МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ ВИЗНАЧНИХ КУЛЬТУРНИХ ПАМ'ЯТОК ЄВРОПИ»**

Виконала: студентка 2 курсу, групи 8.1213-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
С.С. Абдуллаєва
(ініціали та прізвище)
Керівник завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Абдуллаєвій Севді Самандар Кизи
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційно-пошукової системи визначних культурних пам’яток Європи

керівник роботи Гребенюк Сергій Миколайович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз і визначення вимог.
2. Проєктування системи.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.03.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	25.03.2024	
2.	Збір вихідних даних.	17.04.2024	
3.	Обробка методичних та теоретичних	11.06.2024	
	джерел.		
4.	Розробка першого та другого розділу.	21.08.2024	
5.	Розробка третього розділу.	05.11.2024	
6.	Оформлення та нормоконтроль	20.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	11.12.2024	

Студент _____ С.С. Абдуллаєва
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.М. Гребенюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційно-пошукової системи визначних культурних пам’яток Європи»: 71 с., 28 рис., 4 табл., 24 джерела, 1 додаток.

ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА, CSS, EXPRESS, JAVA SCRIPT, HTML, MONGODB.

Об’єкт дослідження – процес розробки інформаційно-пошукової системи, засобами Express.js.

Предмет дослідження – фреймворк Express, бібліотеки для роботи з ним, засоби для створення та налаштування маршрутів системи.

Мета роботи: розробка інформаційної системи для перегляду, пошуку, публікації історичних пам’яток Європи, з можливістю додавання нових об’єктів культурної спадщини.

Метод дослідження – методи збору та аналізу вимог до інформаційно-пошукової системи, метод порівняння існуючих систем, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

При розробці інформаційно-пошукової системи було проведено аналіз предметної області, розроблена загальна схема роботи системи, функціональна модель, діаграма варіантів використання інформаційно-пошукової системи, спроєктована інформаційна модель системи. Середовищем розробки є Visual Studio Code. Мови програмування, використані для розробки, включали JavaScript для фронтенду, HTML для структурування контенту та CSS для стилізації. Крім того, база даних MongoDB використовувалась для забезпечення надійного та масштабованого зберігання даних. В результаті роботи отримано інформаційно-пошукову систему для перегляду, пошуку, публікації історичних пам’яток Європи, з можливістю додавання нових об’єктів культурної спадщини.

SUMMARY

Master’s qualifying paper “Development of an Information and Search System of Notable Cultural Monuments in Europe”: 71 pages, 28 figures, 4 tables, 24 references, 1 supplement.

INFORMATION SEARCH SYSTEM, CSS, EXPRESS, JAVA SCRIPT, HTML, MONGODB.

The object of research is the process of developing an information and search system using Express.js.

The subject of research is the Express framework, libraries for working with it, tools for creating and configuring system routes.

The purpose of the work: development of an information system for viewing, searching, and publishing historical monuments of Europe, with the possibility of adding new objects of cultural heritage.

Research method – methods of collecting and analyzing requirements for the information search system, a method of comparing existing systems, methods of modeling, designing, designing and testing software.

During the development of the information and search system, an analysis of the subject area was carried out, a general scheme of the system’s operation, a functional model, a diagram of options for using the information and search system, and a projected information model of the system were developed. The development environment is Visual Studio Code. The programming languages used for development included JavaScript for the front end, HTML for structuring the content, and CSS for styling. In addition, the MongoDB database was used to provide reliable and scalable data storage. As a result of the work, an information and search system was obtained for viewing, searching, and publishing historical monuments of Europe, with the possibility of adding new objects of cultural heritage.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Скорочення та умовні познаки 8
Вступ..... 9
1 Технічне завдання 11
1.1 Терміни та визначення..... 11
1.2 Функціональні вимоги..... 12
1.3 Нефункціональні вимоги..... 13
1.3.1 Інтерфейс користувача 13
1.3.2 Підтримка браузерів 14
1.3.3 Вимоги до продуктивності..... 14
1.3.4 Вимоги до безпеки 15
1.4 Опис предметної області 15
1.5 Опис інформаційно-пошукової системи 16
1.6 Програми аналоги 18
2 Проєктування інформаційно-пошкової системи 22
2.1 Використання UML під час розробки системи..... 22
2.2 Загальна схема роботи системи..... 23
2.3 Функціональна модель ІПС 25
2.4 Інформаційна модель ІПС..... 28
2.5 Діаграма варіантів використання 32
3 Реалізація та тестування 34
3.1 Опис інструментів розробки..... 34
3.2 Процес створення сервера..... 36
3.3 Процес створення та налаштування MongoDB..... 37
3.4 Архітектура інформаційно-пошукової системи 39

3.5 Тест-кейси 48
3.6 Тестування та аналіз результатів тестування 56
Висновки …. 60
Перелік посилань 62
Додаток А Програмна реалізація 65

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	База даних
ІПС	Інформаційно-пошукова система
ІПС	Операційна система
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
EJS	Embedded JavaScript
HTML	HyperText Markup Language
IDEF0	Integration DEFinition for Function Modeling
IDEF1X	Integration DEFinition for Information Modeling
NFRs	Non-functional requirements
ORM	Object Relational Mapper
UML	Unified Modeling Language
URI	Uniform Resource Identifier

ВСТУП

У сучасному світі інформаційно-пошукові системи стали невід'ємною частиною нашого життя, забезпечуючи швидкий доступ до великого обсягу даних та полегшуючи пошук необхідної інформації. Сьогодні, коли обсяги інформації стрімко зростають, ефективні інструменти для її обробки та пошуку є надзвичайно важливими. Інформаційно-пошукові системи дозволяють користувачам швидко знаходити потрібні дані, що значно підвищує продуктивність та ефективність роботи в різних галузях. Розробка ІПС вимагає використання ефективних та надійних інструментів, які забезпечують високу продуктивність, масштабованість та швидкість. JavaScript та фреймворк Express.js є одними з найпопулярніших технологій, що використовуються для створення таких систем, а прикладом буде виступати ІПС визначних культурних пам'яток Європи «CultureQuest».

Одним із яскравих прикладів актуальності інформаційно-пошукових систем є розробка спеціалізованих систем для культурної спадщини. Такі системи дозволяють зберігати, каталогізувати та знаходити інформацію про визначні культурні пам'ятки, що є надзвичайно важливим для їхнього збереження та популяризації. Таким чином, ІПС не лише полегшують доступ до інформації, але й сприяють збереженню культурної спадщини.

Актуальність інструментів, які були використані в процесі розробки, а саме JavaScript та фреймворку Express.js, зумовлена тим, що вони забезпечують швидкість, продуктивність, гнучкість та легкість у розробці, що робить їх ідеальним вибором для створення ефективних і масштабованих інформаційних систем.

Метою роботи є розробка ІПС для перегляду, пошуку, публікації історичних пам'яток Європи, з можливістю додавання нових об'єктів культурної спадщини.

Об'єкт дослідження – процес розробки інформаційно-пошукової системи, засобами Express.js.

Предмет дослідження – фреймворк Express, бібліотеки для роботи з ним, засоби для створення та налаштування маршрутів системи.

Метод дослідження – методи збору та аналізу, порівняння, моделювання, проєктування, конструювання та тестування програмного забезпечення.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

БД – організована колекція даних, яка зберігається і управляється за допомогою системи управління базами даних (СУБД).

Користувач – людина, яка заінтересована у пошуку визначних культурних пам’яток Європи.

Система – розробка інформаційно-пошукової системи за допомогою JavaScript.

Фреймворк – набір готових бібліотек, інструментів та правил, які спрощують і прискорюють розробку програмного забезпечення.

Bootstrap – CSS-фреймворк для розробки вебінтерфейсів. Він пропонує широкий вибір готових стилів, компонентів та інструментів для швидкого та зручного проєктування вебсторінок. Використання Bootstrap допомагає створити адаптивний та естетичний дизайн [4].

EJS – HTML-шаблонізатор, який дозволяє безпосередньо вбудовувати JavaScript код у сторінки HTML. Використання EJS дозволяє створювати динамічні шаблони, куди можна додавати дані з боку сервера [7].

Express.js – це легкий і гнучкий фреймворк для розробки вебдодатків JavaScript, побудованих на Node.js. Він пропонує зручний засіб створення серверної логіки та обробки HTTP-запитів [8].

MongoDB – документоорієнтована база даних, де дані зберігаються у вигляді документів у форматі, схожому на JSON. Це сучасне рішення з великою гнучкістю та можливістю масштабування [12].

Mongoose – це модуль Node.js, який спрощує роботу з MongoDB. Надає практичний спосіб визначення структури документа, проведення перевірки даних і взаємодії з базою даних. Використовуючи Mongoose, можна створити чітку модель даних і легко інтегрувати її в сервер і базу даних MongoDB [13].

Node.js – кросплатформове середовище виконання з відкритим вихідним кодом і бібліотека, яка дозволяє запускати JavaScript код поза браузером [16].

1.2 Функціональні вимоги

Функціональне призначення системи «CultureQuest» – реалізувати інформаційно-пошукову систему, яка повинна забезпечити функціонал пошуку, перегляду та додавання історичних пам’яток Європи та бути ефективним інструментом для дослідження культурної спадщини. ІПС повинна швидко знаходити запитуваний об’єкт, дозволяти користувачам додавати власні пам’ятки, що збільшуватиме кількість та різноманітність доступних історичних об’єктів.

Основною метою створення пошукової системи з історичними пам’ятками «CultureQuest» є забезпечення зручного та швидкого доступу до великої кількості різноманітних об’єктів культурної спадщини Європи для допомоги та натхнення користувачів у їх подорожах і дослідженнях. «CultureQuest» підтримується сучасними стандартами розробки інформаційно-пошукових систем, забезпечуючи оптимальну функціональність, комфортний інтерфейс та привабливий дизайн.

Інформаційна-пошукова система «CultureQuest» повинна надавати такий фунціонал:

– переглядати історичні пам’ятки;
– здійснювати пошук;
– додавати нові пам’ятки;
– переглядати останні додані об’єкти до системи;
– переглядати випадкові пам’ятки.

Функціонал пошукової системи спрямований на підвищення зацікавленості та зручності для користувачів, які бажають вивчити культурну спадщину, мати натхнення для своїх подорожей, відкрити нові історичні місця

та бажання отримати доступ до різноманітної інформації про культурні об’єкти Європи.

1.3 Нефункціональні вимоги

Нефункціональні вимоги (Non-functional requirements, NFRs) – це характеристики та атрибути системи, які визначають її якість, ефективність та інші аспекти, що не стосуються конкретної функціональності. Вони описують, як система повинна виконувати свої функції, і можуть включати вимоги до продуктивності, безпеки, надійності, зручності використання тощо. Нефункціональні вимоги є критичними для успіху проєкту, оскільки вони впливають на користувацький досвід, зручність та загальну ефективність системи [11].

Одні з основних категорій нефункціональних вимог до ІПС «CultureQuest» описані нижче.

1.3.1 Інтерфейс користувача

Інтерфейс користувача інформаційно-пошукової системи «CultureQuest» розробляється з урахуванням сучасних стандартів юзабіліті та дизайну.

Основні вимоги до інтерфейсу:

– зручність використання (Usability): інтерфейс повинен бути інтуїтивно зрозумілим і доступним для користувачів різного рівня підготовки. Навігація по системі має бути логічною та простою;
– адаптивність (Responsiveness): інтерфейс має коректно відображатися на різних пристроях, включаючи комп’ютери, ноутбуки, планшети та смартфони;

– консистентність (Consistency): всі елементи інтерфейсу повинні мати єдиний стиль оформлення – це стосується кольорової гами, шрифтів, розміщення елементів та взаємодії з ними.

1.3.2 Підтримка браузерів

Система «CultureQuest» повинна коректно функціонувати в наступних браузерах:

– Google Chrome;
– Opera;
– Mozilla Firefox;
– Microsoft Edge.

Система повинна забезпечувати однакову функціональність та відображення у всіх підтримуваних браузерах. Використання полів та технологій, які не підтримуються зазначеними браузерами, є недопустимим.

1.3.3 Вимоги до продуктивності

Продуктивність системи «CultureQuest» є вважливим фактором для забезпечення якісного користувацького досвіду. Основні вимоги до продуктивності включають:

– час відгуку (Response Time): час відгуку системи на запити користувачів не повинен перевищувати 2 секунд для основних операцій пошуку та навігації;
– навантаження (Load Handling): система повинна мати можливість ефективно обробляти зростаючий обсяг даних і кількість користувачів без зниження продуктивності;

– оптимізація запитів (Query Optimization): всі запити до бази даних MongoDB повинні бути оптимізовані для забезпечення швидкої обробки великих обсягів даних.

1.3.4 Вимоги до безпеки

Система повинна забезпечувати надійний захист даних і безперебійну роботу, дотримуючись наступних вимог до безпеки. Всі конфіденційні дані, що передаються між клієнтом і сервером, повинні бути зашифровані з використанням протоколу HTTPS, що забезпечує їх захист від несанкціонованого доступу. Система повинна мати захист від основних типів вебатак. Регулярне резервне копіювання бази даних забезпечує відновлення інформації у випадку збою або втрати даних, що гарантує збереження та доступність критично важливої інформації. Ці заходи забезпечують високий рівень безпеки інформаційно-пошукової системи «CultureQuest» та захист її користувачів.

1.4 Опис предметної області

Предметною областю є розробка інформаційно-пошукової системи для визначних культурних пам’яток Європи, під назвою «CultureQuest». Система має на меті забезпечити користувачів зручним і швидким доступом до інформації про найвизначніші культурні об’єкти Європи, сприяючи їх дослідженню та вивченню. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері та потрапити на головну сторінку ІПС «CultureQuest».

Після входу до системи користувач має можливість взаємодіяти з такими розділами як: «Home», «About», «Submit», «Contact». Перед початком додавання культурної пам’ятки потрібно перейти до розділу «Submit».

Процес внесення інформації про новий культурний об’єкт проходить наступним чином. Користувач обирає розділ «Submit», система відображає інтерфейс взаємодії з даним розділом. Для створення нового запису користувач заповнює форму, яка складається з одинадцяти пунктів, а саме: електронна пошта користувача, назва культурного об’єкта, опис, час роботи, вартість відвідування, контактний номер телефону, вебсайт, вибір відповідної категорії (країни, до якої відноситься пам’ятка), координати (широта та довгота), та останнім пунктом форми є додавання зображення. Після введення даних користувач натискає кнопку «Submit Sight», система зберігає введені дані, у відповідній категорії з’являється новий запис про визначну культурну пам’ятку Європи. Ці дані стають доступними для перегляду та пошуку іншими користувачами системи, що сприяє розширенню бази даних та обміну інформацією про культурні об’єкти.

Окрім основних функцій система надає адміністраторам можливість налаштовувати ІПС, управляти даними, займатися моніторингом та підтримкою системи, забезпечувати безпеку.

1.5 Опис інформаційно-пошукової системи

Інформаційно-пошукова система «CultureQuest» створена для забезпечення доступу до найвизначніших культурних пам’яток Європи. Завдяки своїй функціональності та зручному інтерфейсу, вона надає користувачам можливість швидко та ефективно знаходити та досліджувати історичні об’єкти, пам’ятки архітектури, музеї, пам’ятники та інші цікаві місця культурної спадщини Європи. У зв’язку зі зростаючим інтересом до культурної спадщини та туризму в Європі, така система стає невід’ємною складовою для тих, хто планує подорожі, вивчає історію або просто цікавиться культурними пам’ятками.

Згідно з результатами досліджень, виявлено, що в 2023 році туристичний потік в Європу майже досяг допандемійного рівня. У четвертому кварталі 2023

року кількість іноземних туристів, які прибули до Європи, була на 1,6% нижча, ніж у 2019 році. Однак кількість ночей, проведених іноземними туристами в Європі, виявилася лише на 0,6% нижчою порівняно з даними до початку пандемії. З огляду на це, очікується подальше зростання туризму в Європі у 2024 році [23].

Основною метою «CultureQuest» є надання користувачам зручного та ефективного інструменту для пошуку інформації про визначні культурні об’єкти, такі як пам’ятки архітектури, музеї, пам’ятники, археологічні знахідки та інше. Система надає доступ до різноманітної інформації про кожен об’єкт, включаючи його історію, географічне розташування, відомості, фотографії та інше.

«CultureQuest» вирізняється серед інших систем своєю широкою базою даних, якісною інформацією, зручним та інтуїтивно зрозумілим інтерфейсом, який дозволяє користувачам швидко знаходити потрібну інформацію. Крім того, система постійно оновлюється та розширюється, щоб задовольнити потреби користувачів у доступі до найактуальнішої інформації про культурні пам’ятки Європи.

Користувач виступає в ролі читача та дослідника, який зацікавлений у пошуку та вивченні визначних культурних пам’яток Європи. Його цікавить нова інформація та можливість поділитися своїм досвідом з іншими користувачами.

Адміністратор відповідає за конфігурацію та налаштування системи, забезпечує її надійність та безпеку, вирішує технічні питання і здійснює моніторинг.

Менеджер з розвитку системи залучає нових користувачів та партнерів, розвиває функціональність системи, укладає контракти, які сприяють подальшому розвитку та популяризації ІПС.

Контент-менеджер відповідає за створення та редагування контенту, наповнення бази даних актуальними матеріалами про культурні пам’ятки Європи, а також відслідковує та аналізує тренди у сфері культурного туризму.

Менеджер з взаємодії з користувачами відповідає за взаємодію з користувачами, вирішення їх запитань та проблем, збір фідбеку щодо функціоналу системи та пропозиції щодо її вдосконалення.

Основними функціями системи є:

– публікація актуальних пам’яток;
– можливість перегляду історичних пам’яток за категоріями;
– пошук.

Ці функції забезпечують правильну продуктивність та відповідність завданням, поставленим перед роботою інформаційно-пошукової системи.

1.6 Програми аналоги

На сьогоднішній день існує декілька популярних інформаційно-пошукових систем, які надають користувачам можливість знаходити та досліджувати визначні культурні пам’ятки по всьому світу. Найбільш відомими серед них є:

– Atlas Obscura;
– Culture Trip;
– National Geographic.

Розглянемо більш детально кожну з наведених вище інформаційно-пошукових систем.

Atlas Obscura – це одна з найпопулярніших платформ, яка спеціалізується на незвичайних та маловідомих культурних пам’ятках і місцях. Сайт був заснований у 2009 році і наразі є великою базою даних, що містить понад 22,000 об’єктів. Переваги Atlas Obscura включають:

– широкий вибір унікальних місць: користувачі можуть знайти інформацію про маловідомі та незвичайні місця, які не завжди входять до традиційних туристичних маршрутів;
– зручний пошук та фільтри: сайт дозволяє легко шукати пам’ятки за географічним розташуванням, типом або тематикою;

– спільнота мандрівників: користувачі можуть додавати власні відкриття, коментарі та фотографії, що робить сайт живою та постійно оновлюваною платформою.

Недоліками Atlas Obscura можна вважати те, що основний акцент робиться на незвичайних місцях, що може не відповідати потребам тих, хто шукає інформацію про більш традиційні культурні пам’ятки. Також інтерфейс може здатися трохи перевантаженим для нових користувачів [3].

Culture Trip – це сайт, який надає інформацію про подорожі, культуру, їжу та події в різних країнах. Заснований у 2011 році, сайт пропонує широкий спектр статей та рекомендацій, створених місцевими авторами та експертами. Переваги Culture Trip включають:

– різноманітність контенту: крім культурних пам’яток, сайт містить статті про місцеву кухню, події та інші аспекти культури;
– локальні рекомендації: автори статей є місцевими жителями або експертами, що дозволяє отримати достовірну та детальну інформацію;
– інтерактивність: користувачі можуть зберігати свої улюблені статті та місця, що полегшує планування подорожей.

Серед недоліків Culture Trip можна відзначити те, що інколи статті можуть мати рекламний характер, що впливає на об’єктивність контенту. Також сайт може бути менш зручним для тих, хто шукає суто фактичну інформацію без додаткового контексту [5].

National Geographic – це всесвітньо відомий бренд, який надає інформацію про подорожі, природу, науку та культуру. Сайт National Geographic містить велику кількість статей, фотографій та відео, що робить його одним з найавторитетніших джерел інформації. Переваги National Geographic включають:

– висока якість контенту: матеріали створюються професійними журналістами, фотографами та експертами, що гарантує високу якість та достовірність інформації;

– різноманітність форматів: користувачі можуть знайти статті, фотографії, відео та інтерактивні карти, що робить дослідження культурних пам’яток більш захоплюючим;
– глобальне охоплення: сайт охоплює культурні пам’ятки з усього світу, що дозволяє отримати повну картину культурного різноманіття планети.

Недоліком National Geographic є те, що доступ до деяких матеріалів може бути платним, що може обмежити доступ до інформації для деяких користувачів. Також сайт може бути перевантаженим рекламою [14].

Нижче в таблиці 1.1 наведено порівняльну характеристику системи, що розроблюється, з існуючими аналогами.

Таблиця 1.1 – Порівняльна характеристика

Параметр	**Atlas Obscura**	**Culture Trip**	**National Geographic**	**CultureQuest**
Основний контент	Незвичайні та маловідомі місця	Місцева культура та подорожі	Культура, подорожі, наука	Визначній культурні пам’ятки Європи
Інтерактивність	Висока	Висока	Середня	Висока
Якість контенту	Висока	Середня	Висока	Висока
Рекламний контент	Присутній	Присутній	Присутній	Відсутній
Платний доступ	Відсутній	Відсутній	Присутній	Відсутній
Основний користувач	Мандрівники, дослідники	Мандрівники, місцеві жителі	Широке коло користувачів	Мандрівники, дослідники
Інтерфейс	Пере-вантажений	Простий	Збалан-сований	Простий, інтуїтивний

Усі ці інформаційно-пошукові системи надають користувачам можливість досліджувати культурні пам’ятки та планувати подорожі, але кожна з них має свої унікальні особливості та підходи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНО-ПОШКОВОЇ СИСТЕМИ

2.1 Використання UML під час розробки системи

Використання UML (Unified Modeling Language) під час розробки інформаційно-пошукової системи є невід'ємною частиною процесу проєктування та реалізації програмного забезпечення. UML надає стандартизований набір графічних засобів і нотацій, що дозволяє ефективно моделювати різні аспекти системи перед її впровадженням.

Одним із головних переваг використання UML для одного розробника є можливість чітко візуалізувати та структурувати свої ідеї. Це допомагає зберігати ясність і послідовність у проєкті, а також уникнути можливих помилок на етапі проєктування.

UML також полегшує документування системи, що важливо як для поточної розробки, так і для майбутнього обслуговування та оновлення. Графічні діаграми дозволяють швидко переглядати та розуміти структуру і функціональність системи, що може бути корисним при поверненні до проєкту після певного часу [12].

Для проєктування інформаційно-пошукової системи доцільно використовувати конкретні типи діаграм UML, розглянемо їх.

Діаграма варіантів використання (Use Case Diagram). Показує функціональні можливості системи з точки зору користувачів (акторів). Це загальний огляд того, що система робить і хто взаємодіє з нею [13].

Діаграма основного процесу ІПС у нотації IDEF0. Це методологія моделювання функцій, яка використовується для створення функціональних моделей систем. Вона представляє собою діаграми, які описують функції системи та їх взаємозв'язки [14].

Діаграма декомпозиції основного блоку ІПС у нотації IDEF0. Це процес деталізації функцій верхнього рівня на підфункції нижчого рівня, що дозволяє краще зрозуміти внутрішні процеси системи [14].

IDEF1X-діаграма. Це методологія моделювання даних, яка використовується для проєктування баз даних. Діаграми в IDEF1X допомагають візуалізувати сутності, їх атрибути та взаємозв’язки між ними. Основною метою IDEF1X є створення точних, зрозумілих та стандартизованих моделей даних [15].

У проєкті інформаційно-пошукової системи UML використовується для моделювання всіх ключових аспектів системи:

– аналіз вимог: UML діаграми допомагають чітко визначити функціональні та нефункціональні вимоги до системи, що дозволяє уникнути невизначеностей та непорозумінь;
– проєктування архітектури: діаграми класів і об’єктів використовуються для визначення основних компонентів системи та їхніх взаємозв’язків, що дозволяє побудувати оптимальну архітектуру;
– моделювання поведінки: діаграми діяльності та діаграми станів допомагають змоделювати бізнес-процеси та поведінку об’єктів у різних ситуаціях, забезпечуючи правильну реалізацію функціональності.

Використання UML під час розробки інформаційно-пошукової системи забезпечує структурований підхід до проєктування. Це сприяє більш ефективній розробці та впровадженню програмного забезпечення, яке відповідає вимогам і функціональним очікуванням. UML допомагає зберігати організованість, зменшує ризик помилок і полегшує подальше обслуговування системи.

2.2 Загальна схема роботи системи

Загальна схема роботи системи описує основні компоненти та їх взаємодію для забезпечення її функціонування. Це схематичне представлення

архітектури системи, яке включає основні етапи обробки даних від користувацького інтерфейсу до бази даних і назад.

На рисунку 2.1 зображено загальну схему роботи системи.

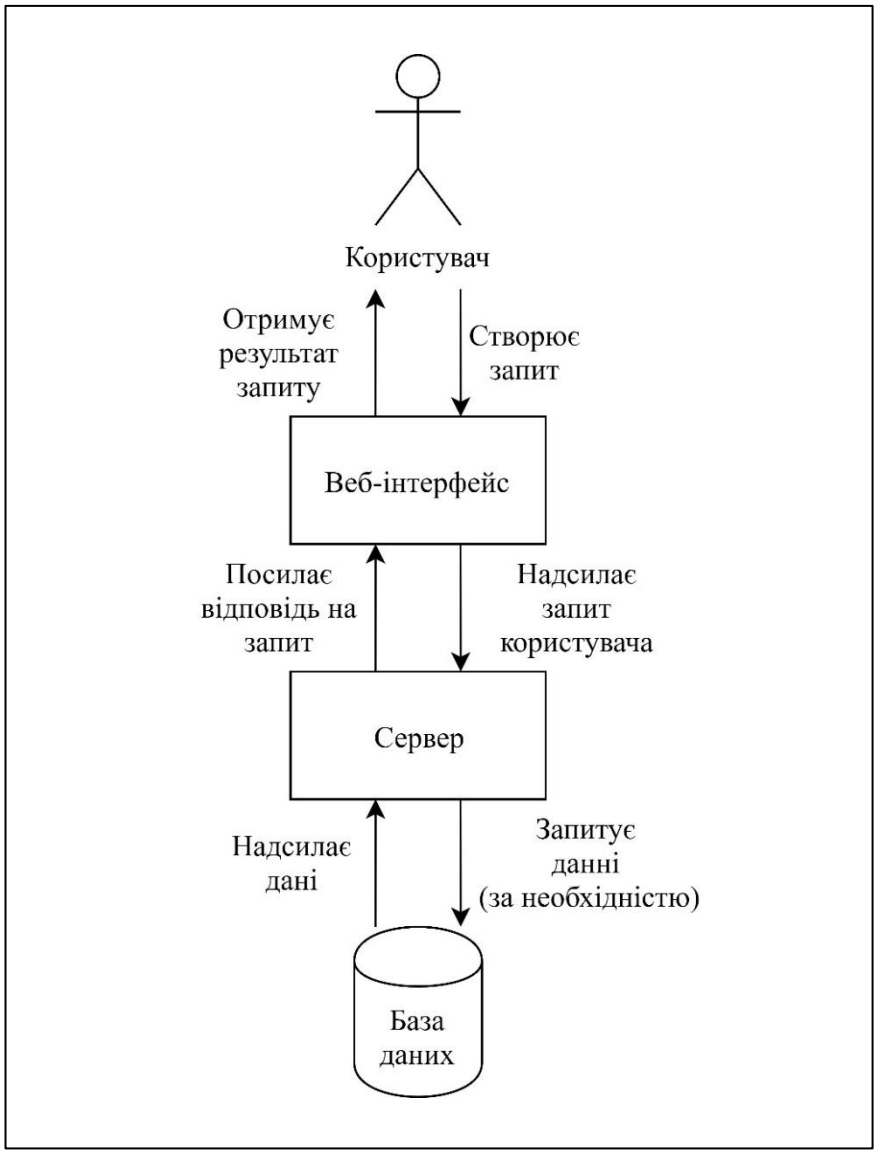


Рисунок 2.1 – Загальна схема роботи програми

Інформаційно-пошукова система «CultureQuest» побудована за принципом клієнт-серверної архітектури, яка включає кілька основних компонентів, що забезпечують її функціонування та взаємодію з користувачами.

Користувач взаємодіє з системою через веббраузер, вводячи пошукові запити, переглядаючи результати та використовуючи різні функції системи.

Клієнтська частина (Frontend), реалізована за допомогою HTML, CSS та JavaScript, відповідає за відображення інтерфейсу користувача. Вона обробляє взаємодії з користувачем, виконує валідацію даних на стороні клієнта та надсилає запити до серверної частини.

Серверна частина (Backend), створена з використанням технологій Node.js, обробляє запити від клієнтської частини. Сервер отримує запити, взаємодіє з базою даних, формує відповіді, які потім повертаються до користувача. База даних MongoDB, зберігає інформацію про культурні пам’ятки. Сервер здійснює запити до БД для отримання, збереження та оновлення даних, оптимізуючи їх для швидкого пошуку і обробки великих обсягів інформації.

Завдяки цим компонентам і механізмам, ІПС «CultureQuest» забезпечує ефективну, безпечну та швидку роботу, надаючи користувачам зручний доступ до інформації про визначні культурні пам’ятки Європи.

2.3 Функціональна модель ІПС

Функціональна модель в нотації IDEF0 (Integration Definition for Function Modeling) – це метод для моделювання бізнес-процесів, який використовується для представлення функцій системи, взаємодії між цими функціями та інформаційними потоками, що підтримують ці функції. IDEF0 моделі допомагають зрозуміти структуру і поведінку системи на високому рівні, що є корисним для аналізу, проєктування та документування процесів [14].

На рисунку 2.2 представлена функціональна модель інформаційно-пошукової системи «CultureQuest».

Опис основного процесу: «Додавання об’єкту культурної спадщини».

Функція: додавання нового об’єкту культурної спадщини.

Інформаційні потоки:

– вхідні дані: інформація про новий об’єкт (назва об’єкту, опис, час роботи, вартість відвідування, номер телефону, вебсайт, категорія, координати, зображення);
– вихідні дані: підтвердження додавання, оновлена база даних.

Механізми: користувач, необхідне обладнання.

Механізми перевірки: політики перевірки даних, документація по створенню інформаційно-пошукових систем.

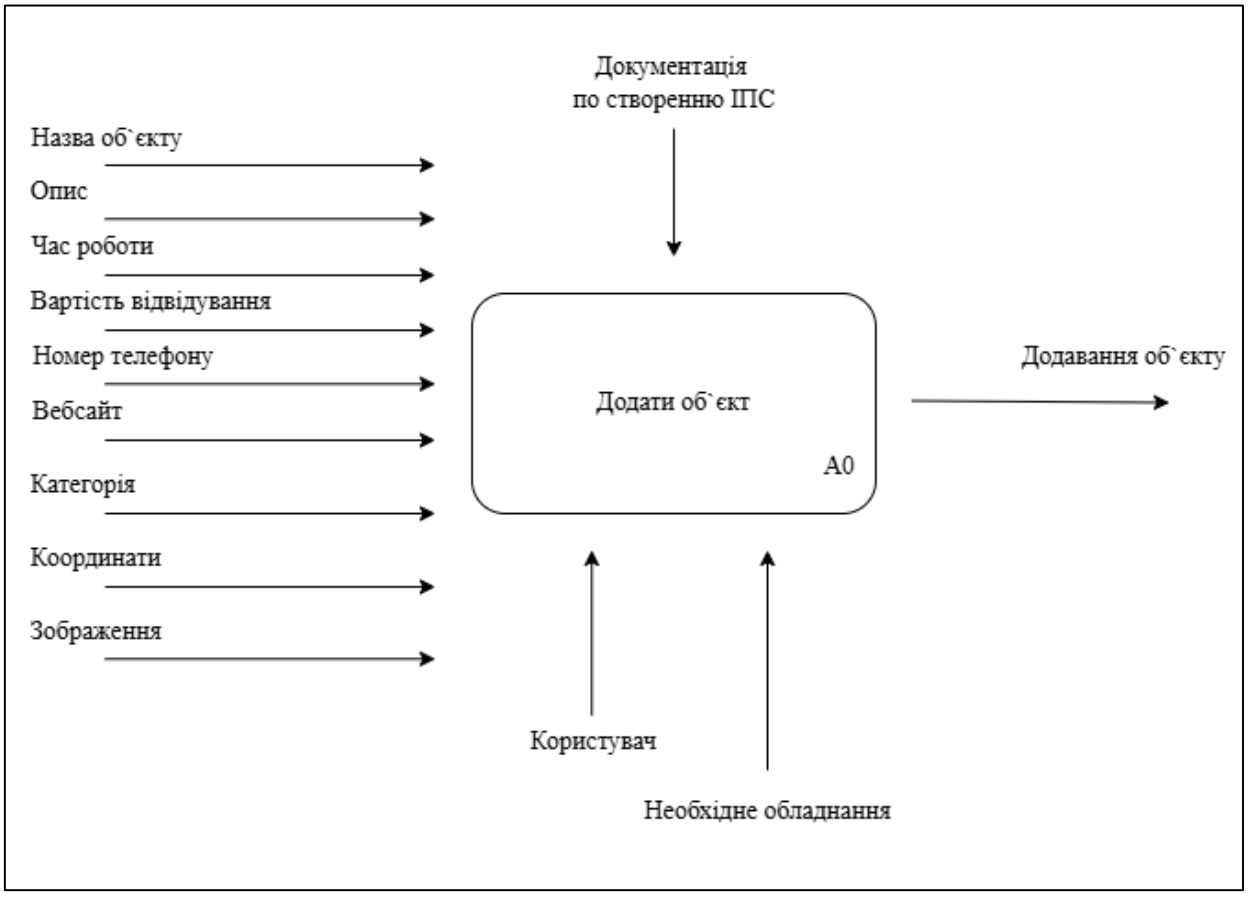


Рисунок 2.2 – Основний процес ІПС у нотації IDEF0

Функціональна модель основного процесу в нотації IDEF0 для інформаційно-пошукової системи «CultureQuest» допомагає зрозуміти, як відбувається додавання нового об’єкту культурної спадщини. Вона описує основні кроки процесу, взаємодії між ними та використовувані механізми, але для більш точного та глибокого розуміння процесу додавання нових об’єктів потрібно провести декомпозицію блоку «Додати об’єкт» (див. рис. 2.3).

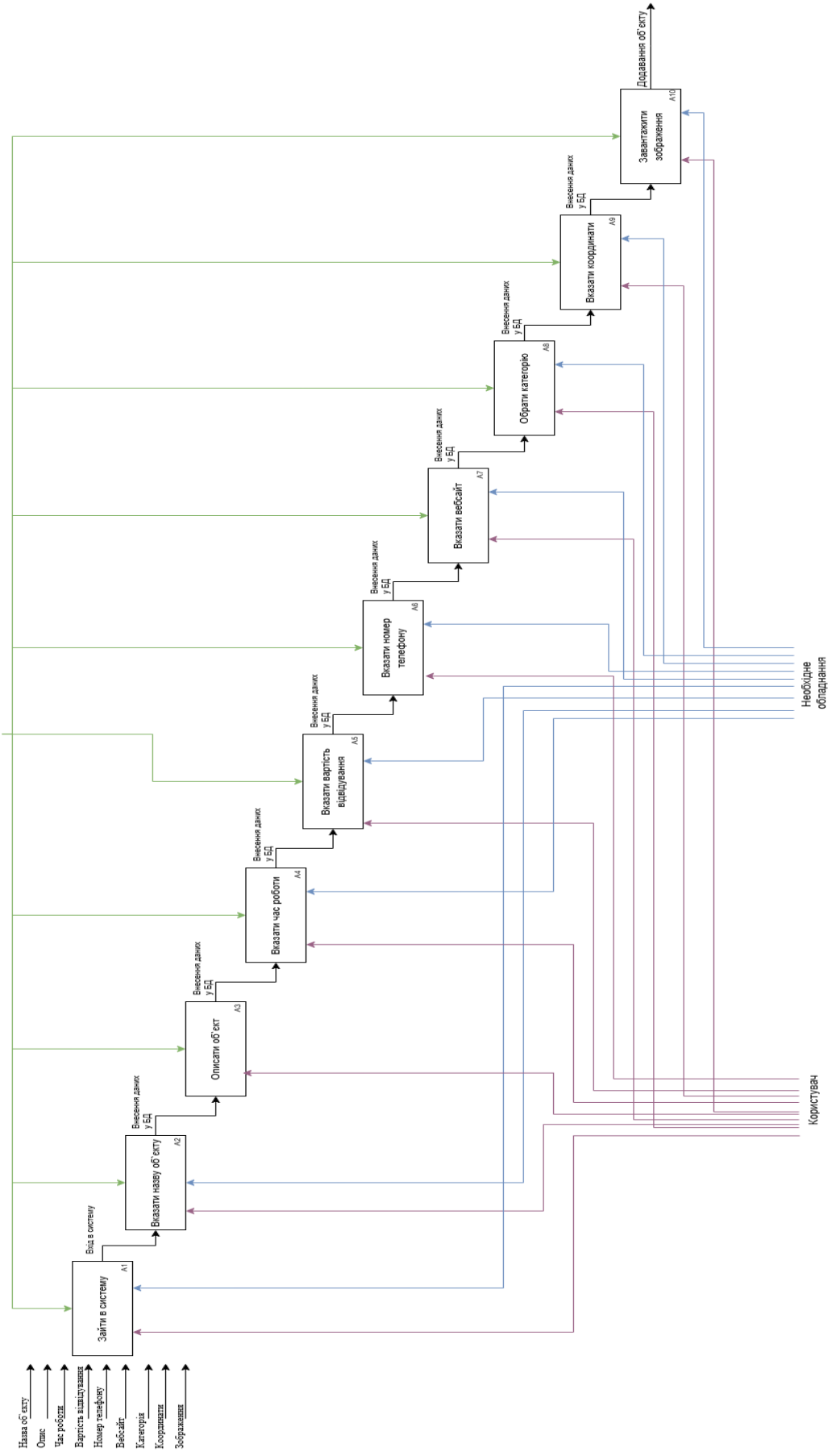


Рисунок 2.3 – Декомпозиція основного блоку у нотації IDEF0

Дана діаграма демонструє детальну декомпозицію процесу «Додати об’єкт» у системі, яка розкриває всі ключові етапи цього процесу, включаючи дії користувача і операції, які виконує система. Вона дозволяє чітко побачити послідовність виконання завдань, починаючи від входу в систему до завантаження зображень і внесення всієї інформації в базу даних. Діаграма допомагає зрозуміти логіку роботи системи, включаючи інтеракцію користувача і внесення даних. Це дозволяє ідентифікувати можливі точки збоїв, оптимізувати процес або додати нові функціональні можливості. Також декомпозицію процесу «Додати об’єкт» слугує основою для проєктування інтерфейсу, алгоритмів або бази даних, така деталізація забезпечує ефективну організацію процесів і підвищення якості роботи системи.

2.4 Інформаційна модель ІПС

Під час розробки інформаційно-пошукової системи «CultureQuest» для організації зберігання та управління даними використовується база даних MongoDB, яка є документно-орієнтованою. MongoDB дозволяє зберігати інформацію у вигляді документів формату JSON (BSON), що робить її гнучкою та зручною для роботи з великими обсягами різноманітних даних. Основною перевагою MongoDB є її здатність легко адаптуватися до змін у структурі даних без необхідності значного перепроєктування системи [16].

Для забезпечення зв’язку між сервером і базою даних використано бібліотеку Mongoose, яка виступає як інструмент для роботи з MongoDB у середовищі Node.js. Mongoose є ORM (Object Relational Mapper), який дозволяє створювати схеми даних, визначати структуру документів, виконувати валідацію, а також забезпечує зручний доступ до даних. Використання Mongoose дозволяє організувати роботу з MongoDB на більш високому рівні абстракції, що спрощує розробку та полегшує підтримку системи [17].

У базі даних створено дві основні колекції: Sights та Categories. Колекція Sights містить інформацію про туристичні об’єкти, такі як назва об’єкта, опис, час роботи, вартість відвідування, контактний телефон, вебсайт, координати, електронна пошта, зображення та категорія. Поле category у цій колекції є посиланням (Reference) на документ з колекції Categories, що забезпечує зв’язок між туристичним об’єктом і його категорією. Колекція Categories, у свою чергу, містить дані про категорії, включаючи назву та зображення.

Для роботи з цими колекціями у Mongoose створено моделі, які ґрунтуються на визначених схемах. Наприклад, схема для Sights визначає всі поля, що повинні бути присутніми в документі, разом із їх типами та можливими обмеженнями. Аналогічно, схема для Categories описує структуру даних для цієї колекції. Схеми також забезпечують виконання валідації даних перед їх внесенням у базу, що гарантує узгодженість інформації в системі.

Ключовою перевагою використання Mongoose є можливість автоматичного управління зв’язками між колекціями. Завдяки методам, таким як populate, сервер може запитувати дані про об’єкти разом із пов’язаною інформацією про їх категорії. Це особливо корисно для оптимізації запитів та мінімізації дублювання даних [22].

Таким чином, поєднання MongoDB та Mongoose забезпечує гнучку, масштабовану та зручну у використанні архітектуру бази даних. MongoDB надає можливість ефективно зберігати різноманітні дані, тоді як Mongoose дозволяє впорядкувати та автоматизувати взаємодію з цими даними, що суттєво спрощує реалізацію функціоналу системи. Це рішення добре підходить для створення динамічних вебдодатків, де потрібна висока продуктивність і простота інтеграції [24].

IDEF1X-діаграми використовуються для моделювання даних і проєктування структур баз даних. Вони відображають сутності, їх атрибути та зв’язки між ними, допомагаючи зрозуміти логіку даних і оптимізувати їх структуру. Основною перевагою цих діаграм є чіткість і зручність графічного представлення складних структур, що спрощує розуміння моделі. Вони також

сприяють нормалізації даних, виявленню помилок на етапі проєктування та ефективній документації системи [10].

У даному проєкті IDEF1X-діаграма забезпечує зручне візуальне представлення зв’язків між колекціями Sights і Categories, що спрощує розробку бази даних «Sights». Завдяки їй вдалося правильно визначити зв’язки між сутностями, уникнути дублювання даних та створити чітку структуру, яка легко масштабується й підтримується. Ця діаграма також полегшила документування моделі, що є корисним для подальшої розробки й підтримки системи.

На рисунку 2.4 зображено схему бази даних «Sights», що використовується у системі.

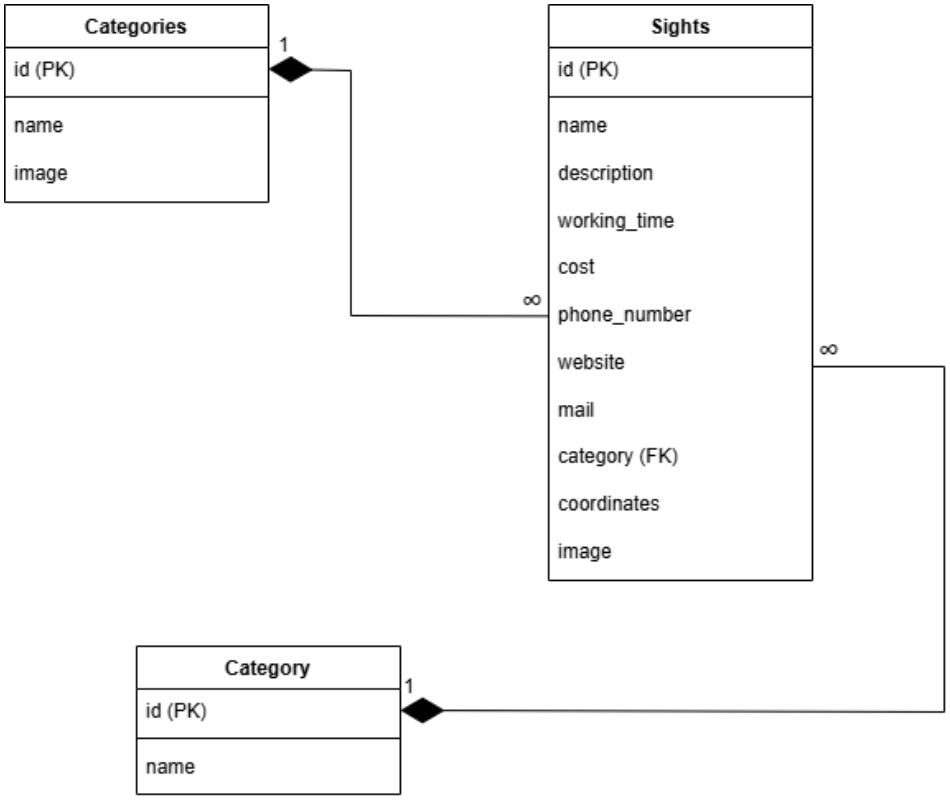


Рисунок 2.4 – IDEF1X-діаграма

Одним із ключових елементів бази даних є таблиці, які зберігають структуровані дані про сутності системи. БД «CultureQuest» має дві основні колекції: «Categories» та «Sights». Перша зберігає інформацію про категорії об’єктів, тоді як друга містить деталі про конкретні туристичні об’єкти та їх

характеристики. Для полегшення розуміння структури цих колекцій були сформовані таблиці, які представлено нижче.

Таблиця «Categories» (див. табл. 2.1) містить інформацію про культурні об’єкти, такі як назва, зображення.

У таблиця «Sight» (див. табл. 2.2) зберігаються дані про туристичні об’єкти з прив’язкою до відповідних категорій.

Таблиця 2.1 – Categories

Назва поля	**Тип даних**	**Опис**
id	ObjectId	Унікальний ідентифікатор категорії (первинний ключ).
name	String	Назва категорії.
image	String	Зображення, яке представляє категорію.

Таблиця 2.2 – Sight

Назва поля	**Тип даних**	**Опис**
id	String	Унікальний ідентифікатор об’єкта (первинний ключ).
name	String	Назва культурного об’єкта.
description	String	Опис об’єкта.
working_time	String	Графік роботи об’єкта.
cost	String	Вартість відвідування.
phone_number	String	Контактний номер телефону.
website	String	URL-адреса вебсайту об’єкта.
mail	String	Електронна адреса користувача
category	String	Зовнішній ключ, що посилається на id категорії в таблиці Categories.
coordinates	String	Географічні координати об’єкта (широта і довгота).
image	String	Зображення, пов’язане з об’єктом.

2.5 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це графічне представлення функціональних можливостей системи з точки зору користувачів (акторів). Вона показує взаємодію між користувачами і системою та описує основні сценарії використання системи. Діаграма варіантів використання допомагає зрозуміти, які дії можуть виконуватися в системі, хто ці дії виконує та як вони пов’язані між собою [19].

На рисунку 2.5 зображено діаграму використання ІПС «CultureQuest».



Рисунок 2.5 – Діаграма використання Користувачем

Для інформаційно-пошукової системи «CultureQuest», основними варіантами використання є:

– перегляд культурних пам’яток Європи за категоріями: користувач може вибрати певну категорію, щоб переглянути список пам’яток, що належать до цієї країни;
– перегляд будь-якого випадкового об’єкту: користувач може переглянути випадково обраний об’єкт культурної спадщини, який доступний у базі даних;

– перегляд найновіших доданих пам’яток: користувач може переглянути список останніх доданих до системи пам’яток;
– пошук необхідного об’єкту культурної спадщини: користувач може використовувати функцію пошуку для знаходження конкретних об’єктів за різними критеріями, такими як назва або категорія;
– розміщення власних пам’яток: користувач може додати новий об’єкт культурної спадщини, заповнивши відповідну форму і надаючи необхідну інформацію.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Інформаційно-пошукова система «CultureQuest» була розроблена з використанням сучасних вебтехнологій, які забезпечують її надійність, масштабованість і зручність у використанні. Детально розглянемо інструменти, які були використані для розробки проєкту.

Мовою програмування системи є JavaScript, дану мову було обрано через її універсальність і широке використання в розробці вебдодатків. Ця мова дозволяє ефективно створювати як серверну, так і клієнтську частину додатку [20]. Серверна частина проєкту побудована з використанням Express.js – популярного і гнучкого фреймворка для Node.js. Його ключові переваги включають простоту налаштування маршрутизації, роботу з middleware і можливість швидкої інтеграції сторонніх модулів. Сервер відповідає за обробку пошукових запитів, взаємодію з базою даних і рендеринг сторінок [21].

Для зберігання даних використовується база даних MongoDB, яка забезпечує гнучку і динамічну роботу з інформацією. БД дозволяє зберігати дані у форматі JSON, що спрощує інтеграцію з JavaScript-середовищем. У проєкті MongoDB використовується для зберігання даних про пам’ятки культури, включно з назвами, описами, зображеннями та іншою інформацією [12]. Для реалізації функції пошуку застосовано текстовий індекс, який підтримує повнотекстовий пошук із врахуванням діакритичних знаків, тобто спеціальних символів або позначок, які додаються до літер у деяких мовах для зміни їхньої вимови, значення або наголосу. Вони можуть використовуватися для позначення тонів, довготи голосних, акцентів, м’якості звуку та інших фонетичних характеристик. Завдяки цьому користувачі можуть легко знаходити потрібну інформацію за ключовими словами [22].

Інтерфейс користувача реалізовано із застосуванням стандартних вебтехнологій, таких як HTML, CSS і JavaScript. Для прискорення розробки та забезпечення адаптивності дизайну використовується фреймворк Bootstrap, який надає готові стилі та компоненти для створення сучасного інтерфейсу. Bootstrap є потужним інструментом для веброзробки, що дозволяє швидко створювати естетично привабливі сторінки завдяки наявності готових компонентів і стилів [4].

Для динамічного створення HTML-сторінок у проєкті застосовується EJS (Embedded JavaScript Templates). Цей шаблонізатор дозволяє ефективно поєднувати серверні дані із фронтендом, що особливо зручно для відображення списків результатів пошуку чи деталей конкретних пам’яток. Наприклад, шаблон search.ejs використовується для рендерингу сторінки з результатами пошуку, переданими із бази даних [7].

Архітектура проєкту базується на модульному підході, де кожна функція реалізована у відповідному файлі. Основний файл додатку app.js відповідає за налаштування сервера, підключення маршрутів і middleware. Логіка обробки пошукових запитів реалізована у вигляді окремого методу searchSight, який використовує асинхронний доступ до бази даних для отримання результатів за ключовим словом. Для маршрутизації запитів і організації роботи з користувачами застосовується файл sightRoutes.js.

Система також підтримує функціонал завантаження зображень пам’яток культури завдяки використанню бібліотеки express-fileupload. Це забезпечує зручне додавання нових записів із візуальним контентом, що покращує сприйняття інформації для користувачів.

Для розробки інформаційно-пошукової системи «CultureQuest» використовувалося середовище розробки Visual Studio Code, яке є одним із найзручніших інструментів для програмістів завдяки своїй універсальності, легкості у використанні та широким можливостям для розширення функціональності. Цей редактор став центральним інструментом у процесі розробки як серверної, так і клієнтської частини проєкту [20].

«CultureQuest» демонструє ефективне використання сучасних технологій для побудови надійної інформаційно-пошукової платформи. Вибір JavaScript та пов’язаних інструментів забезпечив оптимальну інтеграцію між серверною і клієнтською частинами, а використання MongoDB дозволило створити гнучкий механізм роботи з даними. Усі ці елементи у поєднанні забезпечують функціональність, продуктивність і зручність системи, орієнтованої на користувачів.

3.2 Процес створення сервера

Процес створення сервера інформаційно-пошукової системи на платформі Node.js за допомогою Express включає кілька етапів, включаючи налаштування сервера, створення маршруту, обробку запитів і відображення даних через шаблони.

Першим кроком є підключення необхідних залежностей, таких як Express – популярний вебфреймворк для Node.js, який полегшує створення серверів. Файл sightRoutes.js запускає маршрутизатор за допомогою методу express.Router(), який дозволяє визначати різні маршрути для обробки запитів. Цей файл також визначає всі важливі шляхи, які використовуються для відображення сторінок, таких як домашня сторінка, сторінка категорії, культурні відомості та багато інших. Кожен маршрут, який відповідає на HTTP запит, зв’язаний з відповідним методом контролера. Наприклад, для обробки запиту на домашню сторінку визначено маршрут GET/, який викликає функцію homepage з файлу контролера sightController.js.

Контролери відповідають за логіку обробки запитів. Вони використовують моделі для взаємодії з базою даних, отримують дані і передають їх в шаблони для відображення. У файлі sightController.js визначено низку функцій, кожна з яких обробляє запити до різних розділів системи, таких як категорії пам’яток, деталі окремих пам’яток, пошук та інші. Для отримання даних використовуються моделі Mongoose для роботи з MongoDB. Наприклад,

функція exploreCategories отримує список категорій пам’яток і передає їх на рендеринг в шаблон categories.ejs, а функція exploreSight відповідає за відображення детальної інформації про конкретну пам’ятку.

Також реалізована функція завантаження файлів. Коли новий об’єкт додається через форму, зображення завантажується на сервер, зберігається в певному каталозі та пов’язується з новою пам’яткою в базі даних.

Шаблони, зокрема файли index.ejs, main.ejs, відображають дані, передані з контролера. Вони використовують вбудовану мову шаблонів EJS для вставки динамічних даних у HTML. Це дозволяє створювати гнучкі вебсторінки, які змінюються відповідно до запитів від сервера. Наприклад, на домашній сторінці відображаються категорії пам’яток і останні додані пам’ятки, а на сторінці кожної категорії – список пам’яток, що належать до цієї категорії.

Важливою частиною процесу є обробка помилок. Усі запити до сервера повинні бути готові до можливих помилок, таких як проблеми з базою даних або відсутність необхідних файлів. Коли виникають помилки, сервер відповідає відповідними повідомленнями, які дозволяють користувачам зрозуміти проблему.

Крім цього, додаток підтримує функцію пошуку, де користувач може вводити запит, а сервер повертає результат пошуку пам’яток за ключовими словами. Цей процес реалізовано за допомогою текстового пошуку в MongoDB для ефективного пошуку записів, пов’язаних із введеним запитом.

Цей сервер побудовано на основі принципів RESTful, де кожен маршрут відповідає за певну дію з використанням ресурсу програми (категорії, пам’ятки, пошук), і реалізує стандартні HTTP-методи – GET, POST, що робить систему простою та зрозумілою для взаємодії з іншими компонентами.

3.3 Процес створення та налаштування MongoDB

Створення та налаштування бази даних MongoDB є важливим етапом розробки ІПС «CultureQuest», що забезпечує ефективне зберігання,

організацію та доступ до даних. MongoDB – це нереляційна документно-орієнтована система управління базами даних, яка зберігає дані у форматі JSON-подібних документів. Завдяки гнучкості, масштабованості та підтримці кластеризації, MongoDB стає чудовим вибором для проєктів, які потребують динамічної структури даних [12].

Процес налаштування бази даних MongoDB починається зі створення облікового запису на платформі MongoDB Atlas, яка надає хмарний сервіс для роботи з базами даних. Після реєстрації створюється кластер, який є базовою одиницею інфраструктури хмарного зберігання даних. У цьому кластері можна створити кілька баз даних, зокрема базу даних Sights, яка є основою для цього проєкту. Права доступу, IP-адреси та облікові дані адміністратора задаються під час процесу встановлення.

У проєкті для підключення до MongoDB використовується бібліотека Mongoose, яка забезпечує простий інтерфейс для взаємодії з базою даних та підтримує функціонал валідації, схем і запитів. У файлі database.js визначено підключення до бази даних через URI, який вказує на кластер у MongoDB Atlas. З’єднання встановлюється за допомогою методу mongoose.connect, а подальший моніторинг стану з’єднання здійснюється через події error та open.

Моделі Category та Sight є основними компонентами структури бази даних. Модель Sight, яка описана у файлі Sight.js, визначає схему даних для туристичних пам’яток. Вона включає такі поля, як назва, категорія, географічні координати, робочий час, вартість, контактна інформація, опис, вебсайт, електронна пошта та зображення. Всі поля позначені як обов’язкові, що гарантує цілісність даних. Для реалізації ефективного пошуку по тексту до схеми додано індексацію полів name та description.

Важливою частиною проєкту є забезпечення зручного доступу до даних через інформаційно-пошукову систему. Інтерфейс системи містить сторінки, такі як «Home», «About», «Submit», «Contact» та сторінку пошуку, що дозволяють користувачам знаходити та подавати інформацію про туристичні пам’ятки. Дані сторінки надсилаються на сервер, а операція CRUD виконується в базі даних за допомогою Mongoose. Наприклад, пошук пам’яток

можна здійснювати за допомогою ключових слів, які надсилаються на сервер як запити, а зворотні відповіді генеруються у форматі JSON.

З міркувань безпеки URI з'єднання MongoDB Atlas зберігається у файлі a.env, який не публікується через.gitignore. Це запобігає зламу облікових даних бази даних.

Процес створення та інсталяції MongoDB охоплює кілька ключових кроків: встановлення хмарного середовища для зберігання даних, розробка структури бази даних за допомогою схем Mongoose, реалізація механізмів перевірки, індексування та інтеграція бази даних у серверну та клієнтську частини проєкту. Усі ці етапи призначені для забезпечення надійності, ефективності та масштабованості системи.

3.4 Архітектура інформаційно-пошукової системи

У цьому розділі розглядається архітектура інформаційно-пошукової системи, яка була розроблена для забезпечення зручної роботи з даними, їх обробки та представлення користувачеві. Система побудована на основі сучасних вебтехнологій, є гнучкою, масштабованою та простою в обслуговуванні. На рисунку 3.1 показано логічну структуру взаємодії основних компонентів системи, включаючи базу даних MongoDB, серверну частину на основі Node.js та Express, а також клієнтську частину, яка використовує HTML-шаблони EJS і CSS-фреймворк для створення користувацького інтерфейсу.

Представлена архітектура дозволяє забезпечити ефективний зв'язок між усіма рівнями програми, спрощуючи виконання як базових, так і більш складних запитів користувачів.

Діаграма «Архітектура інформаційно-пошукової системи» відображає логічну структуру для створення вебдодатків за допомогою Node.js, Express, MongoDB та використанні шаблонізатора EJS і CSS-фреймворків для візуалізації. Ця архітектура гарантує, що програмна логіка розділена на рівні,

кожен з яких виконує свою функцію, тим самим підвищуючи масштабованість системи, обслуговування та ефективність.

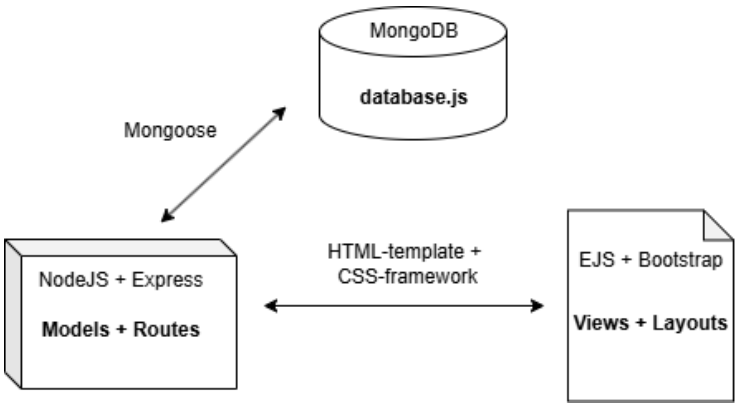


Рисунок 3.1 – Архітектура ІПС «CultureQuest»

Система базується на базі даних MongoDB і зберігає всі дані у форматі документа документів. Під час взаємодії з MongoDB використовується бібліотека Mongoose, яка дозволяє створювати моделі об’єктних документів (ODM) для опису структур даних у вигляді схем. База даних MongoDB представлена у файлі database.js, який містить логіку підключення до бази даних через MongoDB Atlas. У цій архітектурі Mongoose діє як посередник між базою даних і сервером і може реалізовувати запити, додавати перевірку, індексування та інші механізми обробки даних.

На стороні серверної логіки використовується Node.js разом із фреймворком Express, який забезпечує обробку HTTP-запитів та управління маршрутизацією. Express відповідає за зв’язок між моделями даних та маршрутами, які керують обробкою запитів від клієнта. Моделі створюються на основі схем Mongoose, таких як файли Category.js та Sight.js, які визначають структуру даних, що використовується в системі. Ця взаємодія дозволяє запускати операції CRUD, такі як створення, читання, оновлення та видалення даних, а також забезпечує гнучкість у розширенні функціоналу системи.

Клієнтська частина системи представлена HTML-шаблонами, які створюються за допомогою шаблонізатора EJS. EJS дозволяє динамічно відображати інформацію на вебсторінках, використовуючи інформацію,

отриману із серверної частини. Для покращення візуального вигляду інтерфейсу використовуються CSS-фреймворки, такі як Bootstrap, забезпечуючи адаптивність і стильний дизайн сторінки. Усі шаблони організовані у вигляді «Views» та «Layouts», дотримуючись концепції поділу візуальних елементів і повторення таких елементів, як заголовки чи навігація.

Між сервером і клієнтом встановлюється двосторонній зв'язок, при цьому сервер отримує запит клієнта, працює з базою даних MongoDB через Mongoose, обробляє дані та надсилає їх назад клієнту у вигляді HTML-сторінок. Ці сторінки створюються з динамічним вмістом, таким як список туристичних пам'яток або результати пошуку, збережені в базі даних.

Дана архітектура забезпечує чіткий розподіл завдань між рівнями системи. База даних MongoDB зберігає та обробляє дані, серверна частина на Node.js з фреймворком Express керує маршрутизацією та логікою роботи з даними, а клієнтська частина на основі EJS і CSS забезпечує зручний інтерфейс користувача. Завдяки такій модульній структурі функціональність системи легко підтримується та розширюється, що робить її універсальним рішенням для інформаційно-пошукових завдань.

Нижче наведено інтерфейсну частину програми, яка є важливим елементом взаємодії користувача із системою. На рисунках 3.2 – 3.6 представлено сторінку, що демонструє головний інтерфейс системи – сторінку «Home». Програмна реалізація цієї сторінки наведена в додатках А.1 та А.2.

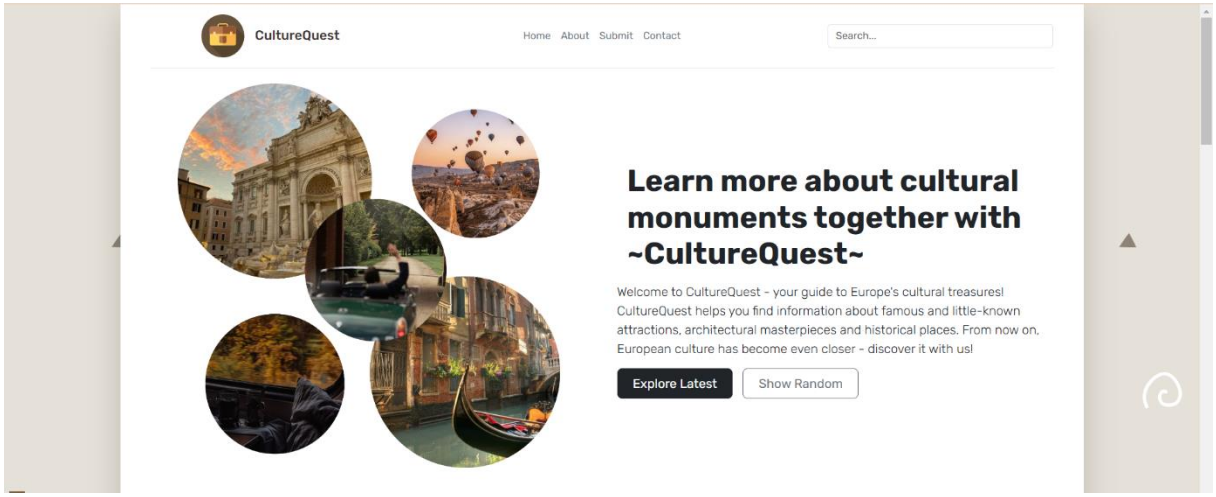


Рисунок 3.2 – Інтерфейс сторінки «Home» (початок)

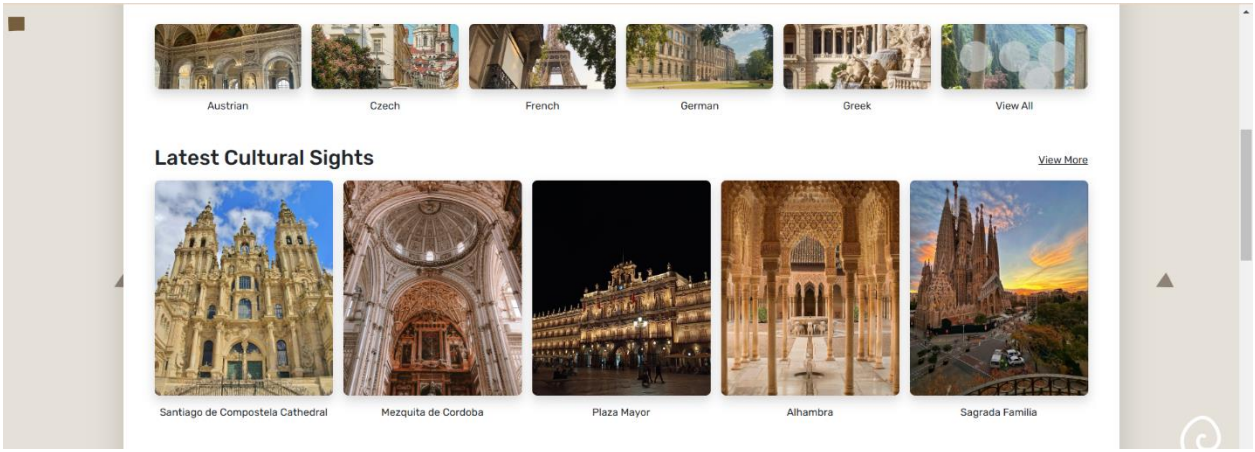


Рисунок 3.3 – Інтерфейс сторінки «Home» (скрол вниз)

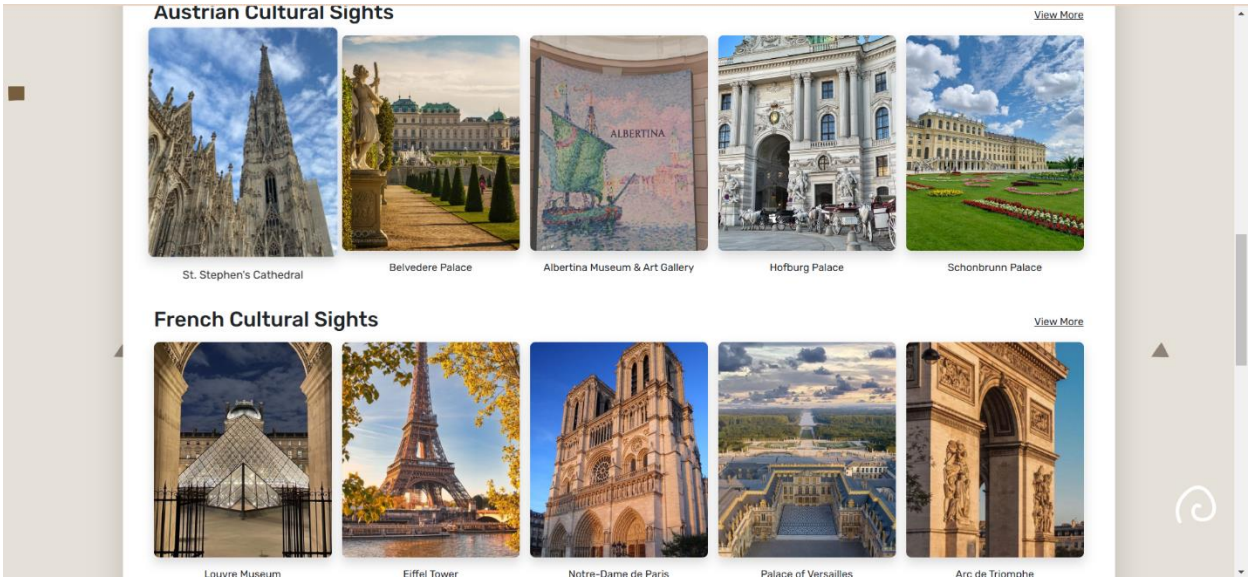


Рисунок 3.4 – Інтерфейс сторінки «Home» (середина)

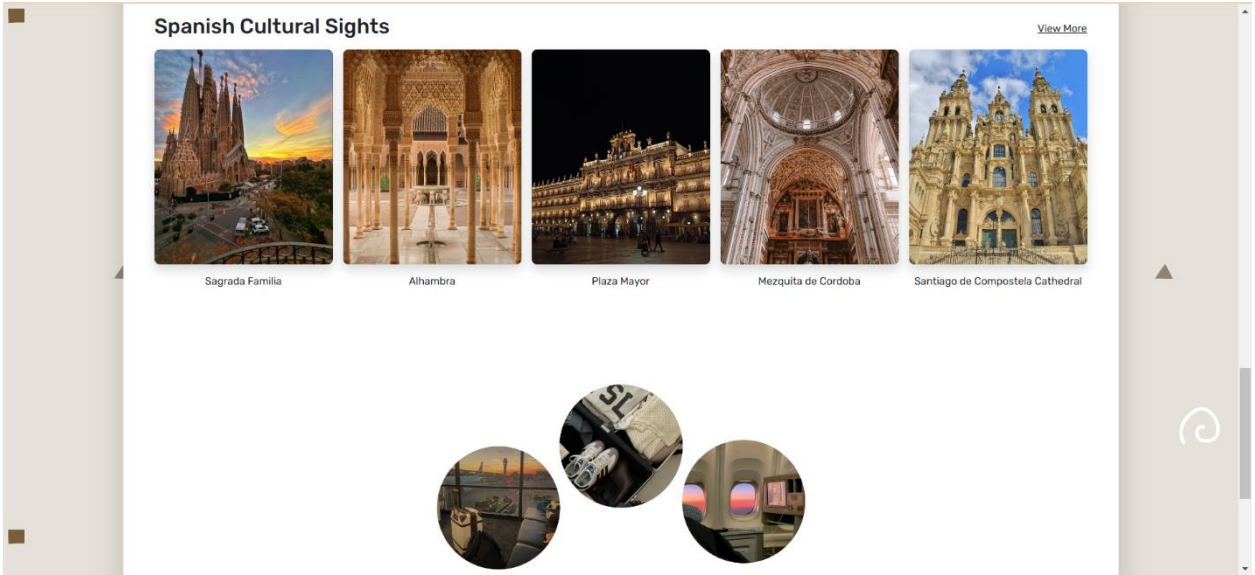


Рисунок 3.5 – Інтерфейс сторінки «Home» (скрол вниз)

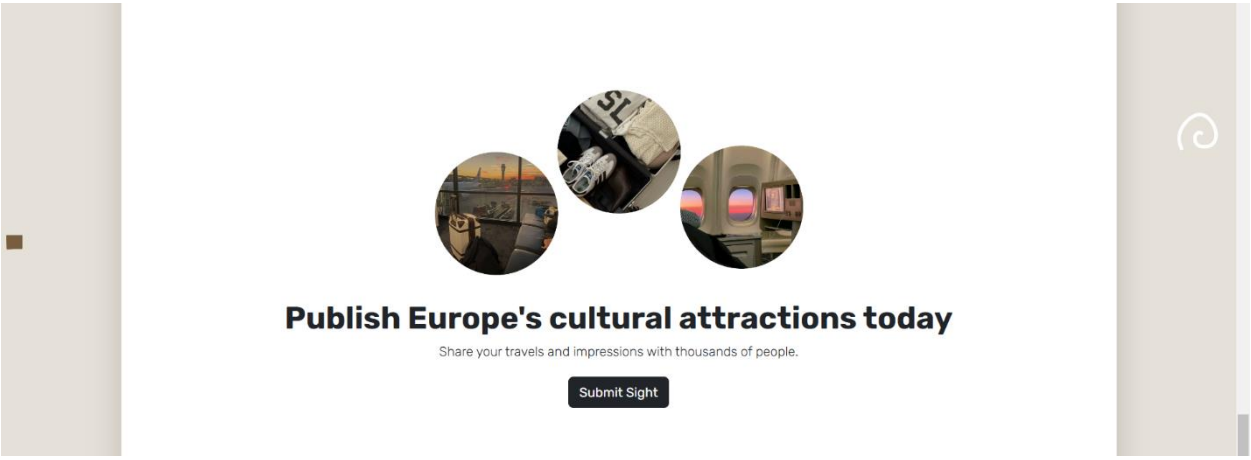


Рисунок 3.6 – Інтерфейс сторінки «Home» (кінець сторінки)

Далі переходимо до сторінки «About», на якій розкривається ідея створення «CultureQuest», її мета та роль у вивченні культурної спадщини. Тут користувач може дізнатися більше про натхнення автора та завдання проєкту, а також отримати контактну інформацію для подальшої взаємодії (див. рис. 3.7).

Програмна реалізація цієї сторінки наведена в додатках А.3 та А.4.

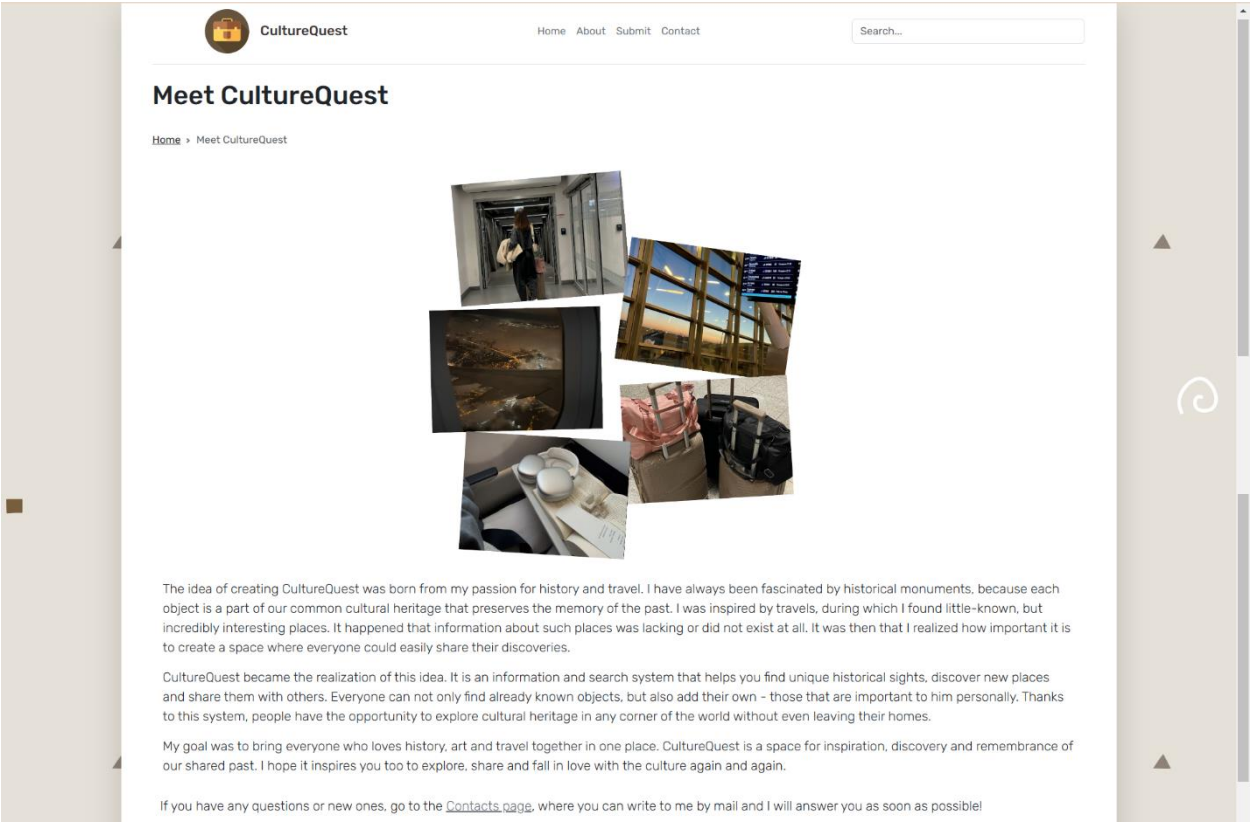


Рисунок 3.7 – Інтерфейс сторінки «About»

Після сторінки «About» наступною є сторінка «Contact», яка служить для зворотного зв’язку між користувачами та адміністрацією системи. На цій сторінці розміщена контактна інформація для зв’язку. Користувачі можуть звертатися з питаннями, пропозиціями чи зауваженнями щодо роботи системи. Це дозволяє підтримувати активний зв’язок із користувачами та забезпечити швидке реагування на їхні запити або побажання.

На рисунку 3.8 наведено інтерфейс сторінки «Contact» інформаційно-пошукової системи «CultureQuest».

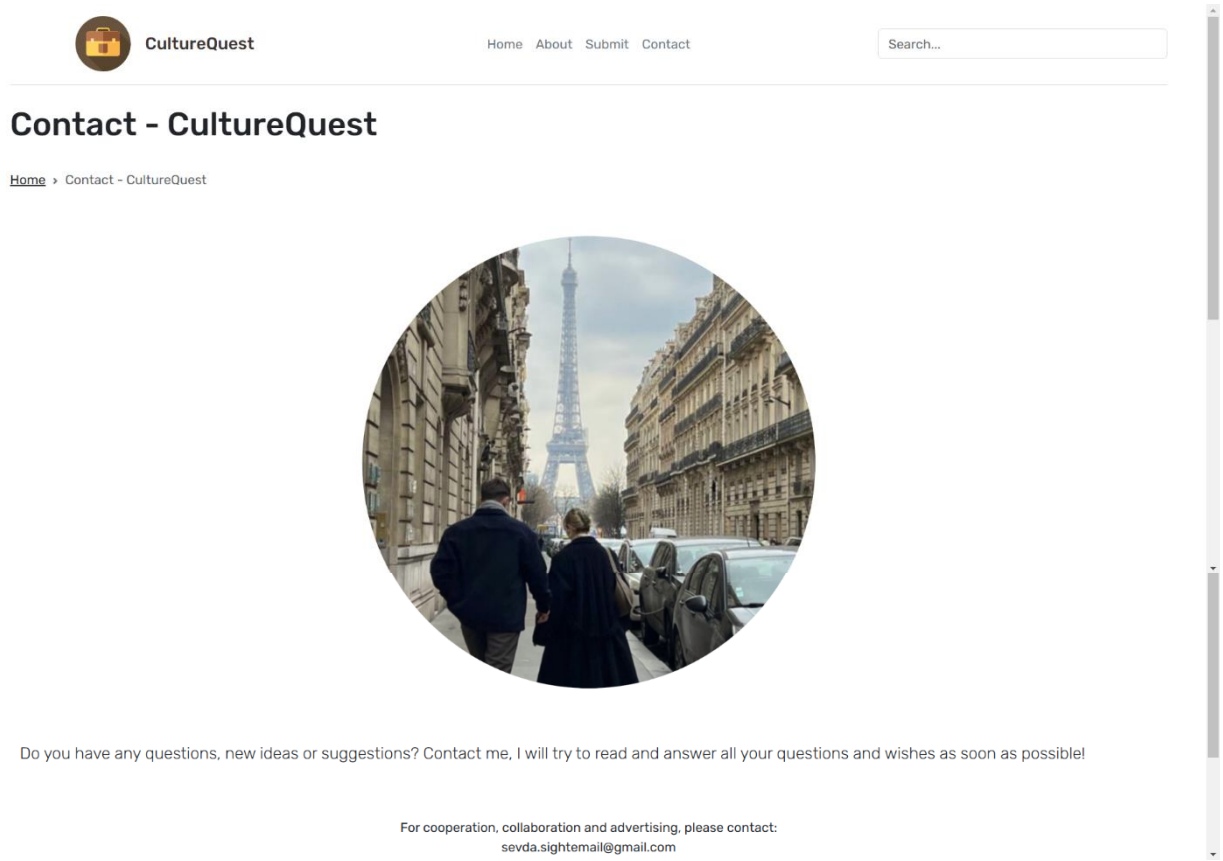


Рисунок 3.8 – Інтерфейс сторінки «Contact»

Далі переходимо до сторінки «Submit» (див. рис. 3.9 – 3.10), яка дозволяє користувачам додавати свої власні історії та враження від культурних пам’яток. Тут користувачі можуть заповнити форму для подачі своїх знахідок, щоб поділитися ними з іншими учасниками спільноти CultureQuest. Цей розділ дає можливість кожному внести свій вклад у збереження та популяризацію культурної спадщини, відкриваючи нові, захоплюючі місця для інших.

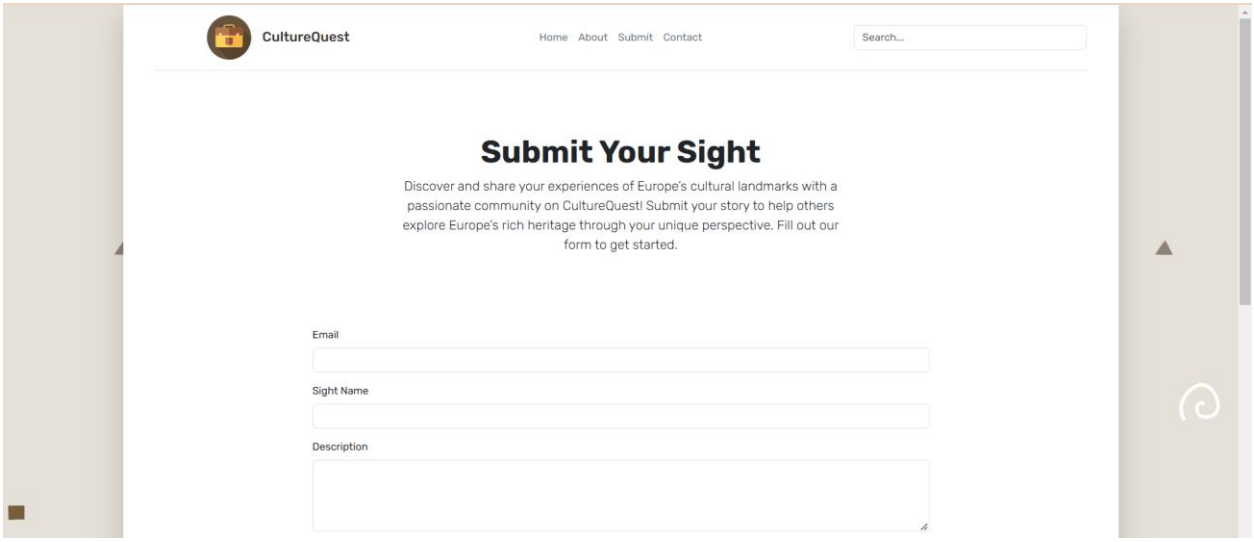


Рисунок 3.9 – Інтерфейс сторінки «Submit» (початок)

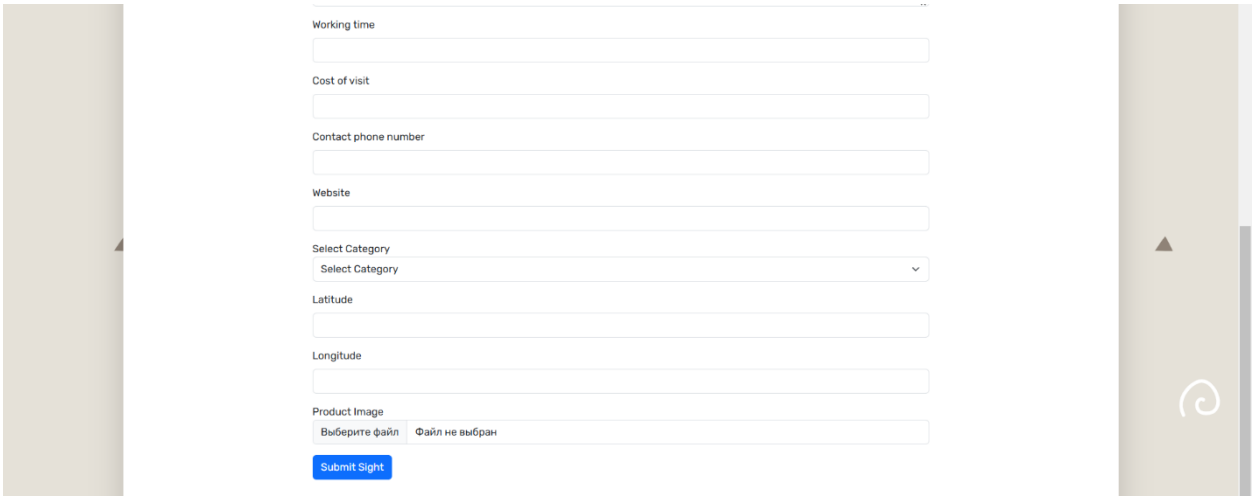


Рисунок 3.10 – Інтерфейс сторінки «Submit» (кінець сторінки)

Програмна реалізація сторінки «Submit» наведена в додатках А.5 та А.7.

Після сторінки «Submit» переходимо до сторінки «Explore Latest», яка надає користувачам можливість переглядати нові культурні об’єкти та пам’ятки, додані до системи. Тут можна знайти останні дописи інших користувачів і дізнатися про найбільш актуальні та цікаві місця, які ще не були Вам відомі. Це чудова можливість для тих, хто шукає нові та унікальні місця для відвідування.

На рисунку 3.11 наведено інтерфейс сторінки «Explore Latest» інформаційно пошукової системи «CultureQuest».

Програмна реалізація сторінки «Explore Latest» наведена в додатку А.8.

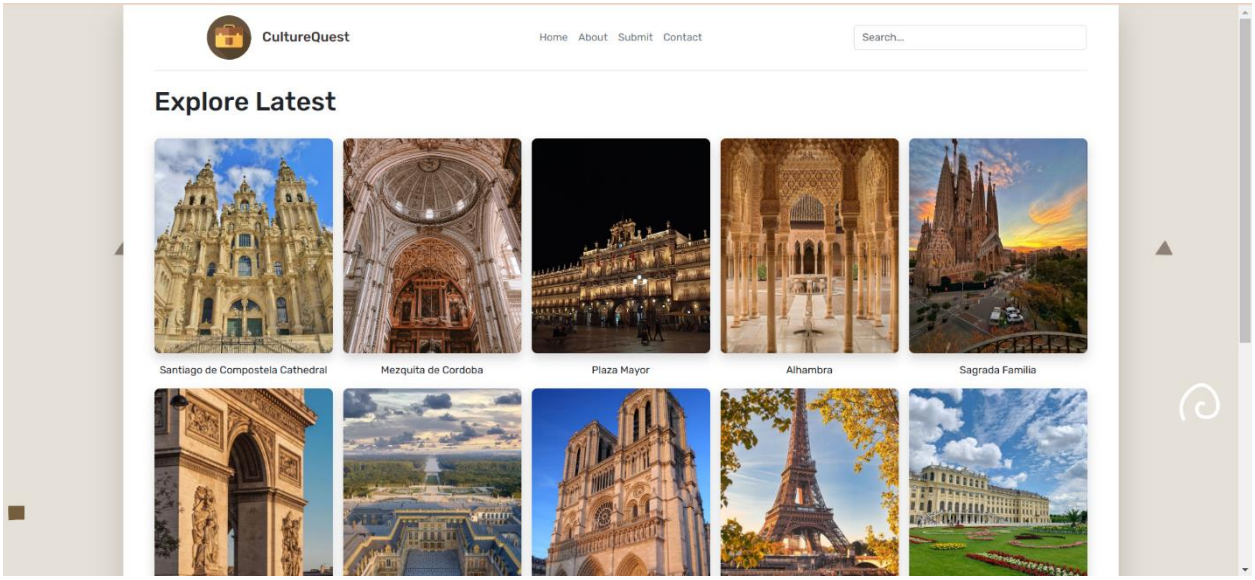


Рисунок 3.11 – Інтерфейс сторінки «Explore Latest»

Далі користувачі можуть перейти до сторінки «Explore Random», де вони мають змогу випадковим чином відкривати нові культурні пам’ятки та історичні об’єкти. Ця сторінка дозволяє користувачам знайти цікаві місця без конкретної мети, додаючи елемент несподіванки та вражень від криттів (див. рис. 3.12). Програмна реалізація сторінки «Explore Random» наведена в додатках А.9 та А.10.

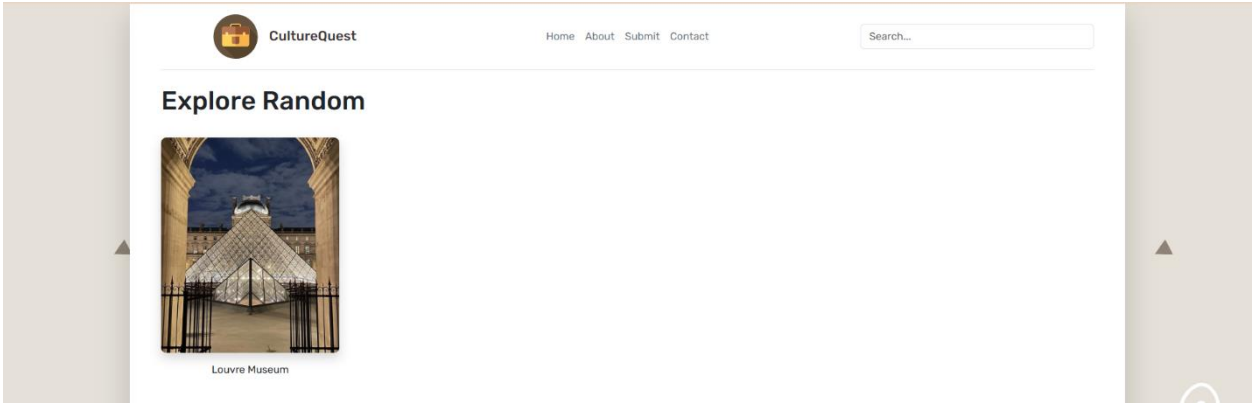


Рисунок 3.12 – Інтерфейс сторінки «Explore Random»

Файлова структура є важливим елементом організації проєкту, оскільки вона визначає, як взаємодіють системні компоненти та як код поділяється на різні модулі. У даному випадку, проєкт «CultureQuest» має таку структуру (див. рис. 3.13).

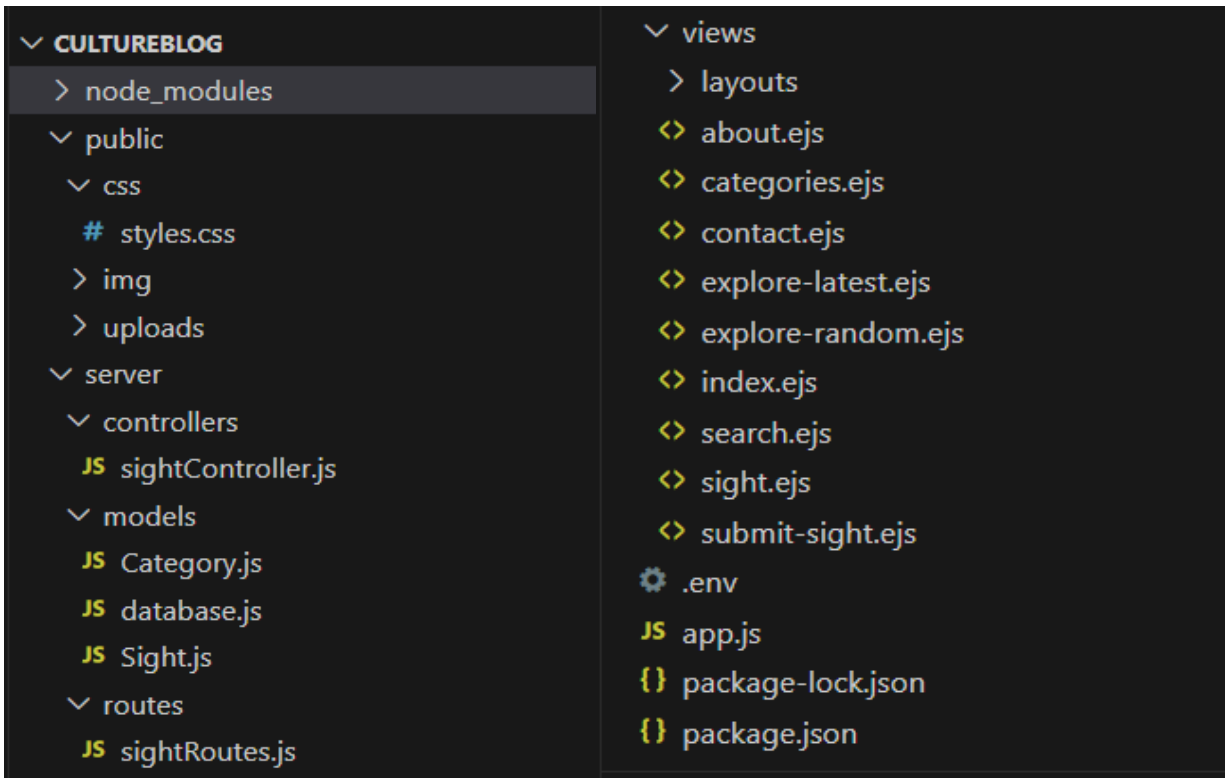


Рисунок 3.13 – Структура файлів проєкту

Основні директорії проєкту включають папки node_modules, public та server. Директорія node_modules містить усі залежності проєкту, які були встановлені за допомогою менеджера пакетів npm. Це основна папка, в якій зберігаються сторонні бібліотеки, необхідні для функціонування додатку, такі як Express, Mongoose та інші.

У папці public зберігаються статичні ресурси проєкту, такі як стилі CSS у файлі styles.css, зображення в папці img, а також завантажені файли в папці uploads. Це важливо, оскільки саме там зберігаються всі статичні файли, до яких клієнт може отримати доступ.

Папка server зберігає в собі важливі серверні файли, включаючи підкатегорії controllers, models, routes, які містять логіку обробки запитів, схеми моделей бази даних і маршрути для роботи з клієнтськими запитами. Файл database.js містить налаштування для підключення до бази даних MongoDB через Mongoose, що дозволяє зберігати та обробляти дані в системі.

Далі, в папці views містяться файли шаблонів для генерації HTML сторінок за допомогою EJS. Ці файли включають main.ejs, about.ejs,

categories.ejs, contact.ejs, explore-latest.ejs, explore-random.ejs, а також спеціальні шаблони для індексації, пошуку та подачі даних, такі як index.ejs, search.ejs та submit-sight.ejs. Вони відповідають за відображення інтерфейсу користувача.

Окрім того, важливі конфігураційні файли включають .env, де зберігаються конфіденційні налаштування, такі як URI підключення до MongoDB, а також файли package.json та package-lock.json, які визначають залежності проєкту та налаштування для npm.

Структура файлів організована таким чином, щоб забезпечити ефективну взаємодію всіх частин програми, спрощуючи її підтримку та розвиток. Кожен компонент проєкту має свою чітко визначену роль у загальній архітектурі, що дозволяє підтримувати високий рівень організації та зручності для подальшої роботи.

3.5 Тест-кейси

Тестування є невід’ємною частиною розробки програмного забезпечення, яке дозволяє перевірити коректність роботи системи, забезпечити її відповідність вимогам і виявити можливі помилки. Одним із основних інструментів тестування є тест-кейси. Тест-кейс – це структурований опис тесту, який містить чіткий набір умов, дій і очікуваних результатів, що необхідні для перевірки певного аспекту функціональності програмного продукту. Зосередимо увагу на ручному тестуванні інформаційно-пошукової системи «CultureQuest», тобто тестування будуть проводитись без використання автоматизованих інструментів, шляхом виконання тестів вручну [17].

Перевіримо функцію виведення випадкового об’єкта при натисканні кнопки «Show Random», провівши ручне тестування функціоналу, щоб переконатися, що при кожному натисканні кнопки відображається різний випадковий об’єкт.

Дії під час тестування:

– натиснути кнопку «Show Random»;
– звернути увагу на виведений об’єкт;
– повторити дію натискання на кнопку;
– порівняти результати кожного натискання, щоб переконатися, що об’єкти змінюються та не повторюються.

Очікуваний результат:

– після кожного натискання кнопки «Show Random» на екрані відображається новий випадковий об’єкт із бази даних;
– об’єкти не повторюються під час послідовного тестування (або повторення є малоймовірним).

Фактичний результат наведено на рисунках 3.14 та 3.15.

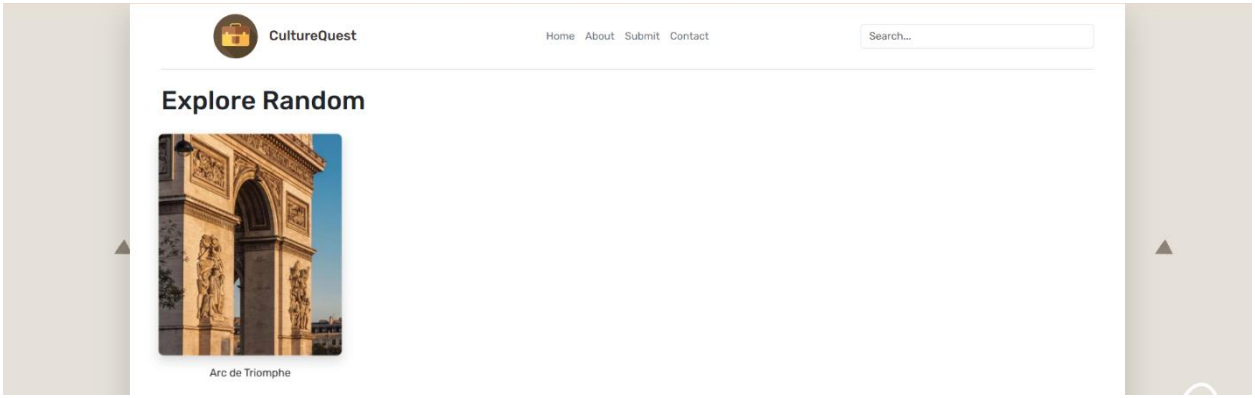


Рисунок 3.14 – Результат тестування після натискання кнопки «Show Random»

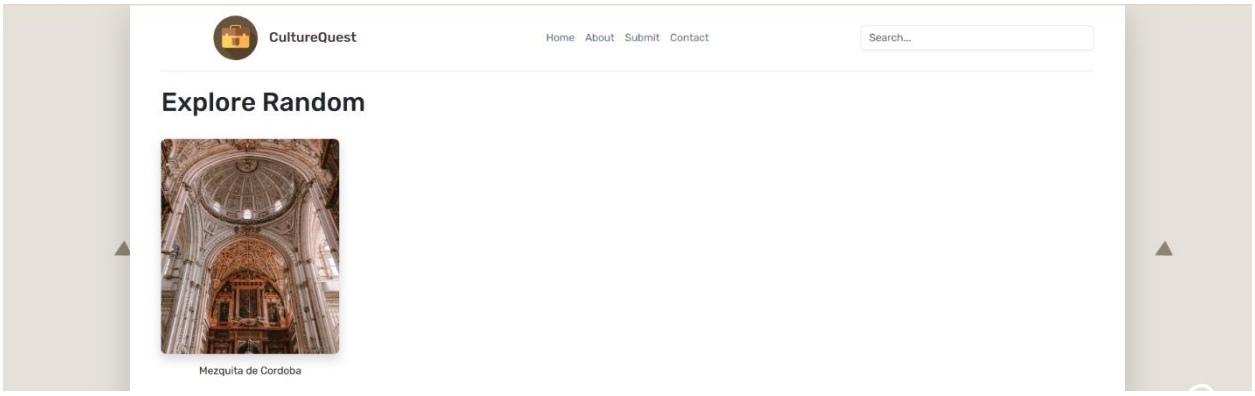


Рисунок 3.15 – Результат тестування після оновлення сторінки

Результати тестування показали, що функціонал кнопки «Show Random» працює коректно. При кожному натисканні системою успішно виводиться випадковий об’єкт із бази даних, що відповідає очікуваному результату. Критичних помилок або збоїв не виявлено, поведінка системи повністю відповідає заявленій функціональності.

Переходимо до перевірки коректності пошуку культурної пам’ятки. Для цього проведемо ручне тестування, спрямоване на перевірку пошуку як за точною назвою, так і за ключовим словом. Це дозволить оцінити, наскільки ефективно система знаходить потрібні об’єкти.

Для проведення тестування пошуку культурних пам’яток у системі «CultureQuest» виконувалися наступні дії, розділені на два сценарії: пошук за точною назвою та пошук за ключовим словом.

Сценарій 1: Пошук за точною назвою:

– перейти до пошукового поля у верхній частині сторінки;
– ввести точну назву об’єкта, наприклад, «Sagrada Familia»;
– виконати пошук;
– дочекатися відображення результатів пошуку на сторінці;
– перевірити, чи є у результатах пошуку лише один об’єкт, який точно відповідає введеній назві.

Очікуваний результат для Сценарію 1: система має повернути лише один результат пошуку, який відповідає введеній точній назві.

Фактичний результат Сценарію 1 наведено на рисунку 3.16.

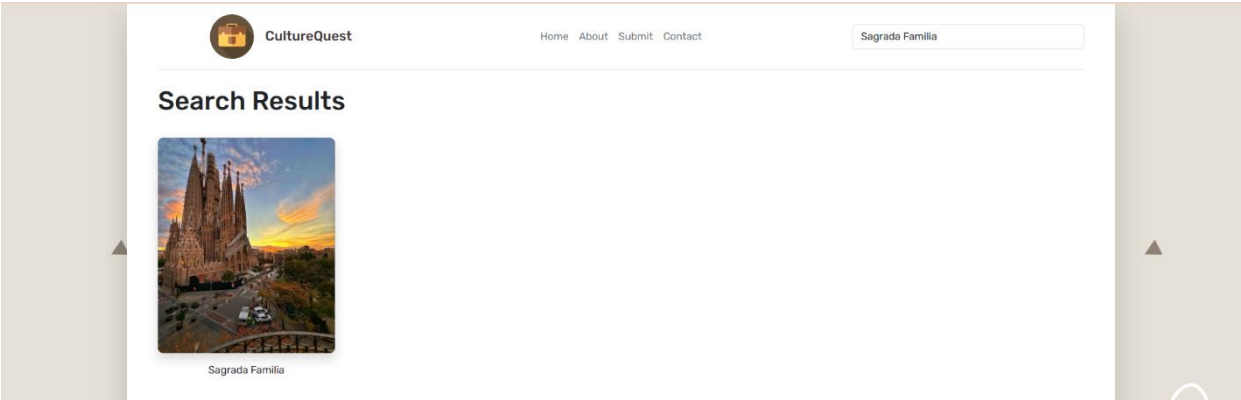


Рисунок 3.16 – Результат пошуку за точною назвою

Сценарій 2: Пошук за ключовим словом:

– у поле пошуку ввести ключове слово, наприклад, «tower»;
– знову виконати пошук;
– дочекатися завантаження результатів пошуку на сторінці;
– перевірити, чи відображаються всі об’єкти, які містять це ключове слово у своїй назві чи описі.

Очікуваний результат для Сценарію 2: система має повернути список результатів, які відповідають введеному ключовому слову. Усі об’єкти, пов’язані із ключовим словом (наприклад, вежа для слова «tower»), мають бути коректно відображені. У списку не має бути нерелевантних об’єктів.

Фактичний результат Сценарію 1 наведено на рисунку 3.17.

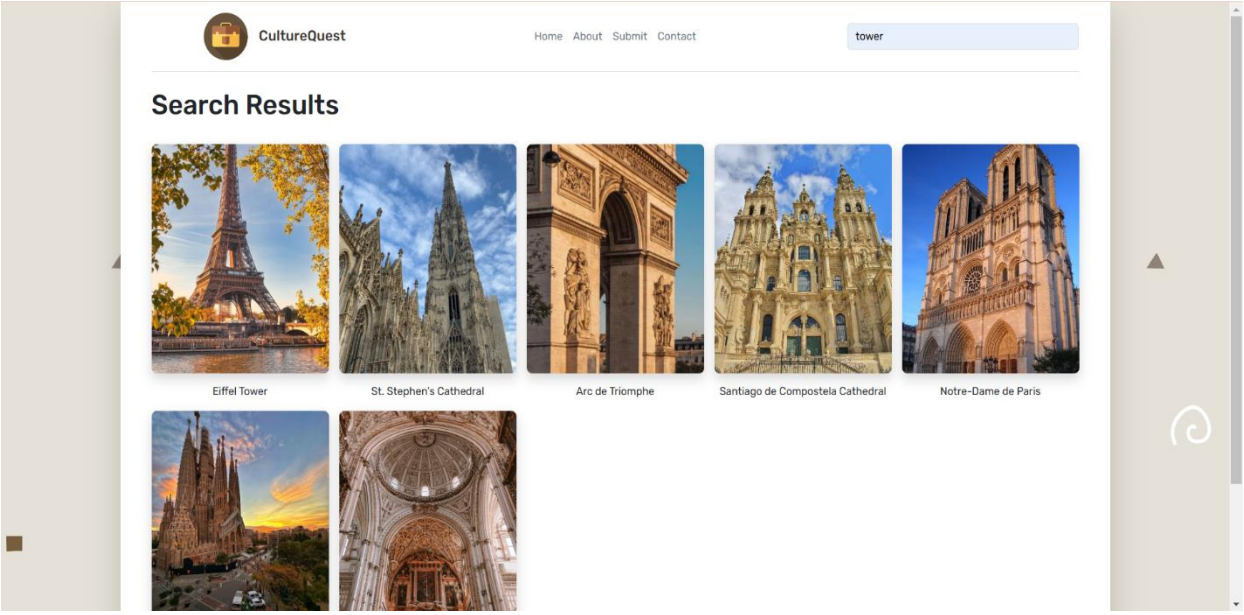


Рисунок 3.17 – Результат пошуку за ключовим словом

В обох сценаріях система «CultureQuest» успішно виконала поставлені завдання. У першому випадку пошук за точною назвою повернув єдиний релевантний об’єкт, а у другому — система коректно відобразила список об’єктів, що відповідають ключовому слову. Тестування підтвердило, що функціонал пошуку працює відповідно до вимог, забезпечуючи зручність і точність для користувачів.

Далі перевіримо функціонал додавання нової культурної пам’ятки в інформаційно-пошукову систему «CultureQuest». На сторінці «Submit Your Sight» користувач може заповнити форму, яка містить поля для введення даних про пам’ятку. Після заповнення форми натискання кнопки «Submit Sight» повинно зберегти дані в базі даних і зробити їх доступними для перегляду на відповідних сторінках системи.

Дії під час тестування:

– перейти на сторінку «Submit Your Sight», де відображається форма для введення інформації про новий об’єкт;
– заповнити поле «Email», ввівши дійсну електронну адресу;
– у полі «Sight Name» ввказати назву нової пам’ятки;
– у полі «Description» ввести опис пам’ятки;
– у поле «Working time» зазначте години роботи;
– заповнити поле «Cost of visit», вказавши вартість візиту;
– у полі «Contact phone number» ввести номер телефону;
– у полі «Website» вказати URL-адресу офіційного сайту пам’ятки;
– у випадаючому меню «Select Category» оберати відповідну категорію;
– у полях «Latitude» та «Longitude» вказати географічні координати пам’ятки;
– завантажити зображення пам’ятки у полі «Sight Image»;
– натиснути кнопку «Submit Sight», щоб відправити дані.

Очікуваний результат:

– після натискання кнопки «Submit Sight» система повинна зберегти всі введені дані в базі даних MongoDB;
– користувач отримує повідомлення про успішне додавання пам’ятки;
– доданий об’єкт стає доступним для пошуку через функціонал «Explore» або перегляд у відповідних категоріях;
– усі дані, включаючи назву, опис, зображення, контактну інформацію та геолокацію, коректно відображаються на сторінці пам’ятки.

Фактичний результат наведено на рисунках 3.18 та 3.19.

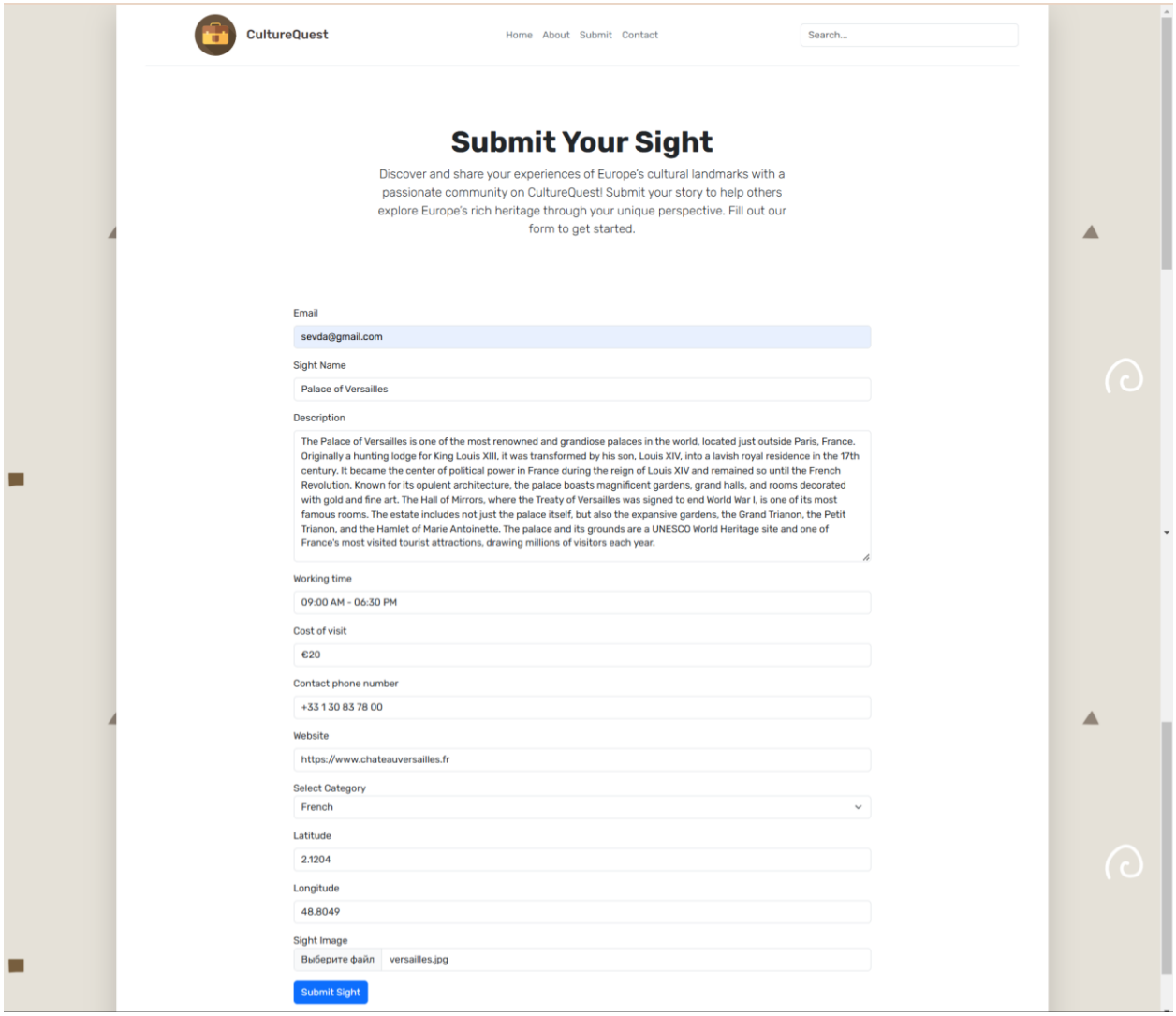


Рисунок 3.18 – Тестування додавання об’єкту до ІПС «CultureQuest»

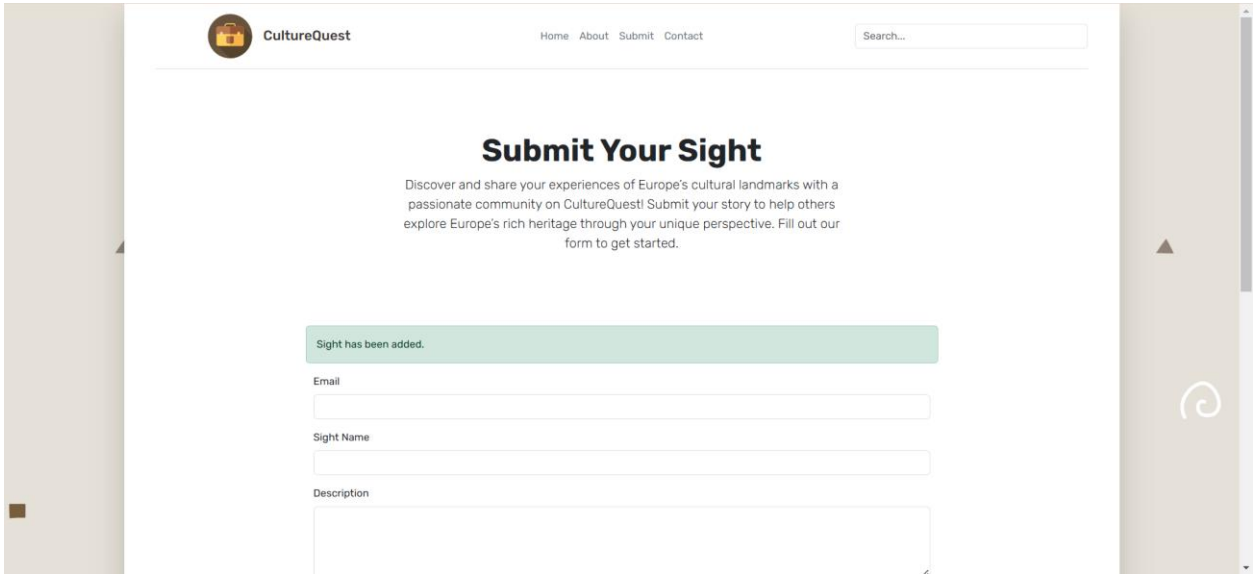


Рисунок 3.19 – Результат тестування додавання об’єкту до ІПС «CultureQuest»

Після заповнення всіх відповідних полів, та натискання кнопки «Submit Sight», користувачу відображається повідомлення про успішне додавання культурної пам’ятки до системи. Для перевірки коректності додавання нового об’єкта, необхідно перевірити, чи з’явився цей об’єкт у базі даних «Sights» (див. рис. 3.20).

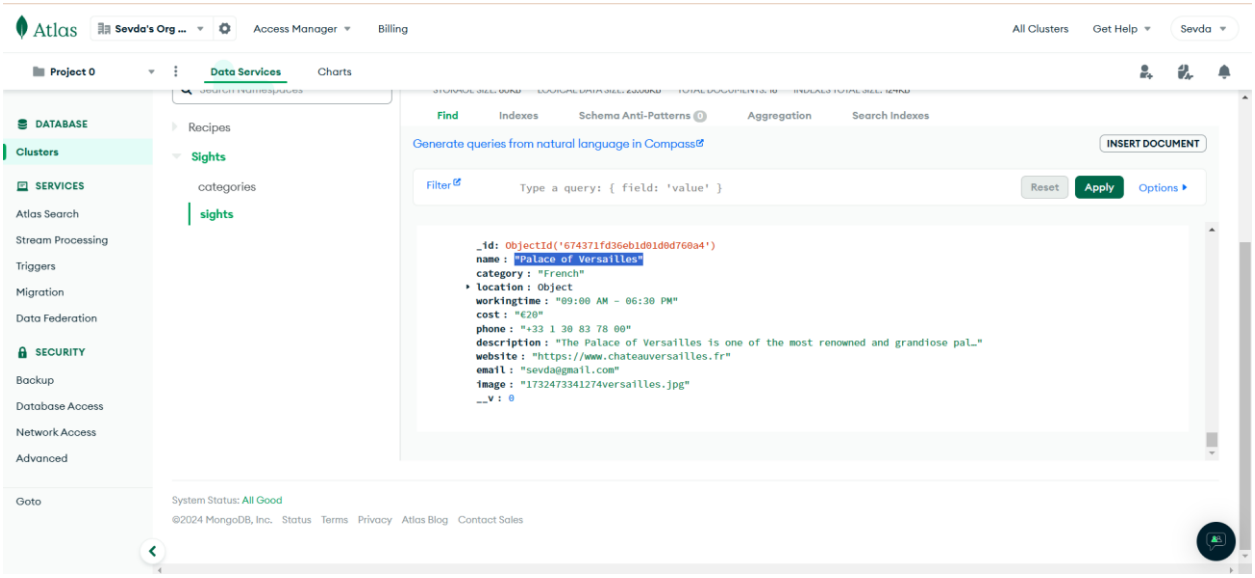


Рисунок 3.20 – Результат тестування додавання об’єкту до бази даних «Sights»

Далі перевіримо чи з’явився новий об’єкт на сторінці «Home» у розділі «Latest Cultural Sights» (див. рис. 3.21).

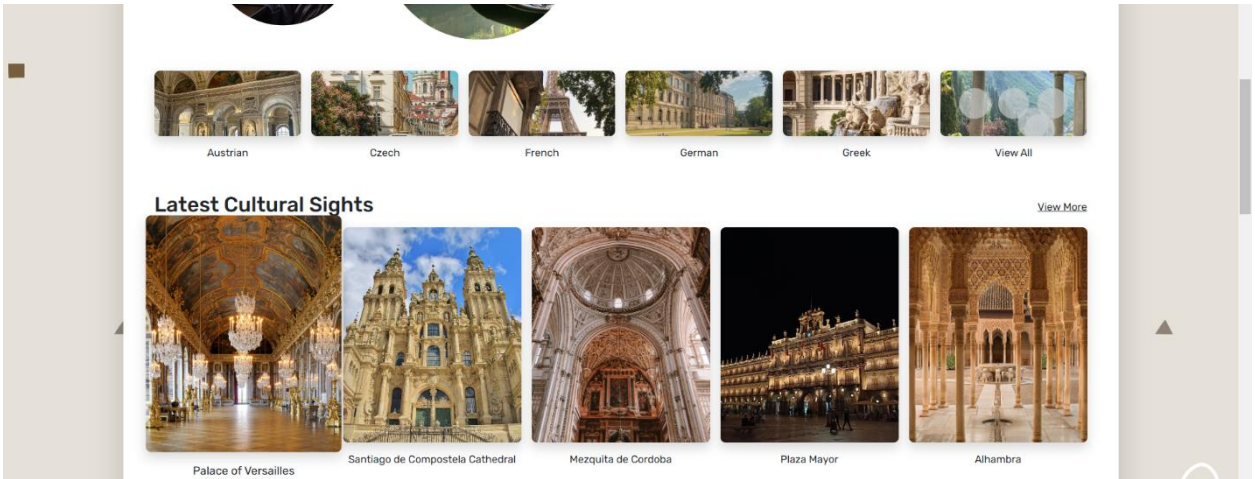


Рисунок 3.21 – Результат тестування додавання об’єкту на сторінці «Home»

Після цього переходимо до розділу «Explore Latest» та переглядаємо останні додані культурні пам’ятки до інформаційної системи (див. рис. 3.22).

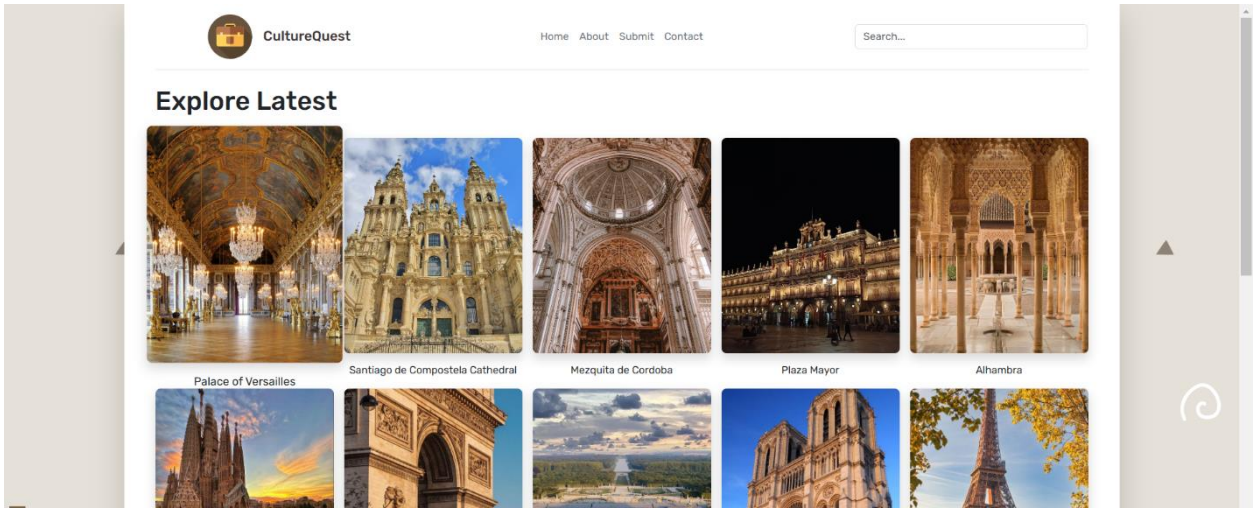


Рисунок 3.22 – Результат тестування додавання об’єкту на сторінці «Explore Latest»

Також необхідно обов’язково перевірити правильне відображення об’єкта «Palace of Versailles» на відповідній сторінці (див. рис. 3.23).

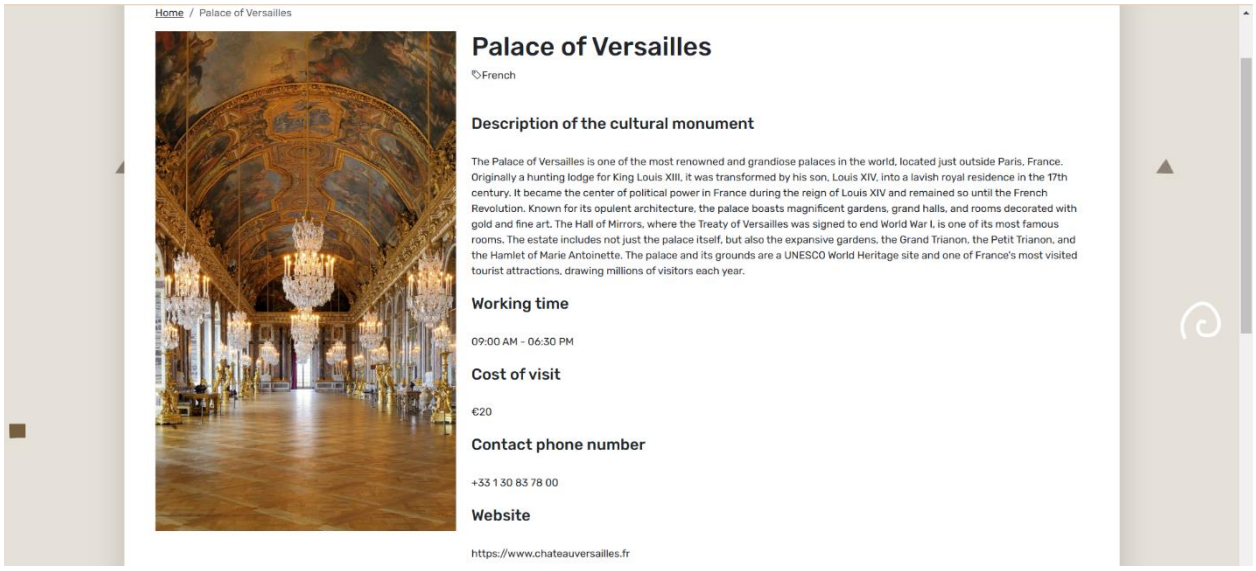


Рисунок 3.23 – Результат тестування коректного відображення культурної пам’ятки

Після проведеного тестування можна зробити висновок, що очікуваний результат збігається з фактичним, тобто культурна пам’ятка була успішно

додана в систему. Всі поля форми були коректно заповнені, і після натискання кнопки «Submit Sight» користувач отримав підтвердження про успішне додавання об’єкта. При перевірці бази даних було підтверджено, що нова пам’ятка з’явилася в системі. Крім того, об’єкт «Palace of Versailles» було успішно відображено на відповідних сторінках сайту, що підтверджує коректність роботи функціоналу додавання культурних пам’яток. Всі дані, включаючи назву, опис, координати та інші параметри, були правильно збережені та відображені відповідно до вимог системи.

3.6 Тестування та аналіз результатів тестування

Тестування та оцінка результатів роботи інформаційно-пошукової системи «CultureQuest».

Опишемо послідовність виконання тестування.

Перевірка на завантаження системи: спочатку перевірити правильність завантаження сайту, в тому числі перевірити швидкість завантаження та відображення всіх елементів на головній сторінці без помилок.

Перевірка адаптивності інтерфейсу: перевірити коректність відображення сайту на різних пристроях і браузерах. Це включає забезпечення того, щоб інтерфейс адаптувався до різних розмірів екрана, щоб елементи не деформувалися та щоб вони були доступні для взаємодії. Приділяється особлива увага версіям для мобільних пристроїв і планшетів.

Перевірка функціональності кнопки «Explore Latest»: перевіряється, чи коректно працює кнопка «Explore Latest». Перевірка включає натискання кнопки та перевірку того, чи останні додані культурні пам’ятки з бази даних відображаються в тому порядку, в якому вони були додані.

Перевірка функціональності кнопки «Show Random»: перевіряється, чи коректно працює кнопка «Show Random». Тестування включає натискання на кнопку і оновлення сторінки для того, щоб переконатися, що кожного разу виводиться новий випадковий об’єкт із бази даних.

Перевірка пошукової системи: перевіра коректності пошуку культурних пам’яток за точними назвами або ключовими словами. Для цього вводяться різні запити, і перевіряється, чи система повертає відповідні результати без помилок.

Перевірка додавання нових об’єктів до системи: тестується функціональність додавання нових пам’яток через форму «Submit Your Sight». Він перевіряє, чи всі вхідні дані обробляються правильно, чи новий об’єкт відображається на сторінці та чи всі поля правильно зберігаються в базі даних.

Перевірка на валідацію даних: перевіряється, чи здійснюється валідація введених користувачем даних, таких як адреси, телефонні номери та електронні адреси, на відповідність вимогам безпеки та коректності.

Перевірка на стресостійкість системи: випробовується стабільність роботи системи за умов високих навантажень. Це включає перевірку відновлення роботи після перезавантаження сервера та збереження даних в базі.

В таблиці 3.1 наведено перелік етапів тестування інформаційно-пошукової системи «CultureQuest».

Таблиця 3.1 – Етапи тестування ІПС «CultureQuest»

Перевірка	ОС: **Window 10** Браузер: **Google Chrome 116.0.5845.111**	ОС: **Window 10** Браузер: **Opera 95.0.4635.37**	ОС: **Window 10** Браузер: **Microsoft Edge 116.0.1938.81**	ОС: **MacOS** Браузер: **Safari 16.6**
Завантаження системи	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено
Адаптивність інтерфейсу	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено

Продовження таблиці 3.1

Перевірка	ОС: **Window 10** Браузер: **Google Chrome 116.0.5845.111**	ОС: **Window 10** Браузер: **Opera 95.0.4635.37**	ОС: **Window 10** Браузер: **Microsoft Edge 116.0.1938.81**	ОС: **MacOS** Браузер: **Safari 16.6**
Функціональність кнопки «Explore Latest»	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено
Функціональність кнопки «Show Random»	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено
Пошукова система	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено
Додавання нових об’єктів	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено
Валідація даних	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено
Стресостійкість системи	успішно пройдено	успішно пройдено	успішно пройдено	успішно пройдено

Результати тестування інформаційно-пошукової системи «CultureQuest» підтверджують її стабільну роботу на різних операційних системах (Windows 10, MacOS) та браузерах (Google Chrome, Opera, Microsoft Edge, Safari). Завантаження та адаптивність інтерфейсу системи були успішно протестовані на всіх платформах, що свідчить про зручність використання системи незалежно від пристрою. Функціональні кнопки «Explore Latest» та «Show Random» працювали коректно в усіх середовищах, тим самим підтвердили свою інтерактивність і стабільність. Пошукова система

продемонструвала правильну обробку запитів, а процес додавання нових об’єктів, включаючи валідацію даних, також пройшов без зауважень. Стресостійкість системи підтверджує її здатність витримувати підвищені навантаження, що є важливим фактором її надійності. У підсумку, можна побачити, що «CultureQuest» готова до використання користувачами, забезпечуючи стабільну роботу, зручність і комфорт незалежно від платформи чи пристрою.

ВИСНОВКИ

При розробці інформаційно-пошукової системи CulturalQuest було проведено аналіз існуючих аналогів для визначення основних вимог до функціональності та дизайну системи. На основі цього аналізу було сформульовано технічне завдання, яке включало перелік функціональних і нефункціональних характеристик, зокрема забезпечення високої продуктивності, безпеки, адаптивності та зручності використання. Основною метою цієї системи є надання користувачам інструменту для дослідження культурної спадщини, що поєднує в собі сучасні технології, адаптивний дизайн та комфортний інтерфейс.

Для реалізації проєкту було обрано мову програмування JavaScript у поєднанні з HTML та CSS, що дозволило створити сучасний, адаптивний і інтерактивний вебінтерфейс. Основним середовищем розробки виступив Visual Studio Code, який забезпечив гнучкість у роботі завдяки численним плагінам і широким можливостям інтеграції з іншими інструментами. Для пришвидшення розробки використано фреймворк Express.js, який забезпечив зручне налаштування серверної логіки, створення маршрутів для обробки HTTP-запитів, а також інтеграцію допоміжних бібліотек.

Під час виконання проєкту, були використані певні технології, а саме:

– MongoDB – документоорієнтована база даних, яка дозволила організувати зберігання інформації про культурні пам’ятки у гнучкому форматі JSON;
– Mongoose – інструмент для створення моделей даних, забезпечення їх валідації та інтеграції з базою даних;
– EJS – шаблонізатор, який дозволив створювати динамічні HTML-сторінки, які взаємодіють із сервером;
– Bootstrap – фреймворк для адаптивного дизайну, що забезпечив естетичний і зручний інтерфейс для різних пристроїв.

Функціонал системи включає:

– пошук і перегляд інформації про пам’ятки;
– додавання нових об’єктів до бази даних через відповідну форму;
– відображення останніх і випадкових пам’яток;
– зручну навігацію між розділами системи («Home», «About», «Submit», «Contact»).

Результати тестування підтвердили відповідність розробленої системи усім заявленим вимогам. Було перевірено стабільність роботи серверної частини, коректність пошукових алгоритмів, а також відображення інтерфейсу на різних пристроях.

Архітектура системи спроєктована таким чином, що вона залишається відкритою для модифікацій і розширень у майбутньому, зокрема розробку мобільних додатків для платформ iOS та Android, що забезпечить ще ширший доступ до функціоналу системи «CultureQuest» та підвищить її зручність для користувачів.

Таким чином, розроблена інформаційно-пошукова система «CultureQuest» успішно досягла поставлених цілей. Вона забезпечує зручний доступ до інформації про культурні пам’ятки Європи, має привабливий дизайн і стабільну функціональність, що робить її корисним інструментом для широкого кола користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Advantages of Express.js. URL: https://binariks.com/blog/express-js-mobile-app-development-pros-cons-developers/ (дата звернення: 28.05.2024).

2. Advantages of JavaScript. URL: https://softjourn.com/insights/the-advantages-and-disadvantages-of-javascript (дата звернення: 28.05.2024).

3. Atlas Obscura. URL: https://www.atlasobscura.com/ (дата звернення: 27.05.2024).

4. Bootstrap. URL: https://getbootstrap.com/ (дата звернення: 23.05.2024).

5. Culture Trip. URL: https://theculturetrip.com/ (дата звернення: 27.05.2024).

6. Diacritical marks. URL: https://www.wikidata.uk-ua.nina.az/%D0%94%D1%96%D0%B0%D0%BA%D1%82%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%BD%D1%96_%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8.html (дата звернення: 15.09.2024).

7. EJS. URL: https://ejs.co/ (дата звернення: 23.05.2024).

8. Express.js. URL: https://expressjs.com/ (дата звернення: 23.05.2024).

9. IDEF0. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%9A%D0%BE%D0%BD%D0%B4%D1%96%D1%83%D1%81%202%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%B2%D0%B0/page9.html (дата звернення: 27.05.2024).

10. IDEF1X. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%9A%D0%BE%D0%BD%D0%B4%D1%96%D1%83%D1%81%202%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%B2%D0%B0/page13.html (дата звернення: 27.05.2024).

11. Introduction to Mongoose for MongoDBA. URL: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/ (дата звернення: 28.05.2024).

12. MongoDB. URL: https://en.wikipedia.org/wiki/MongoDB (дата звернення: 23.05.2024).

13. Mongoose. URL: https://mongoosejs.com/ (дата звернення: 23.05.2024).

14. National Geographic. URL: https://www.nationalgeographic.com/ (дата звернення: 27.05.2024).

15. NFRs. URL: https://scaledagileframework.com/nonfunctional-requirements/#:~:text=Nonfunctional%20Requirements%20(NFRs)%20are%20intended,how%20well%20it%20does%20it (дата звернення: 27.05.2024).

16. Node.js. URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/ (дата звернення: 23.05.2024).

17. Test case. URL: https://dou.ua/forums/topic/46769/ (дата звернення: 01.11.2024).

18. UML. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 27.05.2024).

19. Use Case Diagram. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 27.05.2024).

20. Visual Studio Code. URL: https. URL://code.visualstudio.com/docs/editor/whyvscode (дата звернення: 15.09.2024).

21. Advantages of MongoDB. URL: https://www.mongodb.com/resources/compare/advantages-of-mongodb (дата звернення: 28.05.2024).

22. Advantages of Mongoose. URL: https://medium.com/@sunnysharif154/advantages-of-using-mongoose-b5a81680cb55 (дата звернення: 28.05.2024).

23. Analysis of tourism in Europe. URL: https://www.unian.ua/tourism/news/turizm-v-yevropi-vzhe-vidnovivsya-pislya-pandemiji-koronavirusu-12544629.html (дата звернення: 26.05.2024).

24. Combination of MongoDB and Mongoose. URL: https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/ (дата звернення: 28.05.2024).

ДОДАТОК А

Програмна реалізація

А.1 Частина коду, яка відповідає за вигляд сторінки «Home»

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title><%= typeof title != 'unefined' ? title: 'Culture Blog -
Made with Node.js' %></title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Rubik:ital,wght@0,300
..900;1,300..900&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="./css/styles.css">
</head>
<body>

    <div class="container-xxl px-md-5 bg-white shadow-lg">

        <header class="d-flex flex-wrap align-items-center justify-
content-center justify-content-md-between py-3 mb-4 border-bottom">
            <a href="/" class="d-flex align-items-center col-md-3
mb-2 mb-md-0 text-dark text-decoration-none">
                <img src="/img/logo.svg" width="229" height="68"
alt="CultureQuest - Made with Node.js">
                <span class="blog-title">CultureQuest</span>
            </a>

        <ul class="nav col-12 col-md-auto mb-2 justify-content-
center mb-md-0">
            <li><a href="/" class="nav-link px-2 link-
secondary">Home</a></li>
            <li><a href="/about" class="nav-link px-2 link-
secondary">About</a></li>
```

А.2 Частина коду, яка відповідає за вигляд сторінки «Home»

```
            <li><a href="/submit-sight" class="nav-link px-2 link-
secondary">Submit</a></li>
            <li><a href="/contact" class="nav-link px-2 link-
secondary">Contact</a></li>
        </ul>

        <div class="col-md-3 text-end">
            <form method="POST" action="/search">
                <input type="search" name="searchTerm" class="form-
control" placeholder="Search..." aria-Label="Search">
            </form>
        </div>

       </header>

        <main>
            <%- body -%>
        </main>

        <footer class="py-5">
            <p>Build by Sevda</p>
        </footer>

    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.b
undle.min.js"
    integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
</html>
```

А.3 Частина коду, яка відповідає за вигляд сторінки «About»

```
<h1 class="pb-4">Meet CultureQuest</h1>


<nav style="--bs-breadcrumb-divider:
url(&#34;data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='8' height='8'%3E%3Cpath
d='M2.5 0L1 1.5 3.5 4 1 6.5 2.5 8l4-4-4-4z'
fill='%236c757d'/%3E%3C/svg%3E&#34;);" aria-label="breadcrumb">
    <ol class="breadcrumb">
      <li class="breadcrumb-item"><a href="/">Home</a></li>
      <li class="breadcrumb-item active" aria-current="page">Meet
CultureQuest</li>
    </ol>
</nav>
```

А.4 Частина коду, яка відповідає за вигляд сторінки «About»

```
<body>
    <div class="row justify-content-center px-3 py-3 mu-3">
        <div class="col-8 mb-4">
            <img src="/img/about.png" width="607" height="510"
class="d-block mx-lg-auto img-fluid" loading="lazy"
alt="CultureQuest">
        </div>

        <p class="lead">
            The idea of creating CultureQuest was born from my
passion for history and travel. I have always been fascinated by
historical monuments, because each object is a part of our common
cultural heritage that preserves the memory of the past. I was
inspired by travels, during which I found little-known, but
incredibly interesting places. It happened that information about
such places was lacking or did not exist at all. It was then that I
realized how important it is to create a space where everyone could
easily share their discoveries.
        </p>
        <p class="lead">
            CultureQuest became the realization of this idea. It is
an information and search system that helps you find unique
historical sights, discover new places and share them with others.
Everyone can not only find already known objects, but also add their
own - those that are important to him personally. Thanks to this
system, people have the opportunity to explore cultural heritage in
any corner of the world without even leaving their homes.
        </p>
        <p class="lead">
            My goal was to bring everyone who loves history, art and
travel together in one place. CultureQuest is a space for
inspiration, discovery and remembrance of our shared past. I hope it
inspires you too to explore, share and fall in love with the culture
again and again.
        </p>
        <p class="lead g-3">
            If you have any questions or new ones, go to the <a
href="/contact" class="link-secondary">Contacts page</a>, where you
can write to me by mail and I will answer you as soon as possible!
        </p>
    </div>
</body>
```

А.5 Частина коду, яка відповідає за додавання об’єктів до ІПС

```
<div class="px-4 py-5 my-5 text-center">
    <h1 class="display-5 fw-bold">Submit Your Sight</h1>
    <div class="col-lg-6 mx-auto">
        <p class="lead">Discover and share your experiences of
Europe’s cultural landmarks with a passionate community on
CultureQuest! Submit your story to help others explore Europe’s rich
heritage through your unique perspective. Fill out our form to get
started.</p>
    </div>
</div>

<div class="row justify-content-center">
    <% if(infoSubmitObj != '') { %>
        <div class="col-8 alert alert-success" role="alert">
            <%= infoSubmitObj %>
        </div>
    <% } %>

    <% if(infoErrorsObj != '') { %>
        <div class="col-8 alert alert-danger" role="alert">
            <%= infoErrorsObj[0].message %>
        </div>
    <% } %>
    <div class="col-8">
        <form action="/submit-sight" enctype="multipart/form-data"
method="POST">
            <div class="row g-3">
                <div class="col-12">
                    <label for="email" class="form-
label">Email</label>
                    <input type="email" name="email" id="email"
class="form-control">
                </div>

                <div class="col-12">
                    <label for="name" class="form-label">Sight
Name</label>
                    <input type="text" name="name" id="name"
class="form-control" required>
                </div>

                <div class="col-12">
                    <label for="description" class="form-
label">Description</label>
                    <textarea name="description" id="description"
class="form-control" cols="30" rows="4"></textarea>
                </div>

                <div class="col-12">
                    <label for="workingtime" class="form-
label">Working time</label>
                    <input type="text" name="workingtime"
id="workingtime" class="form-control" required>
                </div>
```

А.6 Частина коду, яка відповідає за додавання об’єктів до ІПС

```
                <div class="col-12">
                    <label for="cost" class="form-label">Cost of
visit</label>
                    <input type="text" name="cost" id="cost"
class="form-control" required>
                </div>

                <div class="col-12">
                    <label for="phone" class="form-label">Contact
phone number</label>
                    <input type="text" name="phone" id="phone"
class="form-control" required>
                </div>

                <div class="col-12">
                    <label for="website" class="form-
label">Website</label>
                    <input type="text" name="website" id="website"
class="form-control" required>
                </div>

                <div class="col-12">
                    <label for="category">Select Category</label>
                    <select class="form-select form-control"
name="category" aria-label="Category">
                        <option selected>Select Category</option>
                        <option value="Austrian">Austrian</option>
                        <option value="Czech">Czech</option>
                        <option value="French">French</option>
                        <option value="German">German</option>
                        <option value="Greek">Greek</option>
                        <option value="Dutch">Dutch</option>
                        <option value="Spanish">Spanish</option>
                    </select>
                </div>

                <div class="col-12">
                    <label for="latitude" class="form-
label">Latitude</label>
                    <input type="text" name="latitude" id="latitude"
class="form-control" required>
                </div>

                <div class="col-12">
                    <label for="longitude" class="form-
label">Longitude </label>
                    <input type="text" name="longitude"
id="longitude" class="form-control" required>
                </div>

                <div class="col-12">
                    <label for="image">Sight Image</label>
                    <input type="file" class="form-control"
name="image" accept="image/*">
                </div>
```

А.7 Частина коду, яка відповідає за додавання об’єктів до ІПС

```
                <div class="col-12">
                    <button type="submit" class="btn btn-
primary">Submit Sight</button>
                </div>
            </div>
        </form>
    </div>
</div>
```

А.8 Частина коду, яка відображає останні додані об’єкти до ІПС

```
<h1 class="pb-4">Explore Latest</h1>


<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">
    <% if(sight != '') { %>
        <% sight.forEach(function(sight, index){ %>
        <a href="/sight/<%= sight._id %>" class="col text-center
category__link">
        <div class="category__img category__img--large shadow">
            <img src="/uploads/<%= sight.image %>" alt="<%=
sight.name %>" loading="lazy">
        </div>
        <div class="pt-1"><%= sight.name %></div>
        </a>
        <% }) %>
    <% } else { %>
        <p>No items found.</p>
    <% } %>
</div>
```

А.9 Частина коду, яка виводить будь-який випадковий об’єкт

```
<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">
    <% if(sight != '') { %>

        <a href="/sight/<%= sight._id %>" class="col text-center
category__link">
        <div class="category__img category__img--large shadow">
            <img src="/uploads/<%= sight.image %>" alt="<%=
sight.name %>" loading="lazy">
```

А.10 Частина коду, яка виводить будь-який випадковий об’єкт

```
        </div>
        <div class="pt-1"><%= sight.name %></div>
        </a>

    <% } else { %>
        <p>No items found.</p>
    <% } %>
</div>
```

А.11 Частина коду, яка виводить результати пошуку

```
<h1 class="pb-4">Search Results</h1>


<div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">
    <% if(sight != '') { %>
        <% sight.forEach(function(sight, index){ %>
        <a href="/sight/<%= sight._id %>" class="col text-center
category__link">
        <div class="category__img category__img--large shadow">
            <img src="/uploads/<%= sight.image %>" alt="<%=
sight.name %>" loading="lazy">
        </div>
        <div class="pt-1"><%= sight.name %></div>
        </a>
        <% }) %>
    <% } else { %>
        <p>No items found.</p>
    <% } %>
</div>
```