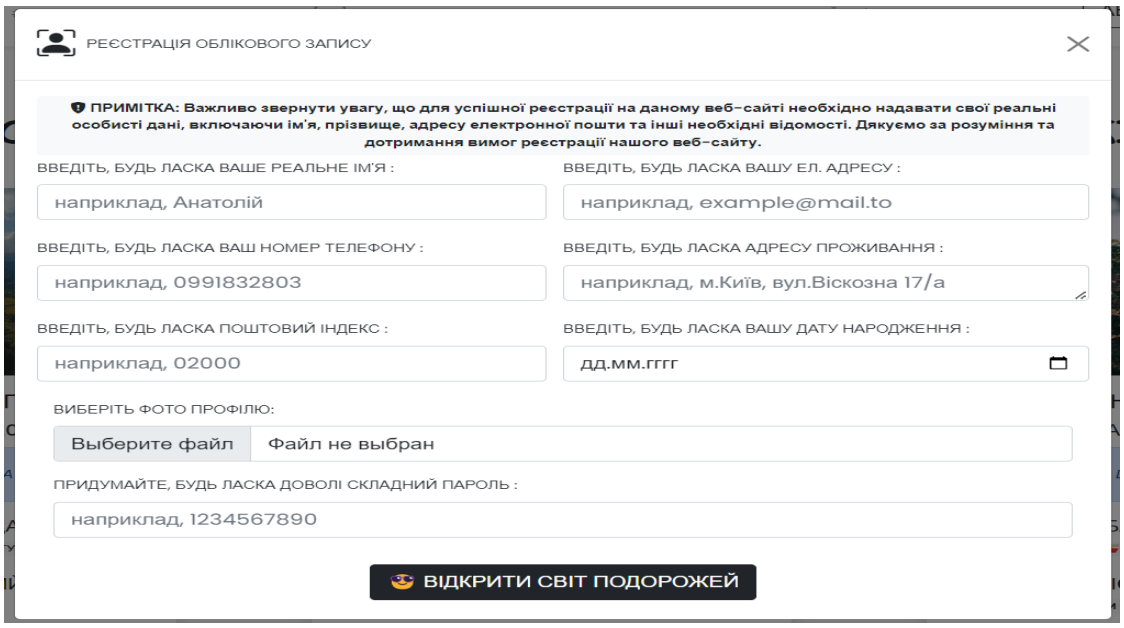


Рис. 2.8 Modal Form реєстрації веб-застосунка DISCOVERING.UA

Результатом проєктування є розроблений інтерфейс кінцевого користувача, який забезпечує зручний та ефективний перегляд каталогу подорожей Україною та управління ним. Цей інтерфейс містить в собі всі необхідні функції та елементи управління, щоб забезпечити користувачеві максимальний комфорт.

3 ОПИС ФУНКЦІОНУВАННЯ ПЛАТФОРМИ ТА ТЕСТУВАННЯ

Даний розділ надає повну характеристику та огляд розроблювальної платформи DISCOVERING.UA та своєю чергою розглядає важливі елементи, які забезпечують зручну навігацію та взаємодію кінцевих користувачів з платформою. В даному розділі також буде міститися опис та результат пройденого тестування платформи.

3.1 Карта веб-застосунка та порядок роботи з нею

Карта веб-застосунку розроблена з метою забезпечити логічний ланцюжок організації та для легшого використання для кінцевого користувача. Дана карта охоплює в собі наступні сторінки, а саме: ГОЛОВНА СТОРІНКА, #WARINUA2022, КАТАЛОГ ТУРІВ, ВІРТУАЛЬНІ ТУРИ, ПРО НАС, КОНТАКТИ, ОБРАНІ ТУРИ, ПРИДБАНІ ТУРИ, ЗАПИТАННЯ / ВІДГУКИ. Кожна розроблена сторінка має своє унікальне наповнення та функціональність, спрямовану на досягнення загальної мети розробки. На рисунку 3.1 представлена карта застосунку.

Рис. 3.1 КАРТА САЙТУ DISCOVERING.UA

Головна сторінка, як в будь-якому веб-проєкті є головною точкою з якої починається знайомство з платформою. Вона загалом містить коротку, але необхідну інформацію про мету проєкту, його цілі, або переваги. На рисунку 3.2 представлена головна сторінка проєкту.

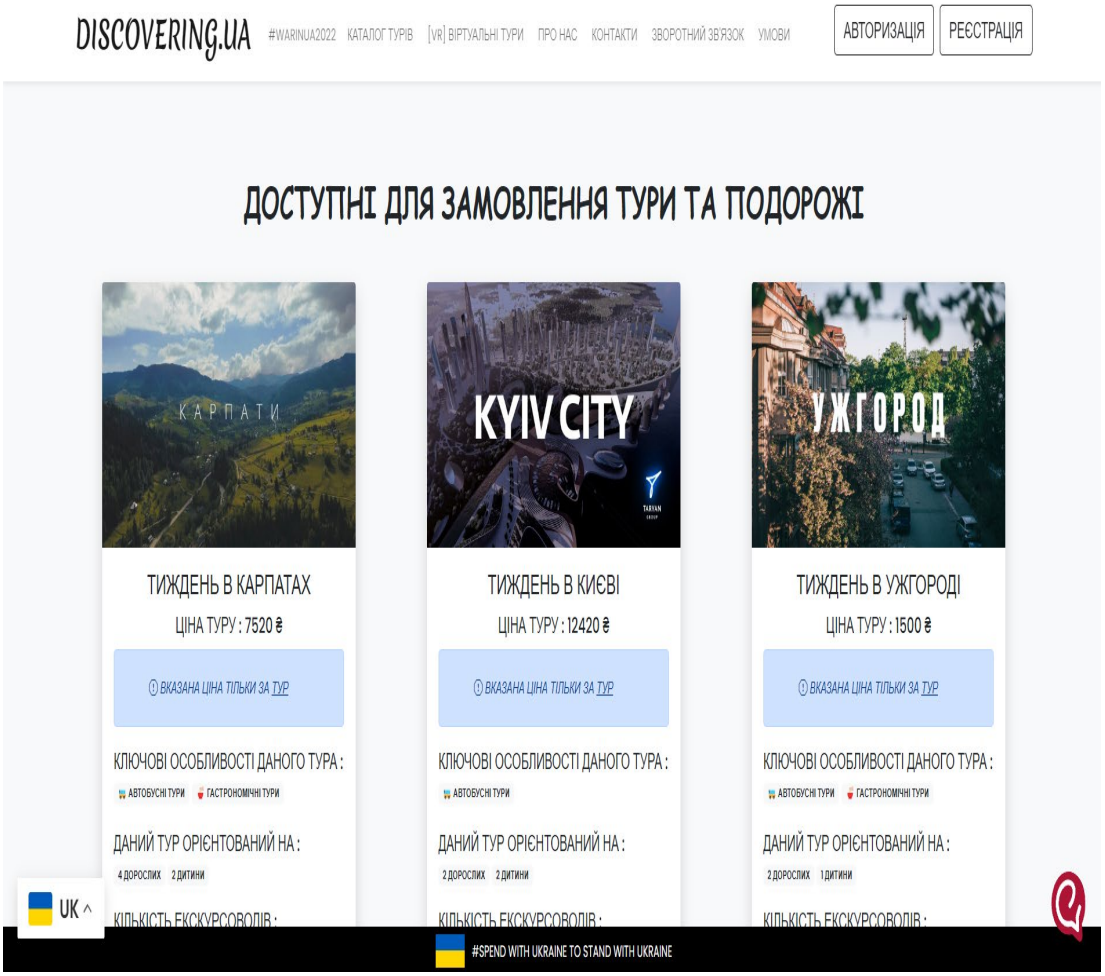


Рис. 3.2 Головна сторінка застосунка DISCOVERING.UA

Сторінка під назвою #WARINUA2022 присвячена для демонстрації тих самих подій, які відбувались в Україні під час початку широко масштабного вторгнення. Так, як платформа надає можливість переглядати деякі віртуальні тури прямо в себе вдома, то було вирішено додати тур по місцях, де проходили визначні події тих самих днів. Користувачі зможуть глибше зануритися в події, що відбувалися, використовуючи інтерактивні матеріали та додаткові ресурси, що допоможуть їм краще зрозуміти подій, які відбувалися в той час. На рисунку 3.3 представлена сторінка платформи #WARINUA2022.

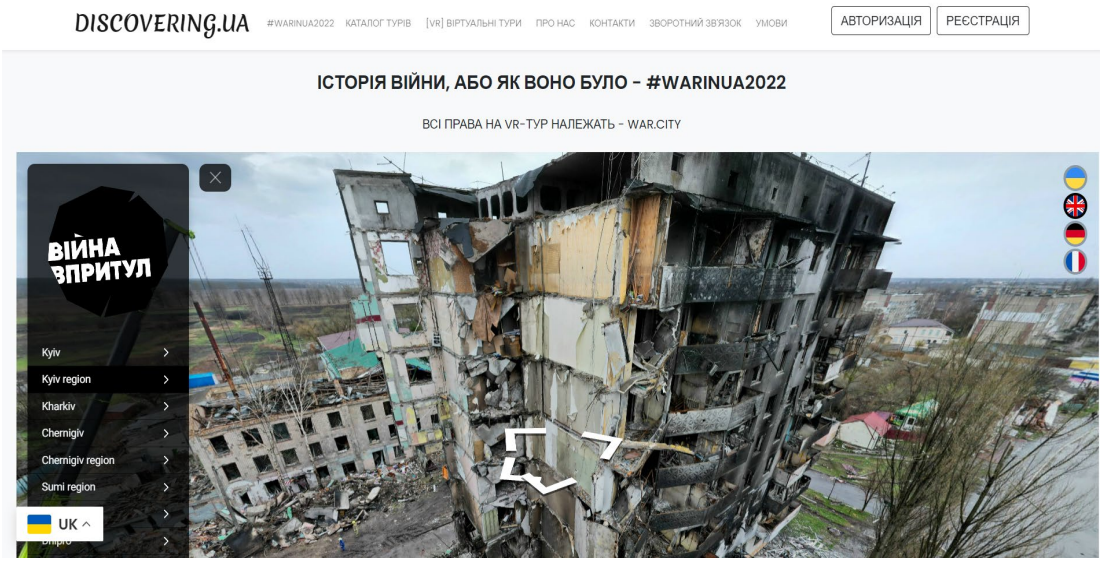


Рис. 3.3 #WARINUA2022 застосунка DISCOVERING.UA

Сторінка під назвою каталог турів, присвячена для загальної демонстрації повного каталогу доступних подорожів та турів для кінцевих користувачів. На даній сторінці, користувач може ознайомитися з каталогом, провести фільтрування турів, наприклад за назвою, або ключовою ознакою, або просто оформити тур. На рисунку 3.4 представлена сторінка каталог турів платформи.

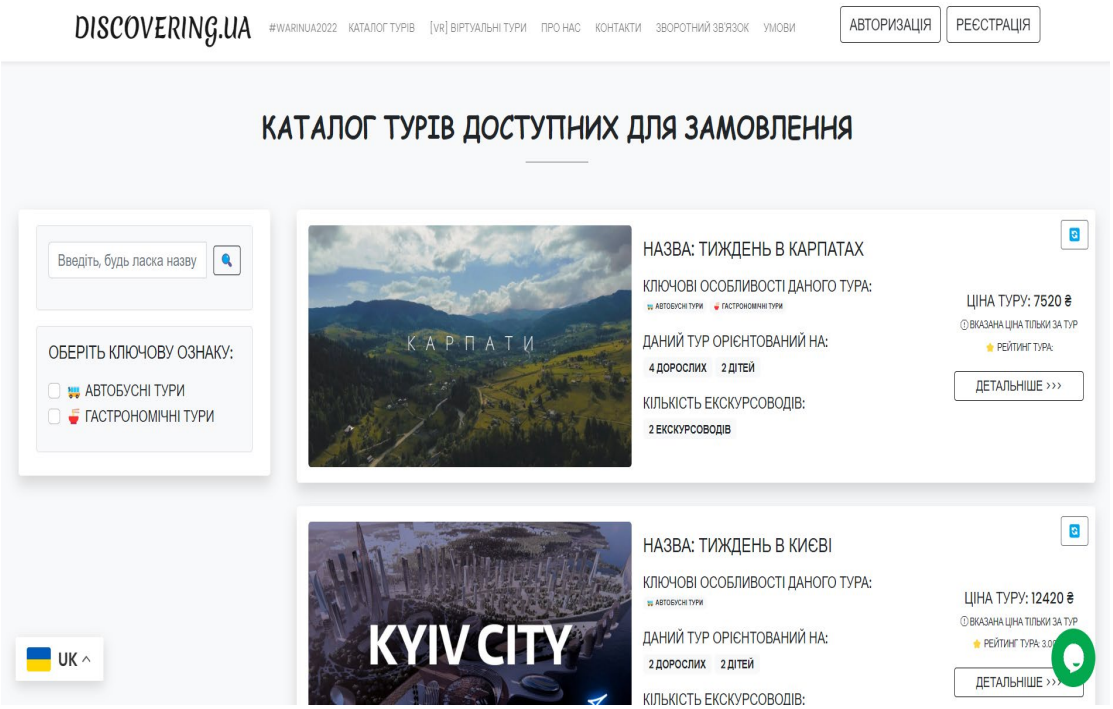


Рис. 3.4 Каталог турів застосунка DISCOVERING.UA

Сторінка під назвою віртуальні тури, являє собою також каталог доступних турів, або подорожей, але без можливості їх оформлення, а лише для перегляду вдома. За допомогою лише однієї кнопки, кінцевий користувач може опинитися на просторах Музею становлення української нації, або Музею видатних діячів. На рисунку 3.5 представлена сторінка каталогу віртуальних турів платформи.

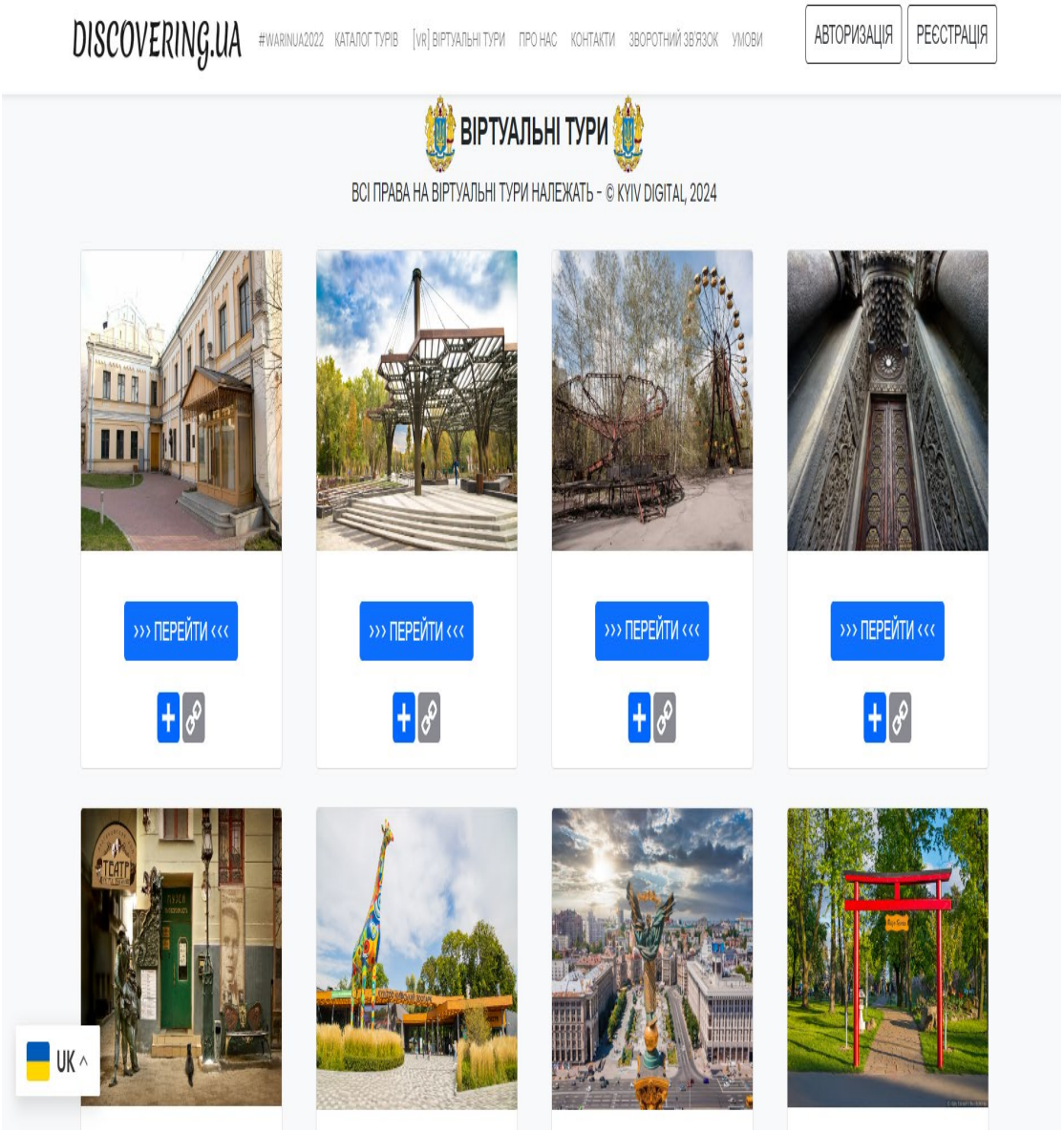


Рис. 3.5 Каталог віртуальних турів застосунка DISCOVERING.UA

Сторінка під назвою про нас, являє собою інформаційний банер, в якому описуються всі ключові аспекти компанії, а також чому саме вони особливі у сфері

туризму. На рисунку 3.6 представлена сторінка про нас платформи.

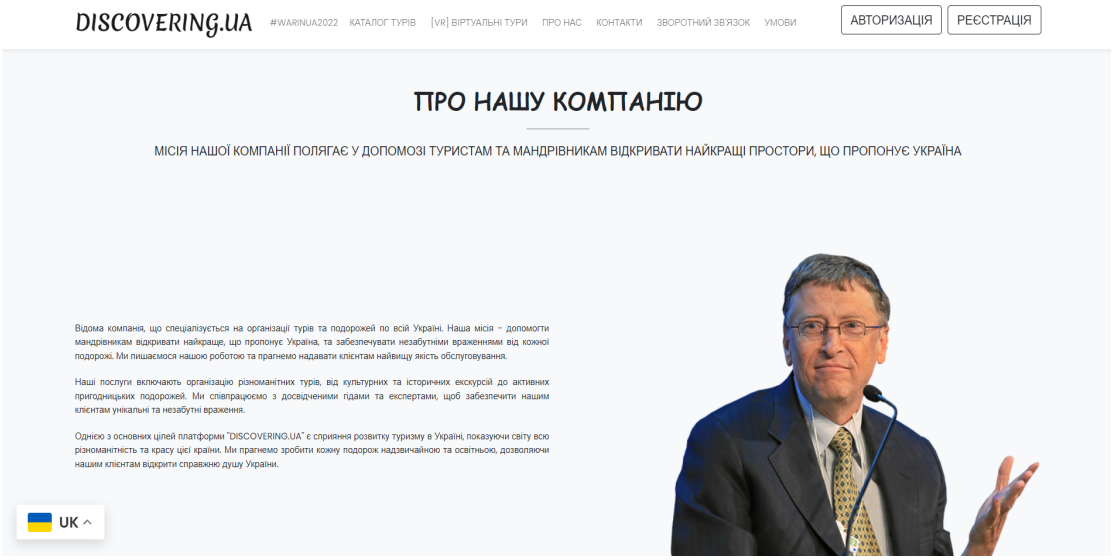


Рис. 3.6 Про нас застосунка DISCOVERING.UA

Сторінка під назвою контакти, являє собою інформаційну сторінку, де користувач зможе дізнатися, де знаходиться, наприклад головний офіс компанії на інтерактивній карті, або ж дізнатися номери телефонів компанії, або її ел. адресу. Також на даній сторінці кінцевий користувач може звернутися до сл. підтримки веб-застосунку, а саме залишити своє ім'я, ел. адресу та вказати, що саме його цікавить. На рисунку 3.7 представлена сторінка контакти платформи.

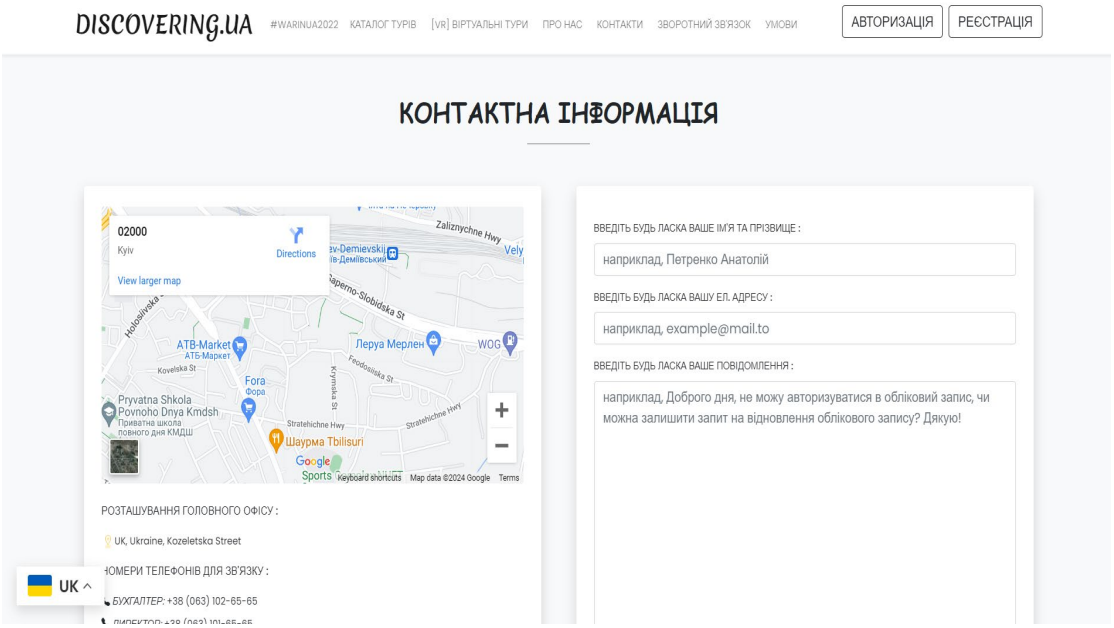


Рис. 3.7 Контакти застосунка DISCOVERING.UA

Сторінка під назвою запитання / відгуки являє собою, так би мовити, сторінку чат, де ще не зареєстровані користувачі платформи можуть обговорити платформу, або якийсь тур в цілому. Для цього необхідно вказати своє ім'я та повідомлення й інші користувачі зможуть оцінювати його, або давати відповідь на нього. На рисунку 3.8 представлена дана сторінка платформи.

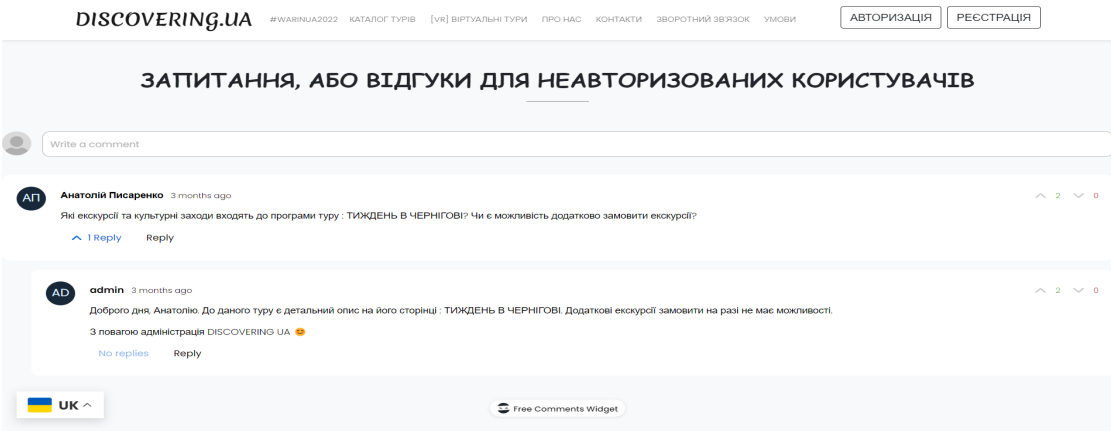


Рис. 3.8 Запитання / відгуки застосунка DISCOVERING.UA

Сторінка під назвою обрані тури, являє собою каталог, де кінцевий користувач зможе без будь-яких проблем додати той чи інший тур в даний список, а надалі, він зможе його оформити й насолодитися ним. На рисунку 3.9 представлена сторінка обрані тури платформи.

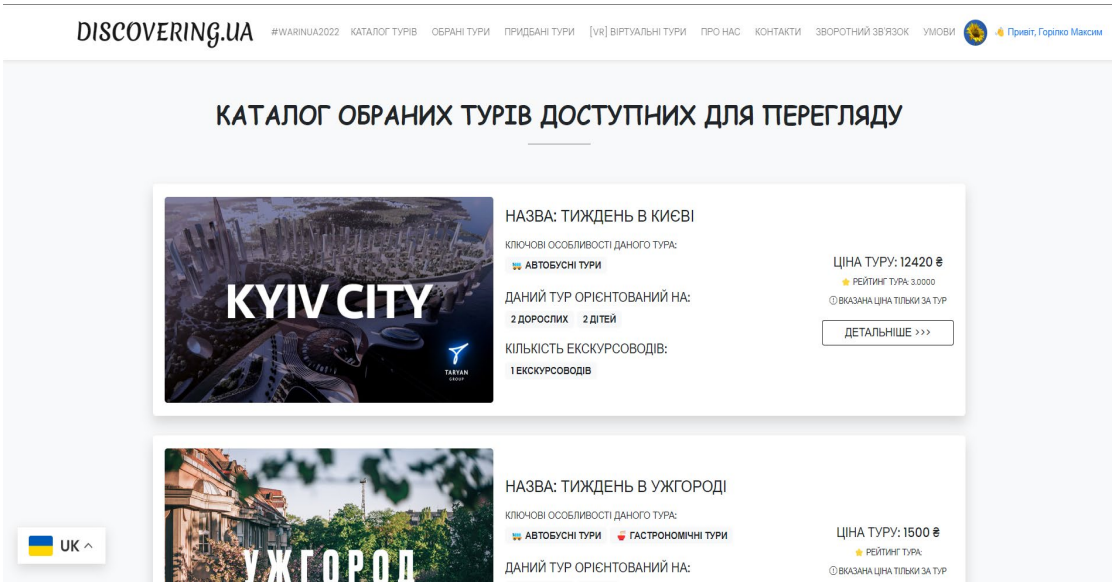


Рис. 3.9 Каталог обраних турів застосунка DISCOVERING.UA

Сторінка під назвою придбані тури, являє собою також каталог, де зберігаються всі оформлені тури та подорожі кінцевим користувачем платформи. Користувачу не потрібно зберігати чек в друкованому вигляді, або на почті, достатньо провести авторизацію, та перейти до каталогу для перегляду придбаних турів. На рисунку 3.10 представлена сторінка придбані тури платформи.

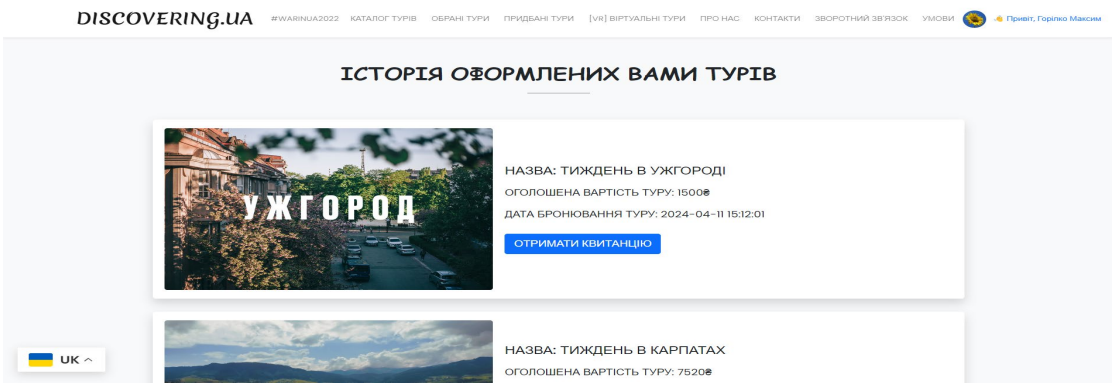


Рис. 3.10 Каталог придбаних турів застосунка DISCOVERING.UA

Сторінка під назвою панель управління являє собою середовище, де адміністратор має можливість змінити назву веб-застосунку, або контактну інформацію. На рисунку 3.11 представлена панель управління платформи.

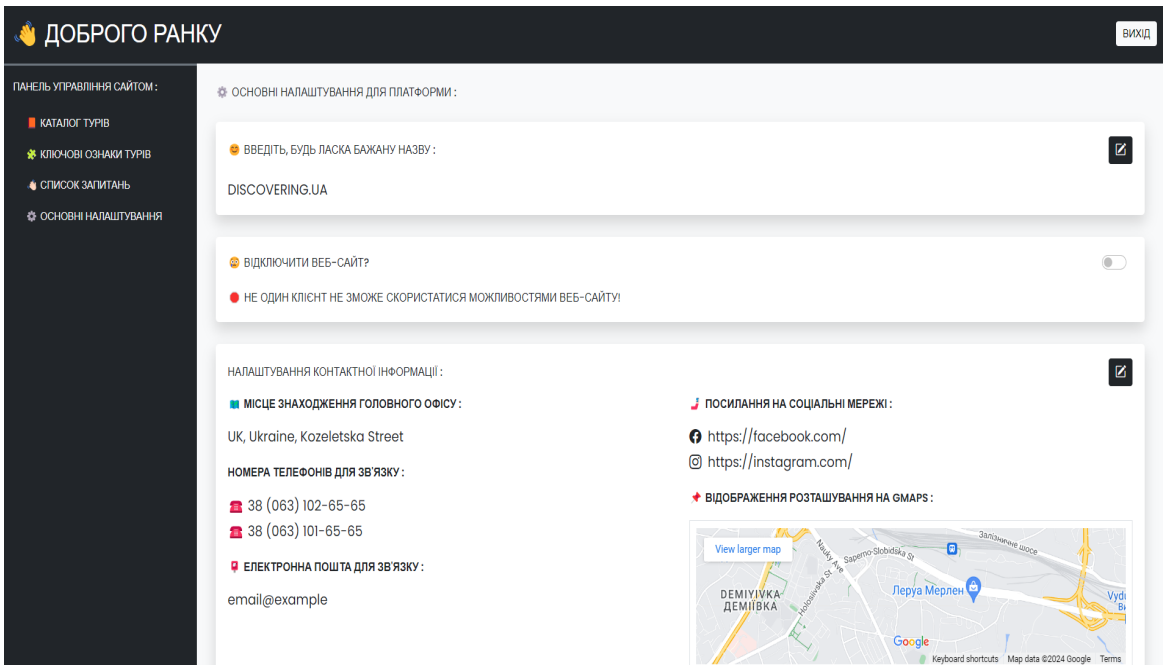


Рис. 3.11 Панель управління застосунком DISCOVERING.UA

Сторінка під назвою каталог турів являє собою сторінку, де адміністратор має можливість переглядати каталог наявних турів на платформі, а також за необхідністю додавати нові. На рисунку 3.12 представлена панель управління платформи.

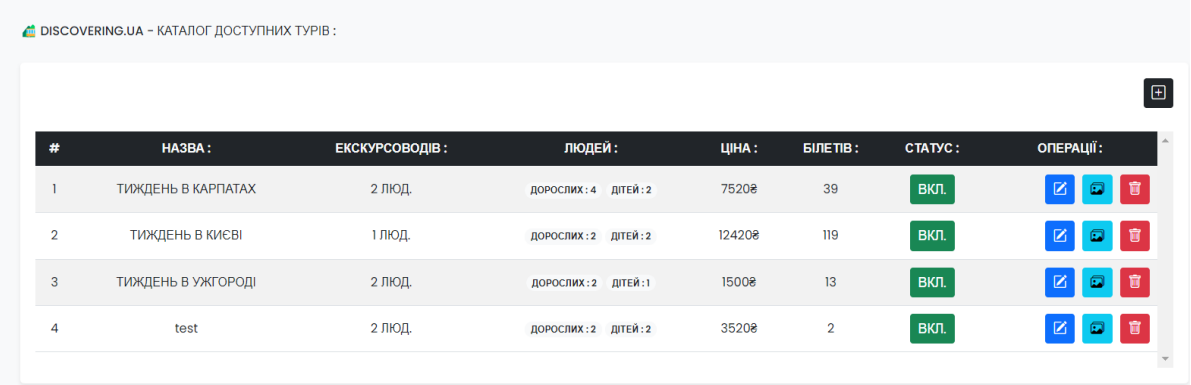


Рис. 3.12 Панель управління застосунком DISCOVERING.UA

Описана структура веб-застосунка дозволяє кінцевим користувачам даної платформи в майбутньому легко знайти потрібну їм інформацію, оформити той чи інший тур, або просто відправити питання про тур адміністрації платформи. Кожна сторінка має не тільки унікальне наповнення, а й функціональність, що своєю чергою спрямовує на досягнення загальної мети розроблювальної платформи.

3.2 Тестування веб-застосунка

Тестування програмного продукту є важливим етапом у процесі його розробки, оскільки воно спрямоване на виявлення помилок, перевірку функціональних можливостей продукту, а також забезпечення його якості на відповідність стандартів. Для веб-застосунків тестування має особливе значення, оскільки вони піддаються впливу широкого кола факторів, включаючи різні конфігурації браузерів та пристроїв кінцевих користувачів. Тестування веб-застосунків також включає оцінку їхньої продуктивності та відповідність вимогам щодо швидкості завантаження та відгуку на взаємодію з користувачем. Для цього проводяться навантажувальні тести, які дозволяють оцінити, як застосунок працює

при великому навантаженні.

Навантажувальні тести містять в собі симуляцію реального навантаження на сервер та мережу, щоб визначити, як швидко веб-застосунок реагує на запити користувачів у різних сценаріях використання. Це допомагає виявити можливі проблеми з продуктивністю та ресурсами, які можуть виникнути під час реального використання.

Одним з ключових аспектів тестування веб-застосунків є перевірка сумісності з різними браузерами та пристроями. Враховуючи різноманітність браузерів (таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо) та пристроїв (комп'ютери, планшети, смартфони), необхідно переконатися, що веб-застосунок працює коректно на будь-якій платформі.

Крім того, тестування веб-застосунків включає аналіз їхньої безпеки. Застосунки повинні бути захищені від потенційних загроз, таких як кібератаки, витік конфіденційної інформації, вразливості в програмному забезпеченні тощо. Тому проведення тестів на проникнення та аудиту безпеки допомагає забезпечити надійність веб-застосунків.

Додатково, тестування веб-застосунків включає перевірку сумісності з різними операційними системами, а також забезпечення коректного відображення та роботи на різних типах пристроїв, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони. Різноманіття пристроїв та їхніх конфігурацій створює складну задачу, яку тестувальні команди повинні вирішувати для забезпечення максимальної доступності та зручності використання.

Основні етапи тестування веб-застосунків включають, а саме:

- функціональне тестування є одним із найважливіших етапів випробування веб-застосунків, оскільки воно спрямоване на перевірку основних функцій програми для забезпечення їх правильної та безперебійної роботи. Першим кроком у функціональному тестуванні є аналіз специфікацій вимог до програмного забезпечення. Це дозволяє тестувальникам розуміти, які саме функції повинен виконувати веб-застосунок та як вони мають працювати. На основі специфікацій вимог розробляються тестові сценарії - послідовності кроків, які потрібно виконати

для перевірки роботи конкретної функції веб-застосунку. Тестувальники виконують тестові сценарії на веб-застосунку. Це може включати введення даних у форми, взаємодію з елементами інтерфейсу, виклик різних функцій та перевірку результатів їх роботи. Після виконання тестів тестувальники аналізують результати та перевіряють, чи працює кожна функція відповідно до очікувань. Вони записують будь-які помилки чи недоліки та повідомляють розробникам для виправлення. Після виправлення виявлених помилок веб-застосунок повторно проходить функціональне тестування, а також проводиться регресійне тестування для перевірки, чи не виникли нові помилки в уже виправлених раніше частинах програми. Усі виявлені проблеми, а також успішно пройдені тестові сценарії, документуються у звіті з тестування. Це допомагає розробникам та іншим учасникам проєкту отримати повну інформацію про стан функціональності веб-застосунку.

- сумісність: впевненість у тому, що веб-застосунок працює на різних браузерах, операційних системах та пристроях кінцевих користувачів. Тестувальники перевіряють, як веб-застосунок працює на різних браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо. Вони переконуються, що всі функції застосунку працюють коректно у кожному з цих браузерів та що він відображається належним чином без жодних відмінностей у вигляді або функціональності. Тестувальники також перевіряють сумісність веб-застосунку з різними операційними системами, такими як Windows, macOS, Linux, Android, iOS тощо. Вони впевнюються, що веб-застосунок працює стабільно та безперебійно на кожній підтримуваній операційній системі. Оскільки веб-застосунки можуть використовуватися на різних пристроях, таких як комп'ютери, ноутбуки, планшети та смартфони, тестувальники перевіряють, як вони працюють на кожному з цих пристроїв. Вони враховують розміри екрана, специфікації пристрою та можливості введення, щоб переконатися, що веб-застосунок відображається та працює коректно на будь-якому пристрої. Особлива увага приділяється адаптивному дизайну, який дозволяє веб-застосунку адаптуватися до різних розмірів екранів та пристроїв. Тестувальники переконуються, що веб-

застосунок правильно адаптується до будь-яких розмірів екрана та забезпечує зручне користування навіть на пристроях з невеликими дисплеями. Під час тестування також важливо перевірити, як веб-застосунок працює при високому навантаженні на сервер та мережу. Тестувальники впевнюються, що він залишається стабільним та продуктивним навіть при інтенсивному використанні користувачами.

- стійкість: тестування стійкості веб-застосунку до можливих негативних сценаріїв, таких як перевантаження сервера, або збій мережі. Під час тестування стійкості веб-застосунку виконуються довготривалі навантажувальні тести, під час яких веб-застосунок випробовується на витривалість та стабільність під час тривалого періоду роботи. Це дозволяє виявити можливі проблеми з витривалістю програми та забезпечити її стійкість протягом тривалого використання. Тестувальники створюють сценарії, які симулюють велике навантаження на сервер веб-застосунку, щоб перевірити, як він реагує на інтенсивне використання. Це дозволяє виявити проблеми з продуктивністю, які можуть виникнути при високому трафіку або великій кількості одночасних запитів. Тестувальники проводять сценарії, під час яких штучно створюються умови для збоїв, наприклад, вимкнення сервера, втрата зв'язку з базою даних або інші технічні проблеми. Це дозволяє перевірити, як веб-застосунок реагує на негативні ситуації та чи може він коректно відновити роботу після збою. Тестувальники проводять експерименти з швидкістю та надійністю мережевого з'єднання для перевірки, як веб-застосунок працює при різних умовах мережі. Вони переконуються, що веб-застосунок може ефективно працювати навіть при обмеженому або нестабільному зв'язку з Інтернетом. Після спровокованих збоїв тестувальники перевіряють, як веб-застосунок відновлюється та чи може він повернутися до нормальної роботи без втрати даних або втрати продуктивності.

- безпека: виявлення та виправлення потенційних порушень безпеки, щоб запобігти можливим вразливостям та кібератакам. Тестувальники аналізують веб-застосунок для виявлення потенційних вразливостей та порушень безпеки. Це може включати перевірку на наявність уразливих точок введення, можливостей

перехоплення даних, атак на сесії, недостатньому контролі доступу та інші. Тестувальники спробують використати виявлені вразливості для здійснення атак на веб-застосунок. Це може бути виконано з метою перехоплення конфіденційної інформації, зміни даних або навіть відмови в обслуговуванні (DoS або DDoS атаки). Цей тип тестування включає спроби проникнути в систему, використовуючи ті ж техніки, що і потенційні зловмисники. Тестувальники використовують різні інструменти та методики для виявлення слабких місць у безпеці веб-застосунку. Після виявлення вразливостей розробники вносять виправлення до коду веб-застосунку для усунення потенційних порушень безпеки. Це може включати патчі, оновлення бібліотек, вдосконалення механізмів аутентифікації та авторизації, імплементацію шифрування даних та інші заходи безпеки. Після внесення виправлень тестувальники перевіряють веб-застосунок на предмет того, чи були ефективно усунуті виявлені вразливості та чи залишилися нові точки доступу для можливих атак. Після випуску веб-застосунку в експлуатацію проводиться постійний моніторинг безпеки для виявлення та усунення нових вразливостей, а також для реагування на потенційні кібератаки.

- швидкодія: вимірювання часу відгуку та завантаження веб-застосунку в цілому для забезпечення оптимальної продуктивності. Час відгуку - це час, необхідний серверу веб-застосунку для обробки запиту від користувача та повернення відповіді. Вимірюється час від відправлення запиту до отримання відповіді. Швидкий час відгуку вказує на ефективну роботу сервера та оптимізовані запити. Час завантаження сторінки - це час, необхідний для повного завантаження всіх ресурсів сторінки, таких як HTML, CSS, JavaScript, зображення тощо. Вимірюється час від відправлення запиту до завершення завантаження всіх елементів. Швидке завантаження сторінки важливо для забезпечення приємного користувацького досвіду та збереження уваги користувачів. Час відгуку окремих запитів важливий для веб-застосунків, які використовують AJAX або взаємодію з API. Вимірюється час від відправлення запиту до отримання відповіді від сервера. Швидкий час відгуку дозволяє забезпечити швидку та плавну взаємодію користувача з веб-застосунком. Проводиться тестування швидкодії веб-застосунку

при різних рівнях навантаження, щоб визначити, як він працює в умовах великого трафіку або інтенсивного використання. Це дозволяє ідентифікувати можливі проблеми з продуктивністю та швидкодією та вчасно реагувати на них. На основі результатів вимірювань веб-застосунок може бути оптимізований для покращення швидкодії. Це може включати усунення зайвих запитів, кешування ресурсів, мінімізацію та об'єднання файлів, використання CDN (Content Delivery Network) та інші техніки оптимізації продуктивності. Після випуску веб-застосунку в експлуатацію важливо постійно моніторити його швидкодію та реагувати на будь-які виникаючі проблеми або погіршення продуктивності.

Загалом в сучасній програмній розробці існує певний ряд підходів до тестування програмного забезпечення, кожен з яких має свої унікальні особливості та застосовується в певних ситуаціях. Відповідно до специфіки розробки веб-застосунків існують різні методи тестування, які дозволяють перевірити якість та надійність їх функціонування. Важливо зазначити, що кожен вид тестування спрямований на виявлення певних аспектів функціональності та може мати відмінності від цілей та потреб майбутнього проєкту.

Основними методами та видами тестування програмного забезпечення можна вважати такі типи, а саме:

- monkey testing (або мавпяче тестування): даний метод полягає у випадковому введенні даних, або виконанні дій у програмі з метою виявлення непередбачених помилок, або помилкових реакцій програми на незвичайні вхідні дані. Даний вид тестування ефективний для виявлення вразливостей та невідомих проблем, але вимагає значної кількості ресурсів.

- black box testing (або тестування "чорної скриньки"): даний метод спрямований на перевірку функціональності програмного забезпечення без знань внутрішньої реалізації. Даний вид тестування розглядає програму, як "чорну скриньку", тобто вхідні дані та очікуваний результат невідомий, через те, що невідомо як програма обробляє введені дані.

- white box testing (або тестування "білої скриньки"): даний метод тестування спрямований на те, що тестувальник вивчає внутрішню структуру програми та

тестує її на основі наданої інформації. Тестувальник перевіряє логіку програми, структуру бази даних, обробку виключень тощо. Даний метод тестування дозволяє ефективно виявляти помилки, пов’язані з програмним кодом, але недоліком є те, що даний метод потребує саме доступу до вихідного коду.

- integration testing (або інтеграційне тестування): даний метод тестування зосереджений на перевірці взаємодії між різними компонентами програмного забезпечення. Під час інтеграційного тестування перевіряється, чи працюють окремі частини програми разом правильно, а також чи взаємодіють вони між собою належним чином.

- regression testing (або регресійне тестування): даний метод тестування використовується для перевірки того, чи впливають зміни в програмі на чинний функціонал. Після внесення змін у програмний код регресійне тестування допомагає впевнитися, що внесені зміни не порушують раніше справної частини програми.

В даній дипломній кваліфікаційній роботі виконувалося тестування веб-застосунку по типу – UAT (USER ACCEPTANCE TESTING) та використовувалася платформа Wrike. UAT (USER ACCEPTANCE TESTING) – даний вид тестування називається як тестування з боку користувача, у ньому група людей, які представляють цільових користувачів, або фінальних користувачів програмного продукту, перевіряє різні аспекти програми з погляду їх очікувань та потреб. Даний вид тестування включає повне тестування, а саме перевірку функціональності, інтерфейсу, взаємодії та загального враження від користування програмним забезпеченням.

Для групового тестування було обрано платформу – Wrike, система управління проєктами та задачами, в тому числі збір команди та підключення до вже відомих послуг виконання спільної роботи, наприклад JIRA. Даний сервіс забезпечує онлайн-середовищем (далі workspace) для робочої взаємодії як у локальній, так і розподіленій команді, дозволяючи планувати проєкти, надавати завдання та відстежувати їх виконання в цілому. На рисунку 3.13 представлена робоча область платформи Wrike, яка містить в собі інтерактивні інструменти для

створення, призначення та відстеження завдань, календарі для планування термінів та визначення дедлайнів, а також інтегровані засоби комунікації для зручної взаємодії між учасниками проєкту.

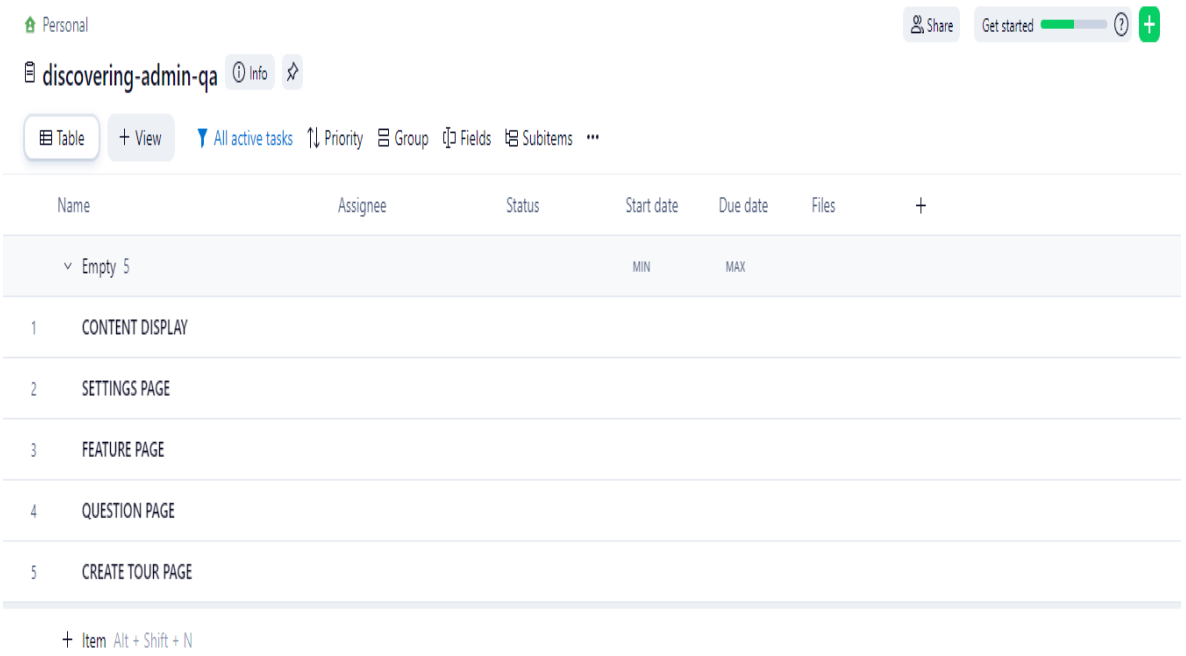


Рис. 3.13 Робоча область платформи Wrike

Платформа Wrike також надає зручні інструменти для моніторингу прогресу проєктів та аналізу результатів. Користувачі можуть створювати звіти та діаграми, що відображають стан виконання завдань, витрати часу, обсяг роботи та інші ключові показники продуктивності. Це дозволяє керівникам проєктів та командам здійснювати швидке прийняття рішень на основі об'єктивних даних.

Однією з переваг платформи Wrike є можливість інтеграції з іншими популярними сервісами та інструментами, такими як Google Drive, Dropbox, Microsoft Office, Slack та інші. Це дозволяє користувачам працювати зі своїми улюбленими інструментами безпосередньо в середовищі Wrike, спрощуючи робочий процес та забезпечуючи безперервну інтеграцію даних та ресурсів.

Крім того, Wrike надає можливість налаштовувати робочі процеси та створювати автоматичні правила для оптимізації робочого потоку. Наприклад, можна налаштувати автоматичне призначення завдань, нагадування про дедлайни

або розподіл робочих ресурсів на основі заданих умов. Це допомагає підвищити ефективність робочого процесу та зменшити ризик помилок або затримок.

У підсумку, платформа Wrike є потужним інструментом для управління проєктами та спільної роботи команд. З її допомогою команди можуть ефективно організовувати свою роботу, взаємодіяти онлайн та досягати поставлених цілей швидко та ефективно. Інтеграція з іншими сервісами та можливість налаштування робочих процесів робить Wrike відмінним вибором для будь-якої команди або організації, що прагне досягти успіху у своїх проєктах.

В даному випадку продемонстровано робочу область, де знаходяться результати тестування веб-застосунку, а саме панелі адміністратора за такими категоріями, як: CREATE TOUR PAGE – сторінка яка відповідає за створення турів та подальше їх відображення на сторінці веб-застосунку, FEATURE PAGE – сторінка, яка відповідає за створення ключових ознак тура (наприклад, гірськолижні тури) та подальше їх відображення в каталозі турів, QUESTIONG PAGE – сторінка, яка відповідає за зворотний зв'язок з кінцевими користувачами веб-застосунку та SETTINGS PAGE – сторінка, яка відповідає за загальні налаштування веб-застосунку в цілому (наприклад, зміна назви веб-застосунку, або зміна контактної інформації). На рисунку 3.14 представлений опис та результати тестування CREATE TOUR PAGE.

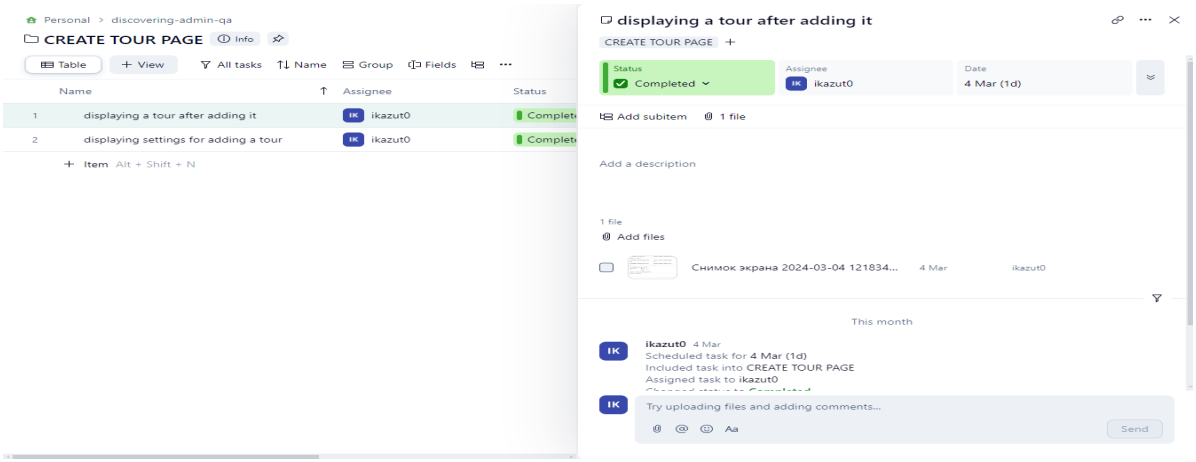


Рис. 3.14 Опис та статус проходження тестування для категорії CREATE TOUR PAGE

Під час тестування категорії - CREATE TOUR PAGE, а саме тестування функціональності створення, редагування та видалення тура. Дана функція веб-застосунку є ключовою, через те, що саме дана функція є базою веб-застосунку в цілому. Протягом тестування перевірялися різні елементи сторінки для створення туру, такі як коректність відображення текстових та графічних матеріалів, належне функціонування кнопок та полів для ведення інформації, а також правильність обробки ведених даних. Також було виконано перевірку взаємодії із системою управління контентом для того, щоб впевнитися, що дані про тур будуть правильно відображатися в каталозі турів для кінцевих користувачів. На даному рисунку тестування категорії - "CREATE TOUR PAGE" має статус завершено, через те, що були проведені всі основні аспекти тестування описані вище. Всі виявленні під час тестування недоліки та пропозиції користувачів, які виконували безпосередньо тестування були враховані та виправлені.

В результаті перевірки були виявленні кілька недоліків та проблем, які вимагали врегулювання та виправлення. Наприклад, були виявленні недоліки в інтерфейсі користувача, та деякі помилки в роботі функції. Завдяки систематичному підходу та уважній роботі вдалося ефективно виправити дані помилки та забезпечити оптимальну продуктивність та функціональність веб-застосунку в цілому.

ВИСНОВКИ

Метою даного проекту було підвищення популярності та привабливості подорожей Україною за рахунок застосування веб-застосунку, який використовує мови програмування HTML, CSS, JavaScript та PHP.

У цьому дослідженні проведено детальний аналіз з метою підтвердження актуальності та наукової новизни даної роботи. Основний акцент був зроблений на оцінку рівня зручності та зацікавленості кінцевого користувача в цілому.

Визначені основні характеристики веб-сервісу для замовлення подорожей по Україні, включаючи: зручний та мінімалістичний дизайн, наявність пошукової системи, детальний опис кожної подорожі у каталозі, систему рейтингу та відгуків, особистий профіль користувача, каталог віртуальних турів, збереження чеків оплати на веб-сайті та підтримку різних мов за допомогою вбудованого віджету.

Було проведено детальний аналіз середовища розробки та інструментів, які використовувалися для створення веб-додатка в цілому. Визначені головні переваги та недоліки кожного з них з метою вибору найзручнішого середовища.

Також був представлений сценарій використання веб-додатку, як користувач буде його використовувати. Після успішного UAT тестування, проведеного за допомогою платформи Wrike, було вирішено, що веб-додаток відповідає всім вимогам та стандартам розробки.

Робота пройшла апробацію на V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 18 квітня 2024 року., Збірник тез. К.: ДУІКТ, 2024. Подано до друку; Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2024 року., Збірник тез. К.: ДУІКТ, 2024. С. 30 та Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 року., Збірник тез. К.: ДУІКТ, 2024. С. 117.

Під час виконання даної кваліфікаційної дипломної роботи був розроблений веб-застосунок для подорожей Україною, який повністю відповідає всім вимогам та стандартам розробки, а також відповідає загальній меті та цілям проекту.

`

ПЕРЕЛІК ПОСИЛАНЬ

1. KRAINA-UA [Електронний ресурс]. – Режим доступу до ресурсу: https://kraina-ua.com/.
2. ETNOSVIT [Електронний ресурс]. – Режим доступу до ресурсу: https://etnosvit.com/.
3. МОЛОДІЖНИЙ ТУРИЗМ [Електронний ресурс]. – Режим доступу до ресурсу: https://mtourism.com.ua/.
4. HTML, ÌÍBÑ. HTML (Hypertext Markup Language). MDN Web Docs [Електронний ресурс], 2022. Режим доступу до ресурсу: https://wiki.gis.com/.
5. SIKOS, Leslie. Web Standards: Mastering HTML5, CSS3, and XML. Apress, 2014.
6. W3C, "HTML5 - Edition for Web Authors", 2014. [Електронний ресурс]. Режим доступу до ресурсу: https://www.w3.org/TR/html5/.
7. W3C, "Cascading Style Sheets (CSS)", [Електронний ресурс]. Режим доступу до ресурсу: https://www.w3.org/Style/CSS/.
8. E. Meyer, "Cascading Style Sheets: The Definitive Guide", 2017.
9. M. Flanagan, "JavaScript: The Definitive Guide", 2020.
10. PHP: Hypertext Preprocessor, [Електронний ресурс]. Режим доступу до ресурсу: https://www.php.net/.
11. Visual Studio Code - Code Editing. Redefined, [Електронний ресурс]. Режим доступу до ресурсу: https://code.visualstudio.com/.
12. SPURLOCK, Jake. Bootstrap: responsive web development. " O'Reilly Media, Inc.", 2013.
13. CHRISTUDAS, Binildas; CHRISTUDAS, Binildas. MySQL. Apress, 2019.
14. DVORSKI, Dalibor D. Installing, configuring, and developing with Xampp. Skills Canada, 2007.
15. BELL, Donald. UML's sequence diagram. IBM.[Online] IBM, 2004, 16.02.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА WEB-ЗАСТОСУНКУ ДЛЯ ПОДОРОЖЕЙ УКРАЇНОЮ "DISCOVERING UA" З ВИКОРИСТАННЯМ HTML, CSS, JS ТА PHP

Виконав студент 4 курсу
групи ПД-44
Горілко Максим Владиславович
Керівник роботи
к. т. н., доц., доцент кафедри ІПЗ Негоденко Олена Василівна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – підвищення популярності та привабливості подорожей Україною за рахунок застосування веб-застосунку створеного мовами HTML, CSS, JavaScript та PHP.

Об’єкт дослідження – процес планування та організації подорожей Україною.

Предмет дослідження – веб-застосунок для планування та організацію подорожей Україною.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз наявних рішень, що забезпечують планування та організацію подорожей Україною.

2. Провести аналіз наявних засобів та середовищ розробки веб-застосунку для подорожей Україною.

3. Розробити функціональні й нефункціональні вимоги до програмного забезпечення для подорожей Україною та спроєктувати модель архітектури.

4. Розробити веб-застосунок для подорожей Україною, використовуючи мови програмування HTML, CSS, JavaScript та PHP, відповідно до визначених вимог.

5. Провести User Acceptance Testing веб-застосунку з використанням платформи Wrike для перевірки відповідності розробленого програмного забезпечення вимогам та очікуванням користувачів.

АНАЛІЗ АНАЛОГІВ

Характеристика	Kraina.UA	EtnoSvit	Молодіжний туризм	discovering.ua
Фільтраційна та пошукова системи	+	-	-	+
Додавання подорожі до списку обраних	-	-	-	+
Наявність системи збереження транзакцій	-	-	-	+
Наявність можливості коментувати та оцінювати подорож	+	+	-	+
Наявність каталогу віртуальних турів	-	-	-	+
Особистий кабінет користувача	+	-	-	+
Підтримка декількох мов	-	-	-	+
Наявність чат-бота	-	-	-	+
Загальна швидкість роботи	0.15 s	0.50 s	0.10 s	0.05 s
Середній час відповіді	0.46 s	0.91 s	0.59 s	0.21 s
Об'єм отриманих даних	0.85 MB	4.04 MB	124.02 MB	267.09 MB

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:

1. Додавати відгуки та оцінки на подорожі.
2. Переглядати каталог подорожів.
3. Переглядати каталог віртуальних подорожів.
4. Авторизація та реєстрація користувачів.
5. Зберігати в веб-застосунку квитанцію про оплату подорожі користувачем.
6. Фільтрувати подорожі за категоріями, які обрав користувач.
7. Додавати подорожі до списку бажаних користувача.

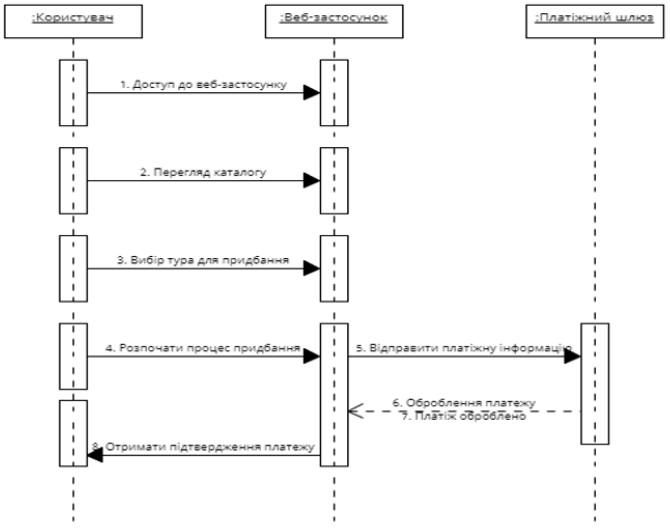
Нефункціональні вимоги:

1. Підтримка різних видів браузерів: Google Chrome, Mozilla Firefox, Microsoft Edge.
2. Забезпечення коректного масштабування на будь-якому пристрої: комп'ютери, телефони, планшети.
3. Шифрувати дані користувачів для захисту конфіденційної інформації.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

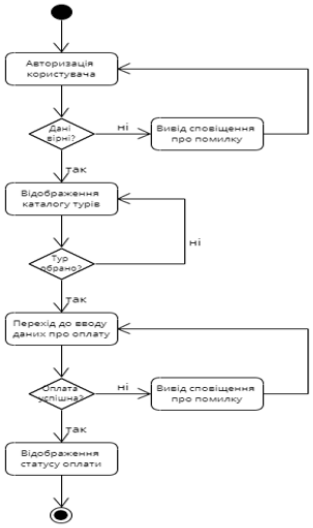
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

ДІАГРАМА ПОСЛІДОВНОСТІ



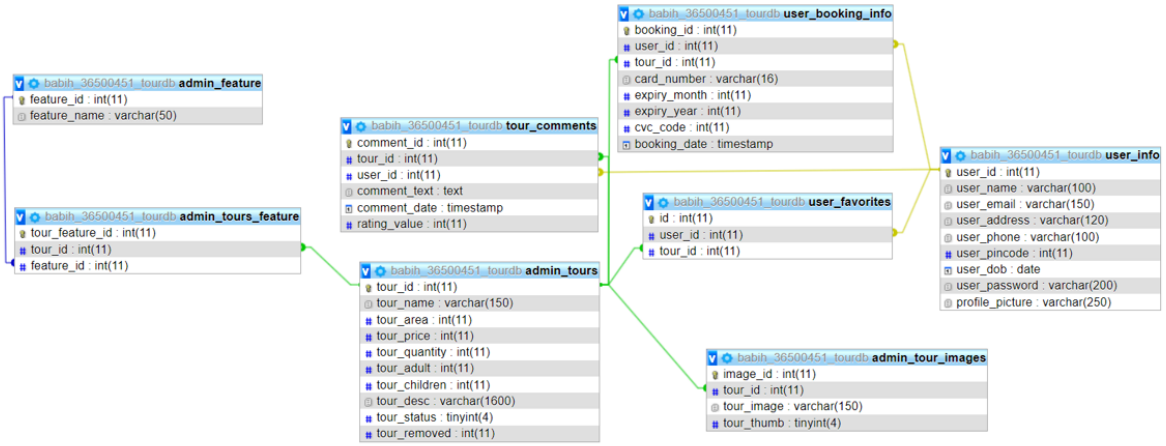
Процес придбання користувачем тура

ДІАГРАМА АКТИВНОСТІ



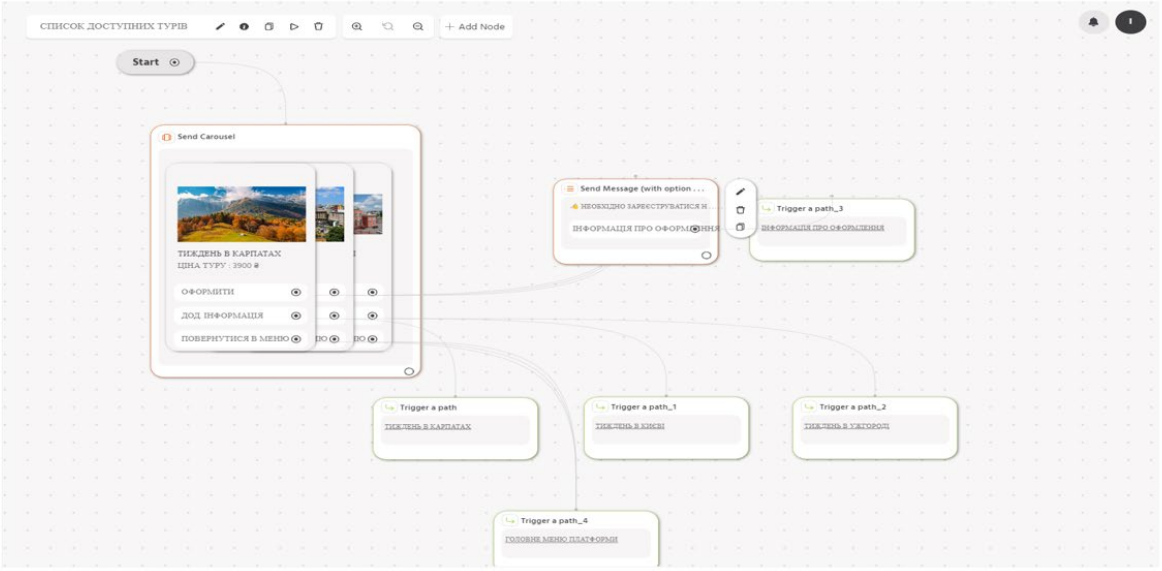
Процес придбання користувачем тура

СХЕМА БАЗИ ДАНИХ



`

СТРУКТУРА ЧАТ-БОТА ENGATI



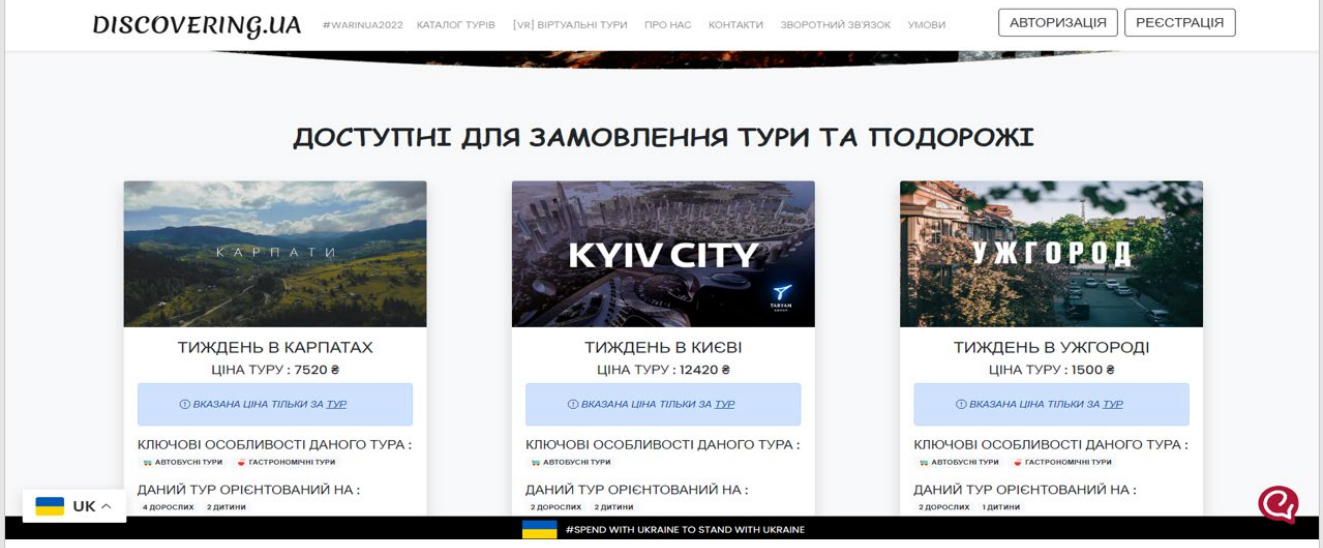
11

КАРТА САЙТУ DISCOVERING.UA

12

`

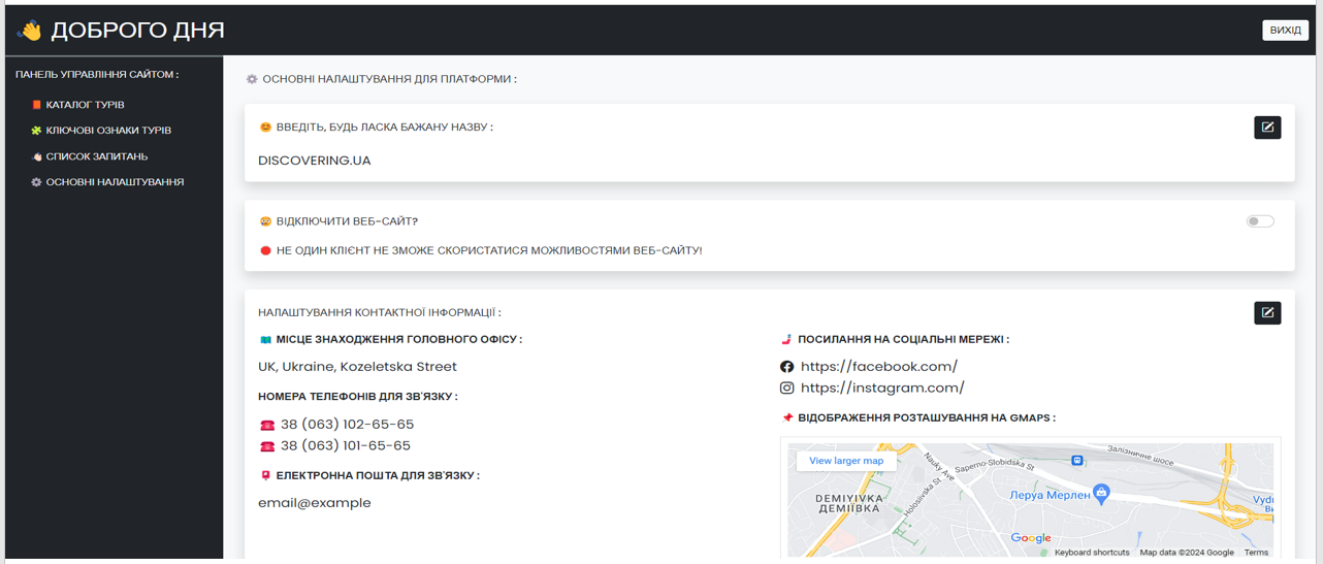
ЕКРАННІ ФОРМИ



Головна сторінка
веб-застосунку

13

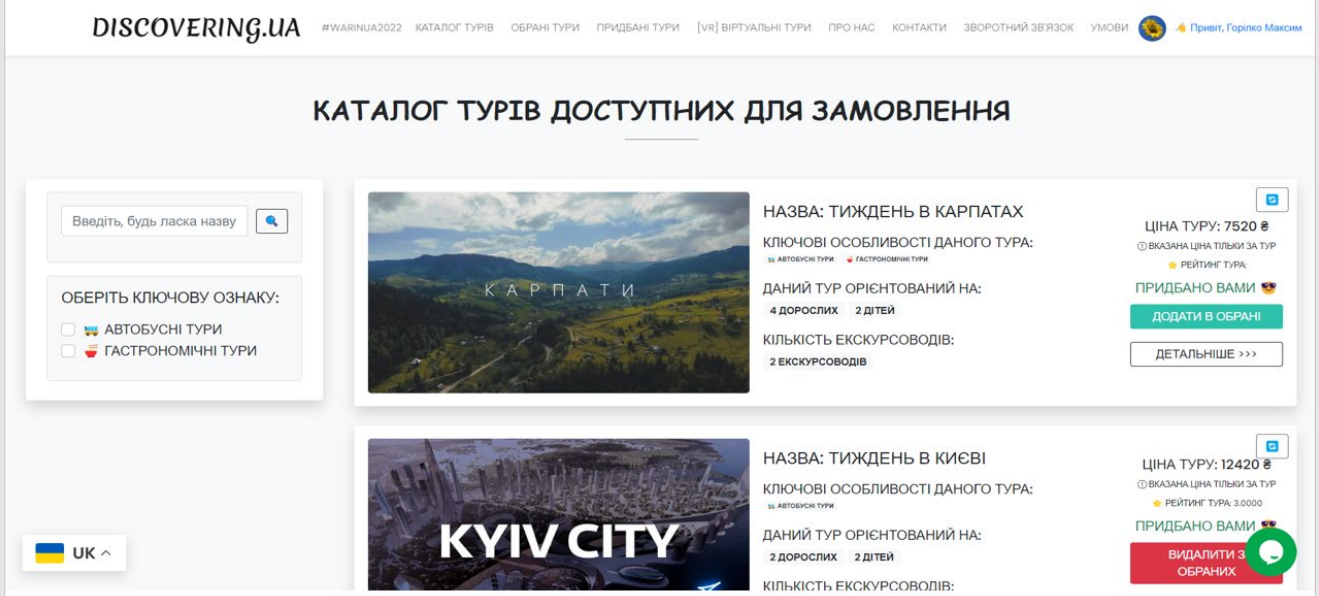
ЕКРАННІ ФОРМИ



Панель управління
веб-застосунку

14

ЕКРАННІ ФОРМИ



Каталог подорожів
веб-застосунку

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Горілко М. В. НАЯВНІ РІШЕННЯ ВЕБ-ДОДАТКІВ ДЛЯ ПОДОРОЖЕЙ УКРАЇНОЮ / Негоденко О. В., Горілко М. В // V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез. 18 квітня 2024 року, ДУІКТ, м. Київ – К: ДУІКТ, 2024. Подано до друку.

2. Горілко М. В. РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ ПОДОРОЖЕЙ УКРАЇНОЮ DISCOVERING UA / Негоденко О. В., Горілко М. В // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24 квітня 2024 року, ДУІКТ, м. Київ – К: ДУІКТ, 2024. С. 30.

3. Горілко М. В. ВИКОРИСТАННЯ ЧАТ-ПЛАТФОРМИ ENGATI В ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОДОРОЖЕЙ УКРАЇНОЮ DISCOVERING UA / Негоденко О. В., Горілко М. В // IV Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15 травня 2024 року, ДУІКТ, м. Київ – К: ДУІКТ, 2024. Подано до друку.

ВИСНОВКИ

1. Проведено аналіз наявних рішень, що забезпечують планування та організацію подорожей Україною.

2. Проведено аналіз наявних засобів та середовищ розробки веб-застосунку для подорожей Україною.

3. Розроблено функціональні й нефункціональні вимоги до програмного забезпечення для подорожей Україною та спроєктовано модель архітектури.

4. Розроблено веб-застосунок для подорожей Україною, використовуючи мови програмування HTML, CSS, JavaScript та PHP, відповідно до визначених вимог.

5. Проведено User Acceptance Testing веб-застосунку з використанням платформи Wrike для перевірки відповідності розробленого програмного забезпечення вимогам та очікуванням користувачів.

`

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО МОДУЛЯ

```
<?php require('admin/include/db_config.php');
require('admin/include/essentials.php'); session_start();

$query = "SELECT `site_shutdown` FROM
`admin_settings` WHERE `settings_id`=?";

$values = [1];

$mysqli = new mysqli("localhost", "root", "", "tourDB");

if ($mysqli->connect_error) {

   die("ПІДКЛЮЧЕННЯ НЕ ВДАЛОСЯ : " . $mysqli-
>connect_error);

}

$stmt = $mysqli->prepare($query);

$stmt->bind_param('i', ...$values);

$stmt->execute();

$result = $stmt->get_result()->fetch_assoc();

$stmt->close();

$mysqli->close();

if ($result && isset($result['site_shutdown']) &&
$result['site_shutdown'] == 1) {

   echo '<div style="display: flex; justify-content: center;
align-items: center; height: 100vh;">';

   echo '<img src="https://assets-
v2.lottiefiles.com/a/6a21fb9a-1178-11ee-a809-
cbf4c1cb708c/9SKB1aSUCp.gif" style="max-width:
100%; max-height: 100%;"/>';

   echo '</div>'; exit;

}

?>

<!DOCTYPE HTML>

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-
width, initial-scale=1.0">

   <link rel="shortcut icon"
href="https://www.freeiconspng.com/uploads/compass-
icon-7.png" type="image/x-icon">

   <title>DISCOVERING - ПОДОРОЖУЙ
КРАЇНОЮ</title>

</head>

<body class="bg-light"><?php
require('include/header.php');?>

   <div class="container-fluid px-lg-4 mt-4">

      <div class="swiper swiper-container">

         <div class="swiper-wrapper">

            <?php $res = selectAll('admin_carousel');
$path = CAROUSEL_IMG_PATH;?>

            <?php while ($row =
mysqli_fetch_assoc($res)):?>

               <div class='swiper-slide'>

                  <img src='<?= $path .
$row['carousel_image']?>' class='w-100 d-block'/>

               </div>

            <?php endwhile; ?>

         </div>

      </div>

   </div>

   <h2 class="mt-5 pt-4 mb-4 text-center fw-bold h-
font">ДОСТУПНІ ДЛЯ ЗАМОВЛЕННЯ ТУРИ ТА
ПОДОРОЖІ</h2>

   <div class="container">

      <div class="row">

         <?php

         $tour_res = selectData("SELECT * FROM
`admin_tours` WHERE `tour_status`=? AND
`tour_removed`=? LIMIT 3", [1, 0], 'ii');

         while ($tour_data =
mysqli_fetch_assoc($tour_res)) {

            $fea_q = selectData("SELECT f.feature_name
FROM `admin_feature` f INNER JOIN
`admin_tours_feature` tfea ON f.feature_id =
tfea.feature_id WHERE tfea.tour_id = ?",
[$tour_data['tour_id']], 'i');

            $features_data = "";

            foreach ($fea_q as $fea_row) {

               $features_data .= "<span class='badge bg-
light text-dark text-wrap me-1 mb-
1'>$fea_row[feature_name]</span>";

            }

            $tour_thumb = TOURS_IMG_PATH .
"empty-image-alert.gif";

            $thumb_q = mysqli_query($connection_info,
"SELECT * FROM `admin_tour_images` WHERE
`tour_id`='$tour_data[tour_id]' AND `tour_thumb`='1'");
```

`

```
            if (mysqli_num_rows($thumb_q) > 0) {

                $thumb_res =
mysqli_fetch_assoc($thumb_q);

                $tour_thumb = TOURS_IMG_PATH .
$thumb_res['tour_image'];

            }

            echo <<< data

            <div class="col-lg-4 col-md-6 my-3">

              <div class='card border-0 shadow'
style='max-width: 350px; margin: auto;'>

                <img src='$tour_thumb' class='card-img-
top' width='350' height='197'>

                <div class="card-body">

                  <h5 class='text-
center'>$tour_data[tour_name]</h5>

                  <h6 class='mb-2 text-center'>ЦІНА
ТУРУ : $tour_data[tour_price] ₴</h6>

                  <div class="alert alert-primary d-flex
align-items-center text-center" role="alert">

                    <svg class="bi flex-shrink-0 me-2"
width="24" height="24" role="img"> <use
xlink:href="#info-fill"/> </svg>

                    <div> <i class="bi bi-exclamation-
octagon"> ВКАЗАНА ЦІНА ТІЛЬКИ ЗА <u> ТУР
</u> </i>

                    </div>

                  </div>

                  <div class="features mb-3">

                    <h6 class="mb-1">КЛЮЧОВІ
ОСОБЛИВОСТІ ДАНОГО ТУРА : </h6>

                    $features_data

                  </div>

                  <div class="peoples mb-3">

                    <h6 class="mb-1">ДАНИЙ ТУР
ОРІЄНТОВАНИЙ НА : </h6>

                    <span class="badge bg-light text-
dark text-wrap">$tour_data[tour_adult]
ДОРОСЛИХ</span>

                    <span class="badge bg-light text-
dark text-wrap">$tour_data[tour_children]
ДИТИНИ</span>

                  </div>

                  <div class="managers mb-3">

                    <h6 class="mb-1">КІЛЬКІСТЬ
ЕКСКУРСОВОДІВ : </h6>

                    <span class="badge bg-light text-
dark text-wrap">$tour_data[tour_area]
ЕКСКУРСОВОД / ГІД [ІВ]</span>

                  </div>

                  <div class="d-flex justify-content-
evenly mb-2">

                    <a href="tour-
details.php?tour_id=$tour_data[tour_id]" class="btn btn-
sm w-100 btn-outline-dark shadow-none mt-2 ml-2">
ДЕТАЛЬНІШЕ >>> </a>

                  </div>

                </div>

              </div>

            </div>

            data;

          }

          ?>

      </div>

    </div>

    <div class="gtranslate_wrapper"></div>

    <script>window.gtranslateSettings =
{"default_language":"uk","native_language_names":true
,"detect_browser_language":true,"languages":["uk","de",
"en"],"wrapper_selector":".gtranslate_wrapper","float_s
witcher_open_direction":"bottom"}</script>

    <script
src="https://cdn.gtranslate.net/widgets/latest/float.js"
defer></script>

    <div class="container">

      <footer class="d-flex flex-wrap justify-content-
between align-items-center py-3 my-4 border-top">

        <p class="col-md-4 mb-0 text-muted">© 2024
DISCOVERING.UA, INC</p>

        <a href="#" class="col-md-4 d-flex align-items-
center justify-content-center mb-3 mb-md-0 me-md-auto
link-dark text-decoration-none"><img class="bi me-2"
width="70" height="32"
src="https://pngimg.com/d/ukraine_stand_PNG28.png">
</a>

      </footer>

    </div>

    <a class="support-ukraine" target="_blank"
rel="nofollow noopener">

      <div class="support-ukraine__flag" role="img">
```

```
`

        <div class="support-ukraine__flag__blue"></div>

        <div class="support-ukraine__flag__yellow"></div>

    </div>

    <div class="support-ukraine__label">#SPEND WITH UKRAINE TO STAND WITH UKRAINE</div>

  </a>

  <script>!function(e,t,a){var c=e.head||e.getElementsByTagName("head")[0],n=e.createElement("script");n.async=!0,n.defer=!0, n.type="text/javascript",n.src=t+"/static/js/widget.js?config="+JSON.stringify(a),c.appendChild(n)}(document,"https://app.engati.com",{bot_key:"0aa32f6222c342f2",welcome_msg:true,branding_key:"default",server:"https://app.engati.com",e:"p" });</script>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

  <script src="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.js"></script>

  <script src="scripts/swiper-slider.js"></script>

 </body>

</html>

<?php require('admin/include/db_config.php'); require('admin/include/essentials.php'); session_start();

$query = "SELECT `site_shutdown` FROM `admin_settings` WHERE `settings_id`=?";

$values = [1];

$mysqli = new mysqli("localhost", "root", "", "tourDB");

if ($mysqli->connect_error) {

  die("ПІДКЛЮЧЕННЯ НЕ ВДАЛОСЯ : " . $mysqli->connect_error);

}

$stmt = $mysqli->prepare($query);

$stmt->bind_param('i', ...$values);

$stmt->execute();

$result = $stmt->get_result()->fetch_assoc();

$stmt->close();

$mysqli->close();

if ($result && isset($result['site_shutdown']) && $result['site_shutdown'] == 1) {

  echo '<div style="display: flex; justify-content: center; align-items: center; height: 100vh;">';

  echo '<img src="https://assets-v2.lottiefiles.com/a/6a21fb9a-1178-11ee-a809-cbf4c1cb708c/9SKB1aSUCp.gif" style="max-width: 100%; max-height: 100%;"/>';

  echo '</div>'; exit;

}

<!DOCTYPE HTML>

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <link rel="shortcut icon" href="https://www.freeiconspng.com/uploads/compass-icon-7.png" type="image/x-icon">

  <style>.a2a_kit { display: flex; justify-content: center; margin-top: 20px; }</style>

  <title>DISCOVERING - ПОДОРОЖУЙ КРАЇНОЮ</title>

</head>

<body class="bg-light"><?php require 'include/header.php';?>

  <div style="text-align: center; font-size: 24px; font-weight: bold; margin: 0 20px;"><img width="45" height="50" src="images/online-tour-pictures/online-tour-picture-ico.png"> ВІРТУАЛЬНІ ТУРИ <img width="45" height="50" src="images/online-tour-pictures/online-tour-picture-ico.png"></div>

  <p style="text-align: center; font-size: 16px; margin: 0 20px;">ВСІ ПРАВА НА ВІРТУАЛЬНІ ТУРИ НАЛЕЖАТЬ - © KYIV DIGITAL, 2024</p><br>

  <div id="street-view-container1" style="width: 100vw; height: 100vh; display: none;">

    <iframe id="pano-item1" width="100%" height="100%"></iframe>

  </div>

  <div id="street-view-container2" style="width: 100vw; height: 100vh; display: none;">

    <iframe id="pano-item2" width="100%" height="100%"></iframe>

  </div>

  <div id="street-view-container3" style="width: 100vw; height: 100vh; display: none;">

    <iframe id="pano-item3" width="100%" height="100%"></iframe>

  </div>
```

```
<div id="street-view-container4" style="width: 100vw; height: 100vh; display: none;">
    <iframe id="pano-item4" width="100%" height="100%"></iframe>
</div>
<div id="street-view-container5" style="width: 100vw; height: 100vh; display: none;">
    <iframe id="pano-item5" width="100%" height="100%"></iframe>
</div>
<div id="street-view-container6" style="width: 100vw; height: 100vh; display: none;">
    <iframe id="pano-item6" width="100%" height="100%"></iframe>
</div>
<div id="street-view-container7" style="width: 100vw; height: 100vh; display: none;">
    <iframe id="pano-item7" width="100%" height="100%"></iframe>
</div>
<div id="street-view-container8" style="width: 100vw; height: 100vh; display: none;">
    <iframe id="pano-item8" width="100%" height="100%"></iframe>
</div>
<div class="container">
    <div class="row">
        <?php
        $tours = [
            ["1.jpg", "", "", "pano-item1"],
            ["2.png", "", "", "pano-item2"],
            ["3.jpg", "", "", "pano-item3"],
            ["4.jpg", "", "", "pano-item4"],
            ["5.jpg", "", "", "pano-item5"],
            ["6.jpg", "", "", "pano-item6"],
            ["7.jpg", "", "", "pano-item7"],
            ["8.jpg", "", "", "pano-item8"],
        ];
        foreach ($tours as $tour) {
            echo '<div class="col-lg-3 col-md-4 col-sm-6 mb-4" id="tour-' . $tour[3] . '">';
            echo '    <div class="card mx-auto" style="width: 18rem;">';
            echo '        <img src="images/online-tour-pictures/online-tour-picture-' . $tour[0] . '" class="card-img-top">';
            echo '        <div class="card-body">';
            echo '            <h5 class="card-title text-center">' . $tour[1] . '</h5>';
            echo '            <p class="card-text text-center">' . $tour[2] . '</p>';
            echo '            <button class="btn btn-primary d-block mx-auto" onclick="showStreetView(\'' . $tour[3] . '\')"> >>> ПЕРЕЙТИ <<< </button>';
            echo '            <div class="a2a_kit a2a_kit_size_32 a2a_default_style">';
            echo '                <a class="a2a_dd" href="https://www.addtoany.com/share"></a>';
            echo '                <a class="a2a_button_copy_link"></a>';
            echo '            </div>';
            echo '        </div>';
            echo '    </div>';
            echo '</div>';
        }
        ?>
        </div>
    </div>
</div>
<div class="gtranslate_wrapper"></div>
<script>window.gtranslateSettings = {"default_language":"uk","native_language_names":true,"detect_browser_language":true,"languages":["uk","de","en"],"wrapper_selector":".gtranslate_wrapper","float_switcher_open_direction":"bottom"}</script>
<script src="https://cdn.gtranslate.net/widgets/latest/float.js" defer></script>
<div class="container">
    <footer class="d-flex flex-wrap justify-content-between align-items-center py-3 my-4 border-top">
        <p class="col-md-4 mb-0 text-muted">© 2024 DISCOVERING.UA, INC</p>
        <a href="#" class="col-md-4 d-flex align-items-center justify-content-center mb-3 mb-md-0 me-md-auto link-dark text-decoration-none"><img class="bi me-2" width="70" height="32" src="https://pngimg.com/d/ukraine_stand_PNG28.png">
```

```
`

</a>

    </footer>

  </div>

  <script> var a2a_config = a2a_config || {};
a2a_config.onclick = 1; a2a_config.locale = "uk";
</script>

  <script async
src="https://static.addtoany.com/menu/page.js"></script
>

  <script>

    document.addEventListener('DOMContentLoaded',
function () {

    document.documentElement.requestFullscreen();

  });

  function showStreetView(panoItemId) {

    var containers =
document.querySelectorAll('[id^="street-view-
container"]');

    containers.forEach(function (container) {

      container.style.display = 'none';

    });

    var streetViewContainer =
document.getElementById('street-view-container' +
panoItemId.substr(-1));

    streetViewContainer.style.display = 'block';

    if (streetViewContainer.requestFullscreen) {

      streetViewContainer.requestFullscreen();

    } else if
(streetViewContainer.mozRequestFullScreen) {

      streetViewContainer.mozRequestFullScreen();

    } else if
(streetViewContainer.webkitRequestFullscreen) {

      streetViewContainer.webkitRequestFullscreen();

    } else if
(streetViewContainer.msRequestFullscreen) {

      streetViewContainer.msRequestFullscreen();

    }

    var panoItem = document.getElementById('pano-
item' + panoItemId.substr(-1));

    switch (panoItemId) {

      case 'pano-item1':

        panoItem.src =
'https://guide.kyivcity.gov.ua/muzey-diachiv/'; break;

      case 'pano-item2':

        panoItem.src =
'https://guide.kyivcity.gov.ua/ecotrail/'; break;



      case 'pano-item3':

        panoItem.src =
'https://tourmkr.com/F1iNjhtnmp/41057292p&356h&98.
78t'; break;

      case 'pano-item4':

        panoItem.src =
'https://tourmkr.com/F1qAhVy7kQ/38819197p&213.95h
&43.54t'; break;

      case 'pano-item5':

        panoItem.src =
'https://www.google.com/maps/embed?pb=!4v16398388
05952!6m8!1m7!1sCAoSLEFGMVFpcE1SU1VvOFVH
djhJcExwck1iYU1LVWNsWV91eUh2bkdLZ3dqd1Iz!2
m2!1d50.460700767123!2d30.514618830242!3f219.09!
4f-21.069999999999993!5f0.7820865974627469';
break;

      case 'pano-item6':

        panoItem.src =
'https://www.google.com/maps/embed?pb=!4v16398389
74727!6m8!1m7!1sCAoSLEFGMVFpcE5RX2lvRHBT
ZFRuWVlXaDRTQ3BZSnl2S19DNzc1QUJnUURyNF
Va!2m2!1d50.456109189169!2d30.462717396269!3f18
0.92661826250787!4f-
8.637894345657458!5f0.4000000000000002'; break;

      case 'pano-item7':

        panoItem.src =
'https://tourmkr.com/F1GJRN7k7h/38768995p&342.8h
&90t'; break;

      case 'pano-item8':

        panoItem.src =
'https://www.google.com/maps/embed?pb=!4v16400179
07560!6m8!1m7!1sCAoSLEFGMVFpcE9fRk45V1lKbk
JyT3pRMDkyb09OTXV4MTRsWU96WTc1LVJ0eDdI!
2m2!1d50.464325397393!2d30.63926390774!3f246!4f0
!5f0.7820865974627469'; break;

      default:

        panoItem.src = 'https://default-pano-url.com';

      }

    }
```