Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка програмного застосунку для бронювання житла

Виконав: студент __IV__ курсу
групи _4ПІ-20Б_
спеціальності
__121 – Інженерія програмного забезпечення
Мельник Б.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
(прізвище та ініціали)
Рецензент:. к.т.н., доц. каф ЗІ Дудатьєв А.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
Романюк О. Н._____
"_12_" _березня_ 2024 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мельнику Богдану Вікторовичу

1. Тема роботи – розробка програмного застосунку для бронювання житла.
Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "_11_" _березня_ 2024року №_80_.
2. Строк подання студентом роботи_3 червня 2024 року_.
3. Вихідні дані до роботи: алгоритми для додавання та видалення орендних пропозицій з кількістю кроків: 4; алгоритм бронювання житла з часом обробки запиту на бронювання не більше 1,5 секунд; оцінка зручності інтерфейсу користувачами на рівні не менше 8 з 10; продуктивність системи: обробка 60 запитів на бронювання за хвилину; час реакції системи на запити користувачів: не більше 0,5 секунди.
4. Зміст пояснювальної записки: вступ; огляд предметної області, постановка задачі, порівняльний аналіз аналогів та аналіз вимог; обґрунтування вибору середовища розробки; проєктування системи; розробка системи; тестування системи; висновки; список використаних джерел; додатки.
5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; новизна отриманих результатів; практична цінність отриманих результатів; завдання дослідження; порівняння аналогів; вимоги до програмного застосунку; блок-схема алгоритму роботи програми; обрані засоби розробки програмного застосунку; алгоритм додавання кімнати; тестування програмного застосунку; елементи застосунку; висновок.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О.М., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.2024

7. Дата видачі завдання 13 березня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка задачі	14.03.24 – 23.03.24	вик
2	Аналіз аналогів та аналіз вимог	15.03.24 – 23.03.24	вик
3	Проєктування системи	24.03.24 – 18.04.24	вик
4	Вибір середовища та мови розробки	25.04.24 – 26.04.24	вик
5	Розробка системних модулів	27.04.24 – 15.05.24	вик
6	Тестування програми	17.05.24 – 26.05.24	вик
7	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24	вик

Студент _____ **Мельник Б.В.**
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ **Хошаба О.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.92

Мельник Б.В. Розробка програмного застосунку для бронювання житла. : бакалаврська дипломна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 110 с.

На укр. мові. Бібліогр. : 20 назв ; рис. 28; табл. 15.

Дипломна робота присвячена розробці програмного застосунку для бронювання житла. У роботі виконано аналіз існуючих систем бронювання житла, визначено їх переваги та недоліки.

На основі аналізу аналогів було розроблено власний програмний застосунок, який надає можливість користувачам зручно бронювати житло через Інтернет. Застосунок включає в себе інтерфейси для користувачів та адміністраторів.

Основною метою є підвищення ефективності процесу бронювання житла. Для досягнення цієї мети розроблено програмний застосунок, який дозволяє користувачам швидко і зручно здійснювати бронювання житла через Інтернет. Застосунок включає в себе алгоритми для зручного вибору та бронювання доступних пропозицій. Реалізація програмного застосунку виконана з використанням мов програмування та технологій, таких як HTML, SCSS, JS, також в роботі використовувалась база даних Firebase. Отримані результати можуть бути використані для полегшення процесу бронювання житла через Інтернет, що сприятиме зручності та швидкості взаємодії між клієнтами та постачальниками житла.

Ключові слова: програмний застосунок, бронювання житла, веб-система, інтернет, користувач, адміністратор.

ABSTRACT

UDC 004.92

Melnik B.V. Development of a Software Application for Accommodation Booking: Bachelor's Thesis in the field of 121 - Software Engineering, Educational Program - Software Engineering. Vinnytsia: VNTU, 2024. 110p.

In Ukrainian. Bibl.: 20 titles; fig. 28; table. 15.

The thesis is devoted to the development of a software application for accommodation booking. The work includes an analysis of existing accommodation booking systems, identifying their advantages and disadvantages.

Based on the analysis of analogs, our own software application was developed, which allows users to conveniently book accommodation online. The application includes interfaces for both users and administrators.

The main goal is to improve the efficiency of the accommodation booking process. To achieve this goal, a software application has been developed that allows users to quickly and conveniently book accommodation online. The application includes algorithms for convenient selection and booking of available offers. The implementation of the software application is done using programming languages and technologies such as HTML, SCSS, JS, and also Firebase database was used in the work. The obtained results can be used to facilitate the process of accommodation booking online, which will contribute to the convenience and speed of interaction between clients and accommodation providers.

Keywords: software application, accommodation booking, web system, internet, user, administrator.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Проблема бронювання житла через Інтернет стає все більш актуальною в сучасному світі, де мобільність та швидкість стають ключовими аспектами у плануванні подорожей та проживання в інших місцях. Зростаюча популярність онлайн-бронювання свідчить про необхідність надійних та зручних рішень для користувачів. Ринковий попит на програмні застосунки для бронювання житла виявляється через ріст кількості людей, які віддають перевагу онлайн-сервісам у забезпеченні своїх потреб. Покращення ефективності цього процесу може значно полегшити життя користувачів і забезпечити конкурентні переваги на ринку. Використання сучасних технологій розробки програмного забезпечення відкриває широкі можливості для створення ефективних та інноваційних рішень у цій галузі.

Обґрунтування вибору теми дослідження базується на потребах сучасного ринку, де швидкість, зручність та надійність стали важливими критеріями для користувачів при виборі сервісів. Розвиток інтернет-технологій дозволяє створювати інноваційні рішення для забезпечення максимальної зручності та ефективності у бронюванні житла. Дослідження цієї теми не лише відповідає сучасним потребам користувачів, але й має потенціал забезпечити покращення процесу бронювання та зробити його більш доступним для різних категорій людей.

Додатково, враховуючи зростання туристичного ринку та постійний попит на зручні та ефективні сервіси, програмний застосунок для бронювання житла має потенціал залучити нових користувачів і підвищити конкурентоспроможність платформи. Послуги онлайн-бронювання нерухомості також мають велике значення для власників житла, які шукають ефективні способи просування своєї власності та залучення більшого числа клієнтів. Таким чином, розробка програмного застосунку для бронювання житла відповідає як потребам користувачів, так і інтересам бізнесу, забезпечуючи вигоди для обох сторін.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі

програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є надання користувачам зручного та ефективного інструменту для бронювання житла через Інтернет, сприяючи підвищенню якості обслуговування та зручності для користувачів.

Завдання дослідження.

1) Провести аналіз існуючих систем бронювання житла та визначити їх переваги і недоліки.
2) Розробка вимог до програмного застосунку, враховуючи потреби користувачів та власників житла.
3) Розробка архітектури програмного застосунку та його інтерфейсу.
4) Розробити програмний застосунок для бронювання житла, який включатиме у себе зручний та інтуїтивно зрозумілий інтерфейс для користувачів та адміністраторів.
5) Реалізувати алгоритми реєстрації та авторизації користувачів.
6) Реалізувати алгоритми зручного та безпечного бронювання та додавання житла.
7) Реалізувати панель адміністратора для зручного управління наявними пропозиціями, та додаванням нових.
8) Провести тестування програмного застосунку.

Об’єкт дослідження – процес бронювання житла через Інтернет та програмний застосунок, розроблений для забезпечення цього процесу.

Предмет дослідження – програмний застосунок для бронювання житла, його функціональність, ефективність та зручність для користувачів.

Методи дослідження. У процесі досліджень використовувались такі методи: методи оптимізації для покращення ефективності та продуктивності системи шляхом вдосконалення алгоритмів та процесів; методи імітації та моделювання використовуються для вивчення різних сценаріїв функціонування системи та виявлення їх впливу на функціонування системи; методи емпіричних досліджень на основі збору та аналізу реальних даних з використанням системи, що розробляється, для перевірки її функціональності та ефективності у реальних умовах; методи розподіленого обчислення для підвищення масштабованості та

роботи системи в умовах великої кількості одночасних запитів.

Новизна отриманих результатів.

1) Запропоновано новий підхід до розробки програмного забезпечення для бронювання житла, особливість якого полягає у зручному та інтуїтивно зрозумілому інтерфейсі для користувачів та адміністраторів, що дозволяє підвищити ефективність взаємодії з системою за рахунок оптимізації процесів навігації та управління бронюванням.
2) Подальшого розвитку отримав механізм управління житлом, у якому, на відміну від існуючих рішень, виконано інтегровані функції додавання та видалення орендних пропозицій, що дозволило зменшити кількість необхідних дій для оновлення інформації про наявні місця проживання і, як наслідок, підвищити зручність адміністрування та знизити ризик помилок в управлінні пропозиціями.

Практична цінність отриманих результатів. Практична цінність розробленого застосунку полягає у спрощенні процесу бронювання житла для користувачів та покращенні обслуговування клієнтів для бізнесу. Застосунок забезпечує зручний інтерфейс, швидкий доступ до пропозицій, та ефективну взаємодію між постачальниками житла і клієнтами. Це підвищує конкурентоспроможність на ринку нерухомості та подорожей, задовольняючи потреби користувачів.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій праці, опублікованій у співавторстві, автору належать такі результати: розробка програмного застосунку для бронювання житла.

1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ, ПОСТАНОВКА ЗАДАЧІ, ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ ТА АНАЛІЗ ВИМОГ

1.1 Огляд предметної області та постановка задачі

Індустрія оренди місць проживання є однією з найбільш динамічних галузей, яка постійно зазнає змін під впливом технологій, економічних та соціокультурних факторів. Розглянемо цей сегмент в контексті систем бронювання житла та постановки задачі для розробки програмного забезпечення.

Огляд предметної області.

Зростання популярності онлайн-бронювань та цифрових платформ для пошуку та замовлення житла змінило парадигму бронювання, надаючи користувачам широкий вибір та зручний доступ до різноманітних пропозицій. Компанії, що надають послуги оренди, стурбовані процесом постійного вдосконалення своїх платформ, адаптації до потреб користувачів та забезпечення ефективного управління бронюванням. Пошук інноваційних технологій, таких як штучний інтелект та блокчейн, стає ключовим завданням для підтримки конкурентоспроможності.

Платформи, такі як Airbnb і Booking.com, стали основними гравцями на ринку завдяки своїй здатності пропонувати широкий спектр послуг, від оренди квартир до готельних номерів. Вони використовують алгоритми для персоналізації пропозицій, аналітику для прогнозування попиту та штучний інтелект для покращення взаємодії з користувачами. Конкуренція на цьому ринку стимулює інновації та підвищення якості послуг, що надаються.

Постановка задачі.

Під час розробки програмного забезпечення для бронювання житла ми маємо на меті створення інтуїтивно зрозумілого та функціонального продукту, який задовольнятиме потреби як користувачів, так і орендодавців.

Основні завдання включають:

1) Провести аналіз існуючих систем бронювання житла та визначити їх переваги і недоліки:

- Вивчення популярних платформ бронювання житла, таких як Airbnb, Booking.com, HomeAway, для виявлення їх ключових функцій,

зручності використання, рівня безпеки та підтримки користувачів.

- Визначення сильних та слабких сторін кожної з систем для врахування їх при розробці власного продукту.

2) Розробка вимог до програмного застосунку, враховуючи потреби користувачів та власників житла:

- Збір та аналіз вимог від потенційних користувачів (орендарів) та власників житла (орендодавців) щодо функціональності, безпеки та зручності використання.
- Формулювання чітких функціональних та нефункціональних вимог до програмного забезпечення.

3) Розробка архітектури програмного застосунку та його інтерфейсу:

- Створення архітектури програмного забезпечення, що включає серверну частину, базу даних, API та клієнтську частину.
- Дизайн інтуїтивно зрозумілого інтерфейсу користувача для різних типів користувачів (орендарі, орендодавці, адміністратори).

4) Розробити програмний застосунок для бронювання житла, який включатиме у себе зручний та інтуїтивно зрозумілий інтерфейс для користувачів та адміністраторів:

- Реалізація основного функціоналу для користувачів, включаючи пошук житла, перегляд детальної інформації, здійснення бронювання та управління бронюваннями.
- Розробка адміністративної частини для управління контентом, користувачами та бронюваннями.

5) Реалізувати алгоритми зручного та безпечного бронювання та додавання житла:

- Розробка та впровадження алгоритмів, що забезпечують швидке та безпечне бронювання житла, включаючи підтвердження бронювання в реальному часі, обробку платежів та скасування бронювань.
- Забезпечення можливості для власників житла легко додавати нові об'єкти, редагувати інформацію та управляти бронюваннями.

6) Провести тестування програмного застосунку:

- Тестування користувацької частини для перевірки коректності відображення інформації, функціональності реєстрації та авторизації, а також роботи алгоритмів бронювання.
- Тестування адміністративної частини для перевірки коректності роботи функцій управління контентом та користувачами, додавання та видалення житла, а також безпеки системи.

Ці завдання визначають напрямок розробки та основні критерії успіху для програмного застосунку для бронювання житла.

Отже, розглядаючи всі аспекти предметної області та враховуючи поставлені завдання, можна зробити висновок, що успішна розробка програмного забезпечення для бронювання житла потребує комплексного підходу та вдосконалення на різних рівнях. Враховуючи потреби користувачів та вимоги ринку, ми маємо можливість створити продукт, який буде конкурентоспроможним і забезпечить якісне обслуговування як для користувачів, так і для агентств оренди. Ретельний аналіз предметної області та правильно сформульована постановка задачі становлять основу успішної розробки та впровадження програмного забезпечення для бронювання житла.

1.2 Порівняльний аналіз існуючих систем бронювання житла

У сфері розробки програмних засобів для бронювання житла спостерігаються значні трансформації, спровоковані не лише збільшенням попиту на подорожі, але й стрімким розвитком технологій. Програмні рішення у цій сфері набули великого значення, оскільки дозволяють здійснювати бронювання житла через Інтернет та ефективно керувати бізнесом спрямованим на надання послуг оренди житла.

Сучасний ринок програмних засобів для бронювання житла пропонує різноманітні продукти, які відповідають різним потребам та вимогам клієнтів. Від глобальних платформ для бронювання місць проживання до невеликих рішень для затишних місць у провінційних містечках - програми цього типу надають широкий спектр функціональності, від базових інструментів бронювання до розвинутих систем аналітики.

Ключові критерії, які визначають конкурентоспроможність програмних

засобів для бронювання житла, включають:

1) Функціональність: здатність програмного засобу задовольняти різноманітні потреби готельного бізнесу.
2) Зручність інтерфейсу: легкість та зрозумілість використання для персоналу готелю та клієнтів.
3) Інтеграція та сумісність: можливість легкої інтеграції з іншими системами управління та зовнішніми сервісами.
4) Безпека та захист даних: забезпечення конфіденційності та цілісності інформації готелю та гостей.

На сьогоднішній день лідерами на ринку програмного забезпечення для бронювання житла є компанії, такі як Airbnb та Booking.com, які надають широкий вибір житла та мають величезну клієнтську базу. Однак, також існують менші компанії, які спеціалізуються на певних ринкових сегментах або пропонують більш індивідуалізовані послуги.

Порівняння таких платформ, як Airbnb і Booking.com, дозволить краще зрозуміти їхні особливості, переваги та недоліки, а також визначити оптимальний вибір для різних типів помешкань для різних типів клієнтів. Цей аналіз сприятиме підвищенню конкурентоспроможності і покращенню задоволення клієнтів.

Airbnb - це онлайн-платформа для здійснення бронювання житла. Заснована в 2008 році в Сан-Франциско, Airbnb стала своєрідним посередником між людьми, які пропонують свої помешкання для оренди, і подорожніми, які шукають комфортне та унікальне житло під час подорожей.

Основними перевагами Airbnb є:

1) Широкий вибір помешкань: Платформа пропонує різноманітні типи житла - від квартир та будинків до вілл, замків та навіть дерев'яних хатинок у лісі. Користувачі можуть знайти житло для будь-яких потреб та бюджетів.
2) Локальний досвід: Airbnb дозволяє подорожнім відчути себе як місцевими, надаючи можливість зупинятися в житлах від фактичних мешканців міста або району. Це дозволяє подорожнім зануритися в місцеву культуру та отримати унікальний досвід.
3) Безпека та відгуки: Airbnb забезпечує безпеку користувачів, проводячи

перевірку якості житла та власників помешкань. Крім того, користувачі можуть залишати відгуки та оцінки після свого перебування, що допомагає іншим подорожнім приймати рішення щодо бронювання.

4) Гнучкість: Airbnb дозволяє користувачам знаходити житло на будь-який смак та на будь-який термін - від однієї ночі до тривалого перебування.
5) Локалізація та місцева підтримка: Платформа працює у багатьох країнах світу, що дозволяє знаходити житло в будь-якій частині світу. Крім того, в деяких місцевостях Airbnb пропонує місцеву підтримку та поради для подорожніх.

Airbnb стала популярним вибором для тих, хто шукає унікальний та комфортний досвід під час подорожей. Завдяки широкому вибору житла та гнучкості у виборі, Airbnb продовжує приваблювати як індивідуальних мандрівників, так і сімейні подорожі, ділові поїздки та інших подорожніх. Інтерфейс Airbnb зображено на рисунку 1.1.

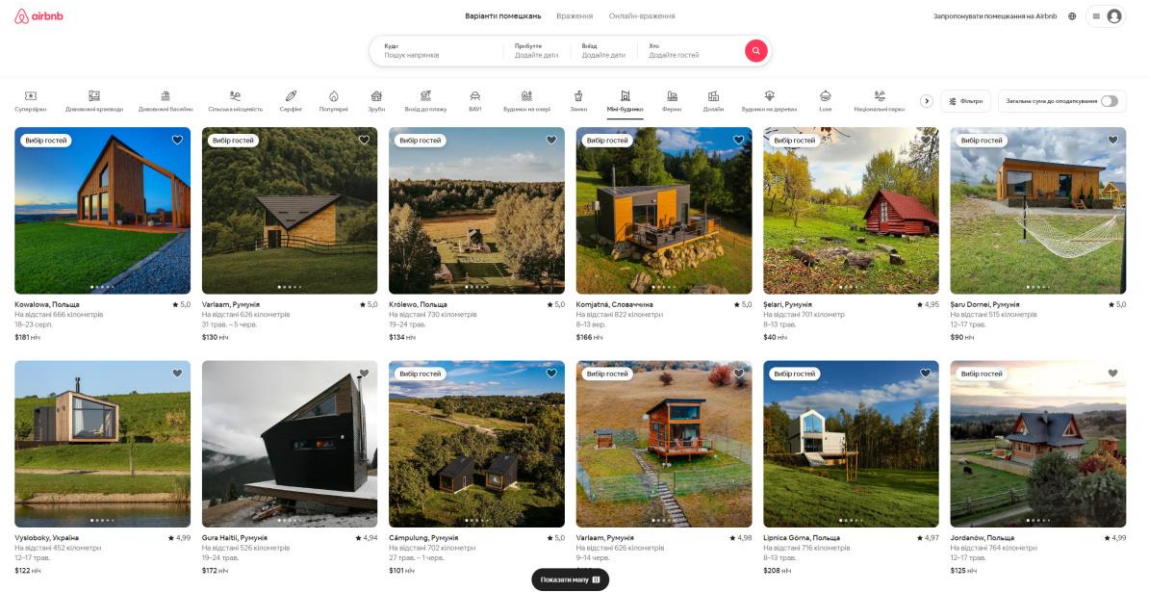


Рисунок 1.1 – Інтерфейс головної сторінки «Airbnb»

З недоліків Airbnb можна виділити такі:

1) Ризик непередбачуваних ситуацій: Іноді подорожні можуть зіткнутися з неприємностями, такими як скасування бронювання або недотримання умов оренди від власників житла.
2) Неоднакова якість помешкань: Якість та стандарти житла можуть значно відрізнятися, що може призвести до негативних вражень під час подорожі.

3) Платформні витрати: Airbnb вимагає плати за обслуговування, яка може збільшити вартість перебування.
4) Відсутність гарантій: На відміну від традиційних готелів, Airbnb не може гарантувати однаковий рівень сервісу та комфорту у всіх випадках.
5) Потенційні проблеми з правовими аспектами: Деякі місцеві закони та правила можуть обмежувати здійснення користувачами оренди через Airbnb, що може призвести до юридичних проблем.

Airbnb - популярна платформа для бронювання житла, яка пропонує широкий вибір унікальних помешкань у різних куточках світу. Її основні переваги включають широкий вибір житла, унікальний досвід для подорожніх, гнучкість у виборі та локальні поради. Однак, серед недоліків можуть бути ризик непередбачуваних ситуацій, різна якість помешкань та платформні витрати. В цілому, Airbnb є популярним вибором для подорожніх, проте важливо бути обережним та уважним під час вибору житла.

Booking.com - це одна з найбільших онлайн-платформ для бронювання апартаментів, гостьових будинків та інших типів помешкань по всьому світу. Заснована в 1996 році, вона належить до групи Booking Holdings Inc. Booking.com пропонує широкий вибір житла у різних цінових категоріях та країнах, а також надає можливість бронювання трансферів, автомобілів оренди та екскурсій. Сервіс відомий своєю зручністю використання, широким вибором фільтрів для пошуку, а також відгуками та оцінками від користувачів. Booking.com також пропонує програму лояльності для постійних клієнтів та додаткові знижки. Інтерфейс Booking.com зображено на рисунку 1.2.

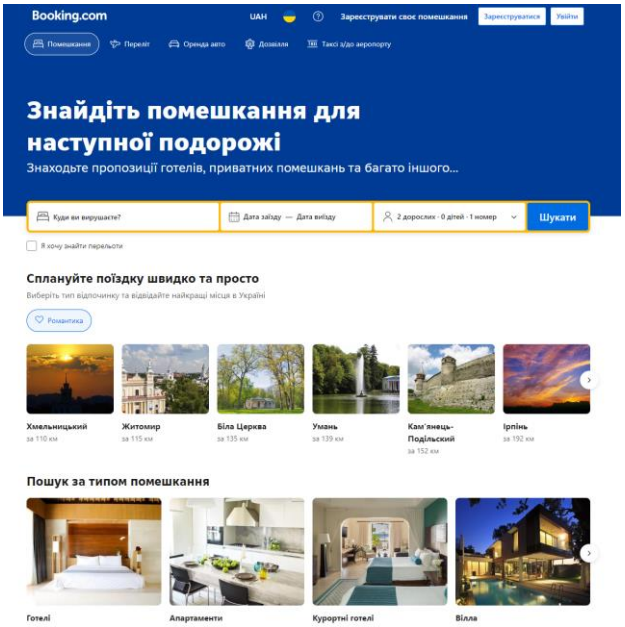


Рисунок 1.2 – Інтерфейс головної сторінки «Booking.com»

Основні переваги Booking.com включають:

1) Широкий вибір житла: Платформа пропонує великий вибір готелів, апартаментів, гостьових будинків та інших типів помешкань у різних країнах і містах.
2) Зручний пошук та фільтрація: Booking.com має зручний і простий інтерфейс, який дозволяє швидко знаходити та фільтрувати пропозиції за різними критеріями, такими як ціна, рейтинг, розташування тощо.
3) Відгуки та оцінки: Користувачі можуть переглядати відгуки та оцінки інших подорожуючих, що допомагає приймати обґрунтовані рішення про вибір житла.
4) Програма лояльності: Booking.com надає можливість отримання знижок та додаткових переваг для постійних клієнтів через свою програму лояльності.

Недоліки Booking.com включають:

1) Платформні витрати: Деякі користувачі можуть виявити, що ціни на Booking.com включають додаткові платформні витрати, які роблять деякі пропозиції дорожчими, ніж при бронюванні напряму через готелі або інші сервіси.
2) Залежність від відгуків: Хоча відгуки є корисними для прийняття рішень, іноді вони можуть бути неповними або недостовірними, що

може призвести до ризику неприємних сюрпризів при заселенні.

3) Обмежена доступність: У деяких випадках найкращі пропозиції можуть бути обмежені або відсутні для певних дат або місць, що може створювати нестабільність у пошуку.

Booking.com - популярна онлайн-платформа для бронювання житла, яка пропонує широкий вибір готелів та інших типів помешкань у різних країнах та містах. Платформа відома своїм зручним пошуком та фільтрацією пропозицій, великою кількістю відгуків та оцінок від інших користувачів, а також програмою лояльності для постійних клієнтів. Однак варто враховувати деякі недоліки, такі як можливі платформні витрати та обмежена доступність деяких пропозицій. Загалом, Booking.com залишається популярним вибором для багатьох подорожуючих завдяки своїй широкій базі житла та зручному інтерфейсу користувача.

Власний програмний додаток для бронювання пропонує зручний у використанні інтерфейс та привабливий дизайн додатку, що забезпечує приємний користувацький досвід і роблять процес бронювання житла максимально зручним та ефективним. Користувачі швидко орієнтуються в програмі, легко знаходять потрібні функції та швидко здійснюють бронювання.

Власний програмний додаток зображено на рисунку 1.3.

Рисунок 1.3 – Дизайн власного програмного додатку

Однак відмінність продукту полягає не лише в його зовнішньому вигляді. Він надає користувачам декілька способів реєстрації та авторизації, що робить процес входу в систему більш гнучким та адаптованим до потреб різних користувачів. Відчуття особистого підходу та можливість вибору зручного методу входу створюють позитивне враження від користування програмою та сприяють формуванню лояльності користувачів.

Порівняння Airbnb і Booking.com, та власного продукту наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів з власною розробкою

Критерій	Airbnb	Booking.com	Власна розробка
Дизайн	+/-	+/-	+
Реєстрація	+/-	+/-	+
Інтуїтивність	-	-	+
Клієнтська підтримка	+/-	+	+
Швидкість роботи	+/-	+/-	+
Результат	2	3	5

Власний програмний додаток було розроблено з урахуванням аналізу переваг та недоліків відомих аналогів, таких як Airbnb і Booking.com. Під час розробки враховувалися найкращі практики цих платформ, а також приділено увагу недолікам цих аналогів, які можна було покращити.

Основні переваги власного програмного додатку включають зручний та привабливий інтерфейс, різноманітні способи реєстрації та авторизації, підримка користувачів та швидкодія роботи. Ці аспекти роблять додаток конкурентоспроможним на ринку бронювання житла та надають користувачам значний комфорт та задоволення від використання.

Таким чином, розробка власного програмного додатку була спрямована на створення інноваційного та конкурентоспроможного рішення, яке поєднує в собі найкращі аспекти відомих платформ та вирішує їхні недоліки.

1.3 Визначення типів користувачів та ключових вимог

Для успішного розроблення програмного застосунку важливо враховувати різноманітні потреби та очікування різних типів користувачів, та їх потреб.

Для початку визначимо основні критерії для вибору типів користувачів програмного забезпечення для бронювання житла:

1) Різноманітні потреби та очікування. Користувачі мають різні очікування від сервісу, в залежності від їхніх цілей подорожі, фінансових можливостей та інших факторів. Наприклад, деякі можуть шукати бюджетні варіанти, тоді як інші спрямовуються на розкішні місця проживання з вищим рівнем обслуговування.
2) Різноманітність технічних можливостей. Різні користувачі мають відмінні технічні потреби та вимоги. Деякі можуть користуватися лише мобільними пристроями, тоді як інші використовують комп'ютери.
3) Аналіз поведінки користувачів. Вивчення попередніх дій користувачів може допомогти визначити їхні попередні вибори та уподобання стосовно житла.
4) Географічні або демографічні аспекти. Сегментація за географічними або демографічними параметрами може бути корисною для вирішення питань локалізації та персоналізації.
5) Частота користування. Розрізнення користувачів за їхньою активністю у системі може допомогти визначити пріоритети у покращенні та обслуговуванні користувачів.

Наступним кроком визначимо декілька типів користувачів та їхні очікування:

1) Туристи: Це ключова категорія, яка включає індивідуальних мандрівників та сімейні групи, що шукають затишне місце для перебування під час відпочинку. Для них важливо знаходитися в центрі, близько до туристичних об'єктів та мати зручний доступ до громадського транспорту.
2) Сімейні відпочиваючі: Ці користувачі шукають житло, яке підходить для всієї сім'ї. Вони вимагають від програми вибір розміру кімнати та додаткові послуги, такі як наявність кондиціонеру, доступ до Wi-Fi, тощо.
3) Бізнес-подорожні: Ці користувачі шукають комфортабельні готелі або апартаменти з усіма необхідними зручностями для успішних ділових

поїздок. Вони можуть цінувати функціонал для зручного бронювання, наявність робочого простору та швидкий доступ до послуг готелю.

4) Бюджетні мандрівники: Ці користувачі шукають доступні варіанти проживання з недорогими цінами та основними зручностями. Для них важливо мати можливість швидко дістатися з житла до транспорту, а також мати доступ до магазинів та закладів харчування з прийнятними цінами.

Врахування потреб та очікувань цих, та інших типів користувачів допоможе підвищити ефективність та використовуваність програмного застосунку для бронювання житла. Розуміння та врахування індивідуальних потреб різних категорій користувачів дозволить налаштувати програму так, щоб вона відповідала їхнім очікуванням та надавала необхідний функціонал.

Визначивши типи користувачів, можна створити маркетингову стратегію, яка враховує їхні потреби та пропонує відповідні послуги та функціонал для програмного забезпечення. Це означає, що для кожного типу користувачів має бути розроблено окремі стратегії реклами, персоналізовані пропозиції та інші маркетингові заходи, які будуть спрямовані на задоволення їхніх потреб.

Розробка стратегії маркетингу для кожного з сегментів користувачів програмного забезпечення для бронювання житла може включати такі етапи:

1) Рекламна стратегія: Вибір маркетингових каналів та інструментів для просування продукту серед кожного сегменту.
2) Персоналізація інтерфейсу: Адаптація програмного забезпечення для кожного сегменту користувачів, забезпечення зручного та інтуїтивного інтерфейсу, який відповідає їхнім потребам.

Враховуючи різноманітність типів клієнтів, розробимо унікальні стратегії маркетингу для кожного з них.

Розроблені стратегії маркетингу для різних типів користувачів:

1) Туристи

- Рекламна стратегія: Реклама в соцмережах, туристичних блогах, виставках.

- Персоналізація інтерфейсу: Пошук за місцем розташування та зручне бронювання.

2) Сімейні відпочиваючі

- Рекламна стратегія: Реклама в дитячих форумах, журналах для батьків.

- Персоналізація інтерфейсу: Можливість вибору кімнати за розміром та додатковими послугами.

3) Бізнес-подорожні

- Рекламна стратегія: Реклама через бізнес-статті, бізнес-форуми та конференції.

- Персоналізація інтерфейсу: Можливість замовлення додаткових послуг.

4) Бюджетні мандрівники

- Рекламна стратегія: Реклама в блогах про бюджетні подорожі, сайтах зі знижками.

- Персоналізація інтерфейсу: Наявність помешкань економ класу.

Визначення типів користувачів, та розробка індивідуальних маркетингових стратегій для кожного типу користувачів програмного забезпечення є важливим етапом у залученні і утриманні цільової аудиторії. Ефективне використання рекламних каналів та персоналізація інтерфейсу дозволять привернути увагу різних категорій користувачів та забезпечити їм зручний та відповідний сервіс.

Аналіз потреб і очікувань різних типів користувачів важливий для розробки успішного програмного забезпечення для бронювання житла. Розглянемо потреби та очікування кожного з вищезазначених типів користувачів:

1) Туристи:
- Потреби: Затишне та комфортне житло в центрі або поруч з туристичними об'єктами. Швидке та зручне бронювання.
- Очікування: Інтуїтивний та зручний інтерфейс програми для швидкого бронювання житла. Фотографії та описи житла, щоб зрозуміти, чи відповідає воно їхнім потребам.

2) Сімейні відпочиваючі:
- Потреби: Просторе та зручне житло для всієї сім'ї. Наявність додаткових послуг.
- Очікування: Можливість вибору кімнат залежно від кількості членів сім'ї.

3) Бізнес-подорожні:

- Потреби: Комфортабельне житло з усіма зручностями для відпочинку та роботи.
- Очікування: Наявність робочого простору, Wi-Fi та інших послуг для роботи. Швидке та надійне бронювання.

4) Бюджетні мандрівники:

- Потреби: Доступне житло з базовими зручностями за прийнятну ціну. Близькість до громадського транспорту та магазинів.
- Очікування: Можливість швидкого та простого бронювання. Інформація про житло.

Розуміння вимог та очікувань різних категорій користувачів є критичним кроком у процесі розробки програмного забезпечення.

Під час реалізації такого програмного продукту важливо враховувати не лише потреби зазначених категорій, але й інших, таких як подорожуючі з домашніми тваринами, мандрівники з обмеженими можливостями та інших. Адаптація програми до різноманітних потреб користувачів підвищить їхню задоволеність від користування системою та сприятиме її ефективності.

Цей аналіз також допомагає визначити пріоритети у вдосконаленні та розвитку програмного забезпечення. Наприклад, якщо існують значні вимоги корпоративних клієнтів, може бути доцільним вдосконалення системи для підтримки масового бронювання та ефективної обробки великого обсягу даних. З іншого боку, якщо туристи є основною аудиторією, акцент може бути зроблений на інтуїтивно зрозумілий інтерфейс та великий вибір місць проживання та послуг для клієнтського задоволення.

Загалом, аналіз вимог та очікувань різних категорій користувачів є ключовим етапом у процесі розробки програмного забезпечення, оскільки він дозволяє точно налаштувати функціональність та інтерфейс програми для максимального задоволення різноманітності потреб користувачів.

Важливо підкреслити, що розуміння потреб різних груп користувачів є вирішальним для успішного розроблення та впровадження програмного забезпечення для бронювання житла. Адаптація продукту до цих вимог допомагає забезпечити його популярність серед широкого кола користувачів та зробить його використання максимально зручним та ефективним.

1.4 Обґрунтування вибору середовища розробки

Вибір середовища розробки є важливим етапом у процесі створення програмного забезпечення, оскільки впливає на ефективність роботи розробників та якість кінцевого продукту. Для розробки системи бронювання житла було обрано Visual Studio Code (рис.1.4).

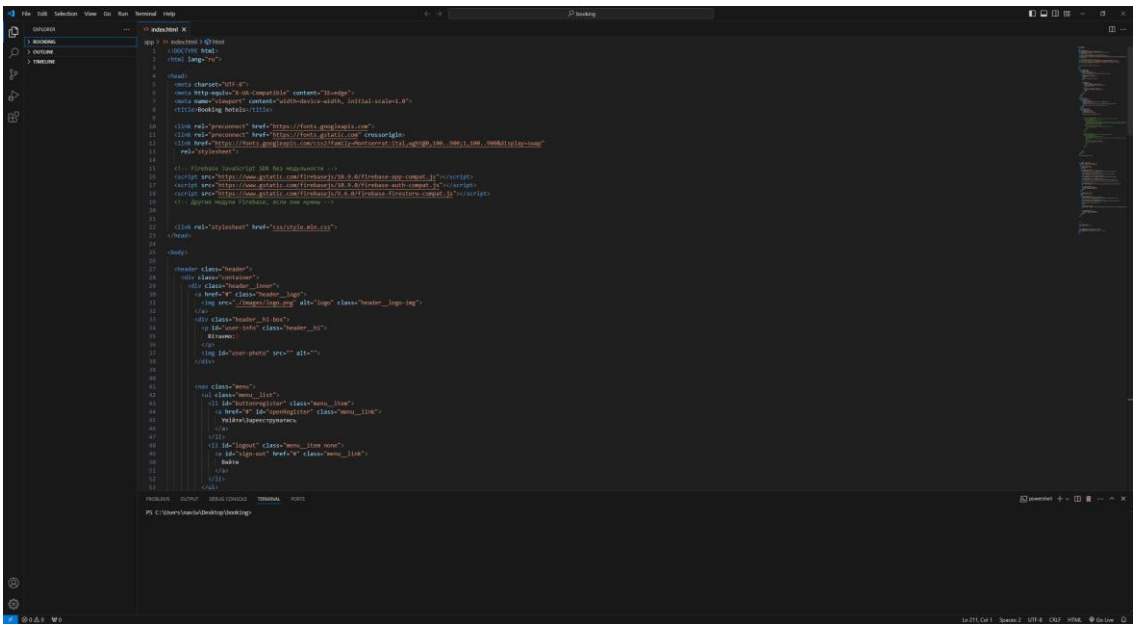


Рисунок 1.4 – Інтерфейс середовища Visual Studio Code

Цей вибір обґрунтовується наступними факторами:

1) Широкий спектр функціональності:

Visual Studio Code (VS Code) пропонує багатий набір функцій, які сприяють продуктивній роботі. Це включає в себе автозавершення коду, інтегровану систему контролю версій (Git), дебаггер, підтримку численних мов програмування та багато іншого. Завдяки цьому розробники можуть швидко та ефективно писати, тестувати та відлагоджувати код.

2) Розширюваність:

VS Code підтримує велику кількість розширень, які можуть бути додані для покращення функціональності середовища розробки. Це дозволяє налаштувати середовище під конкретні потреби проекту. Для розробки системи бронювання житла було використано розширення для роботи з JavaScript, Firestore, а також інструменти для перевірки якості коду та форматування.

3) Підтримка сучасних технологій:

VS Code добре інтегрується з сучасними технологіями, такими як Firebase та Firestore, що є ключовими компонентами у нашій системі бронювання. Підтримка цих технологій дозволяє без проблем працювати з базою даних та виконувати необхідні запити.

4) Висока продуктивність:

Незважаючи на багатий функціонал, VS Code залишається легким і швидким середовищем розробки. Це дозволяє зберігати продуктивність навіть при роботі з великими проектами та великою кількістю файлів.

5) Кросплатформеність:

VS Code доступний на різних операційних системах, включаючи Windows, macOS та Linux. Це забезпечує гнучкість у виборі робочого середовища для розробників, що є особливо важливим у командних проектах, де розробники можуть використовувати різні ОС.

6) Активна спільнота:

VS Code має активну спільноту користувачів та розробників, що забезпечує регулярне оновлення та підтримку продукту. Спільнота також надає велику кількість ресурсів, таких як документація, навчальні матеріали та приклади коду, що спрощує навчання та вирішення проблем.

7) Інтеграція з хмарними сервісами:

Visual Studio Code легко інтегрується з різними хмарними сервісами, що є важливим для проектів, які використовують хмарну інфраструктуру для зберігання та обробки даних. У нашому випадку, інтеграція з Firebase дозволяє ефективно керувати базами даних, забезпечувати масштабованість та високу доступність системи.

8) Розширена підтримка мов програмування:

Крім основних мов, таких як JavaScript, Visual Studio Code підтримує широкий спектр інших мов програмування та фреймворків. Це забезпечує гнучкість у виборі технологій та можливість додаткової інтеграції з іншими системами або сервісами в майбутньому.

9) Безпека та конфіденційність:

Visual Studio Code забезпечує високий рівень безпеки, включаючи підтримку SSL/TLS для захищеного з'єднання з віддаленими серверами та

сервісами. Це важливо для захисту даних користувачів та забезпечення безпечної роботи всієї системи бронювання.

Вибір Visual Studio Code для розробки системи бронювання житла обумовлений його багатим функціоналом, гнучкістю та можливістю налаштування під потреби проекту. Це середовище розробки забезпечує підтримку сучасних технологій, високу продуктивність та безпеку. Завдяки активній спільноті та регулярним оновленням, VS Code залишається актуальним та конкурентоспроможним вибором для будь-якого програмного проекту.

Таким чином, Visual Studio Code є надійним та зручним інструментом, який дозволяє швидко та ефективно створювати високоякісні програмні продукти, що відповідають вимогам користувачів та ринку. Використання цього середовища розробки сприятиме успішному виконанню проекту з розробки системи бронювання житла, забезпечуючи його якість та конкурентоспроможність.

Складемо таблицю порівняння Visual Studio Code з аналогами (табл 1.2).

Таблиця 1.2 – Порівняння середовищ розробки

Критерій	VS Code	IntelliJ IDEA	Atom
Підтримка мов	+	+/-	+/-
Швидкодія	+	+	+/-
Відкритий вихідний код	+	-	+
Інтеграція з хмарними сервісами	+	+/-	+/-
Розмір інсталяції	+	-	+/-
Розширення	+	+	+
Результат	6	3	4

Visual Studio Code отримує найбільший бал завдяки своїй універсальності, широкій підтримці мов програмування, розширенням та інтеграції з хмарними сервісами. Це робить його відмінним вибором для широкого спектру

розробників, забезпечуючи комфортну і ефективну роботу. IntelliJ IDEA також є потужним інструментом, особливо для розробників на JVM, але має обмежену підтримку мов програмування. Atom, хоча і має багато можливостей налаштування, поступається VS Code в деяких ключових аспектах, таких як швидкодія і інтеграція з хмарними сервісами.

1.5 Висновки

У цьому розділі було проведено огляд предметної області, пов'язаної з розробкою програмного забезпечення для бронювання житла. Було розглянуто ключові аспекти індустрії оренди місць проживання, зокрема зростання популярності онлайн-бронювань та цифрових платформ, а також зміни в підходах до бронювання через вплив нових технологій та соціокультурних факторів.

Постановка задачі полягає у визначенні основних цілей та вимог до програмного забезпечення для бронювання житла, враховуючи потреби різних типів користувачів, від туристів до бізнес-подорожніх.

Порівняльний аналіз аналогів дозволив виявити переваги та недоліки існуючих систем бронювання житла, таких як Airbnb і Booking.com, що допомогло визначити ключові функції та можливості для власної розробки.

2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Створення вимог до програмного забезпечення

Розробка архітектури системи - це ключовий етап у процесі створення програмного забезпечення для системи бронювання житла. Цей підрозділ включає в себе низку кроків та рішень, спрямованих на створення структури системи, яка відповідає вимогам до продукту та забезпечує його ефективну роботу. Для початку потрібно визначити основні функціональні і не функціональні вимоги програми.

Перед створенням функціональних і не функціональних вимог – розглянемо можливі варіанти використання системи користувачами, та адміністраторами. На рисунку 2.1 зображена діаграма варіантів використання системи користувачем.

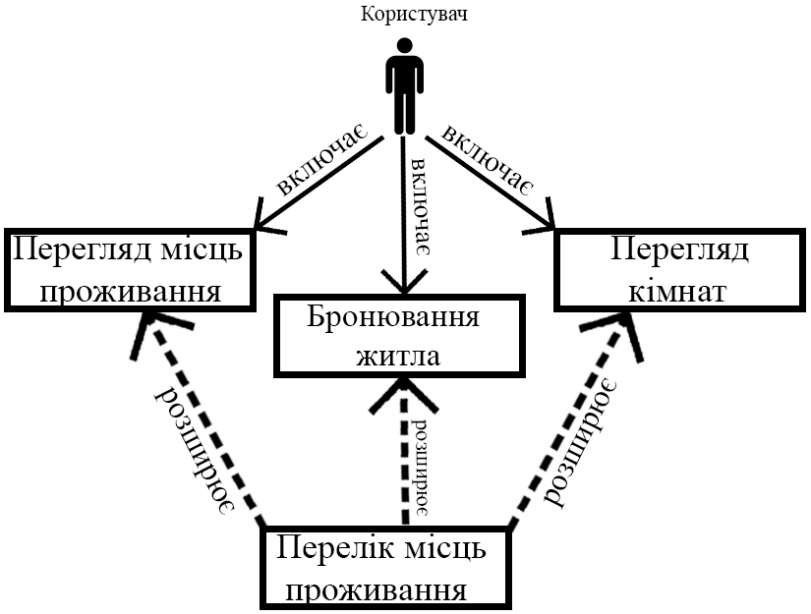


Рисунок 2.1 – Діаграма варіантів використання системи користувачем

Далі розглянемо можливі варіанти використання системи адмі ністратором. Діаграма варіантів використання системи адміністратором зображена на рисунку 2.2.

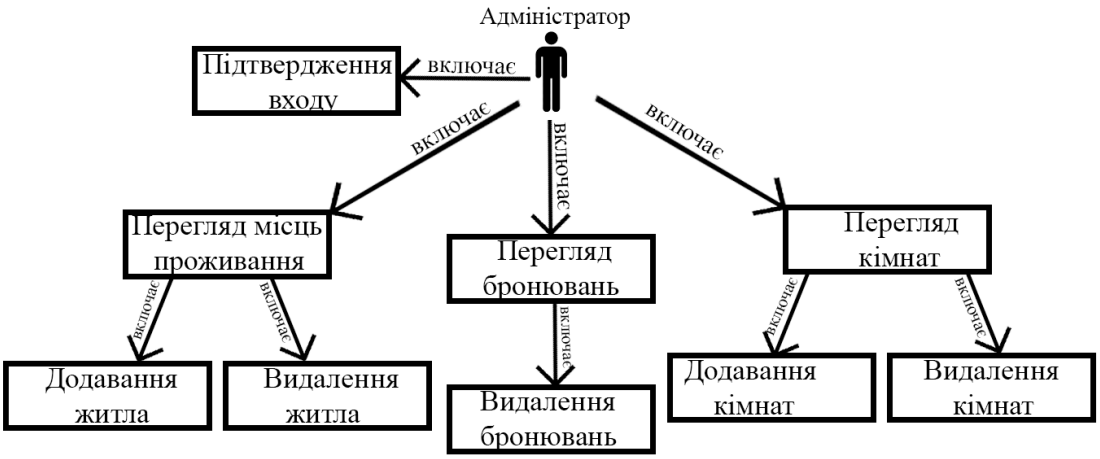


Рисунок 2.2 – Діаграма варіантів використання системи адміністратором

Далі перейдемо до визначення функціональних і не функціональних вимог.

Визначення ключових вимог до програмного забезпечення для бронювання житла є критичним етапом у процесі розробки, оскільки воно визначає основні функціональні та нефункціональні характеристики системи. Розглянемо деякі функціональні і не функціональні вимоги.

Першим кроком розглянемо функціональні вимоги. Вони включають в себе:

1) Бронювання:

- Можливість швидкого та зручного бронювання житла для користувачів (рис. 2.3).

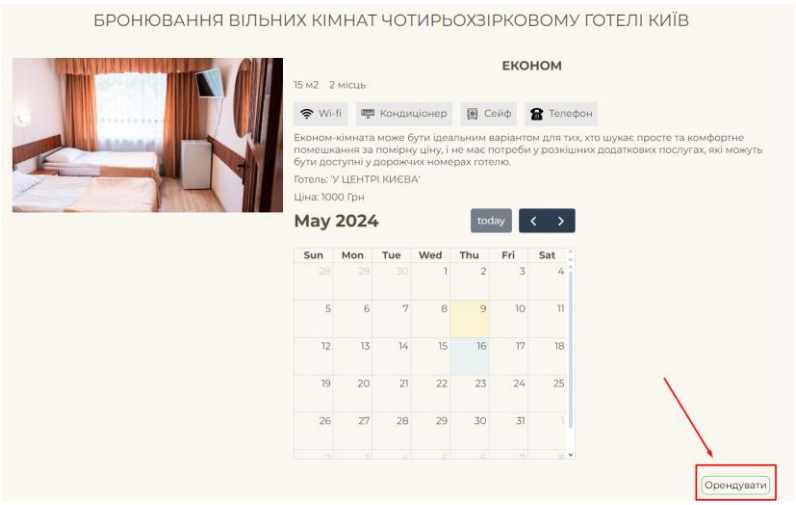


Рисунок 2.3 – Оренда житла

Орендувати житло можуть тільки авторизовані користувачі.

- Можливість відмінити бронювання (рис. 2.4).

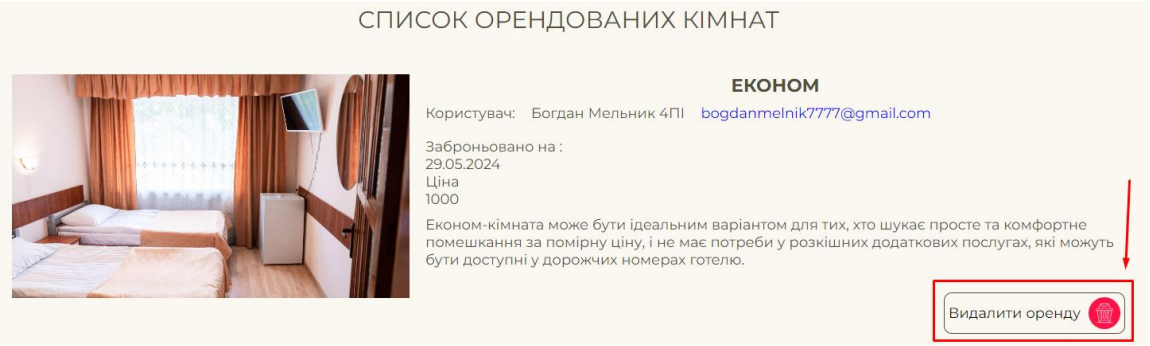


Рисунок 2.4 – Відміна оренди житла

Після того, як користувач орендував житло – з ним зв’язується адміністратор для планування подальших дій. Якщо ж користувач вирішив відмінити оренду – він повинен повідомити про це адміністратора і в такому випадку адміністратор може відмінити оренду.

- Забезпечення можливості перегляду детальної інформації про обране житло, включаючи фотографії та опис (рис. 2.5).

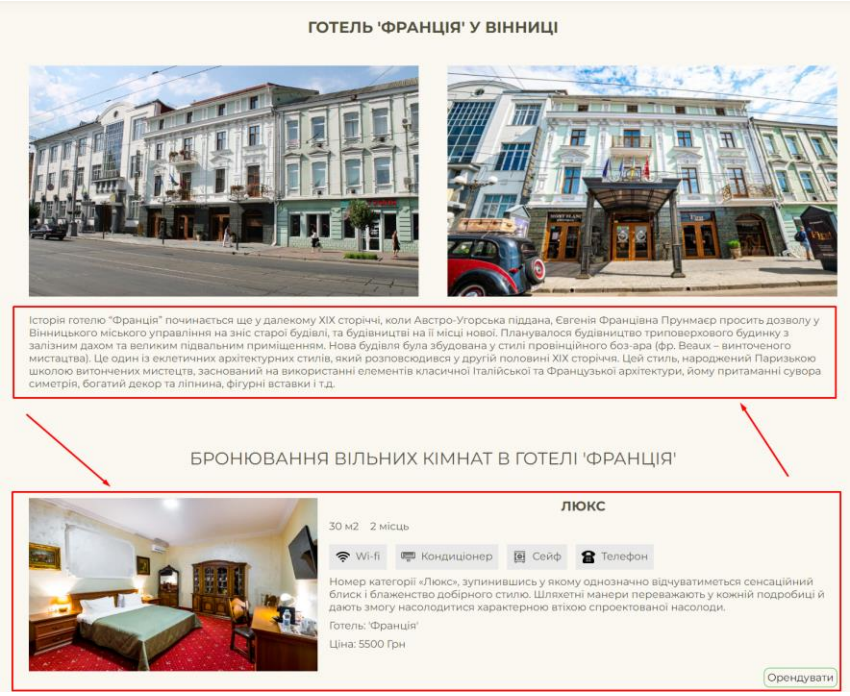


Рисунок 2.5 – Інформація про місце проживання

У користувачів є можливість переглянути детальну інформацію як про місце проживання, так і про окрему кімнату, яка зацікавила користувача, включаючи можливість перегляду фотографій житла.

2) Управління:

- Інтерфейс для адміністраторів готелів або власників нерухомості для керування бронюваннями, наявністю кімнат та їх видаленням (рис 2.6).

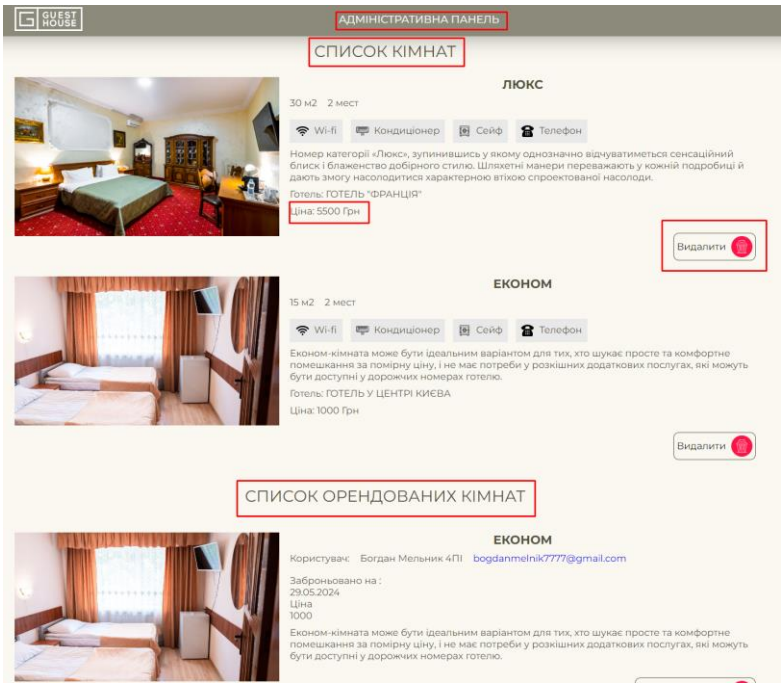


Рисунок 2.6 – Панель адміністратора

Для входу в панель адміністратора необхідно вписати «admin_panel» перед «.html», потім ввести код доступу і натиснути «увійти»

- Забезпечення функціоналу додавання житла (рис. 2.7).

Рисунок 2.7 – Функціонал адміністратора

Далі розглянемо не функціональні вимоги. Вони включають в себе:

1) Зручність використання:

- Інтуїтивний та легкий у використанні інтерфейс як для користувачів, так і для адміністраторів.

2) Надійність:

- Забезпечення безперебійної роботи системи та захист від можливих відмов.

3) Безпека:

- Захист персональних даних користувачів.
- Альтернативний варіант реєстрації / входу користувачів (рис. 2.8).

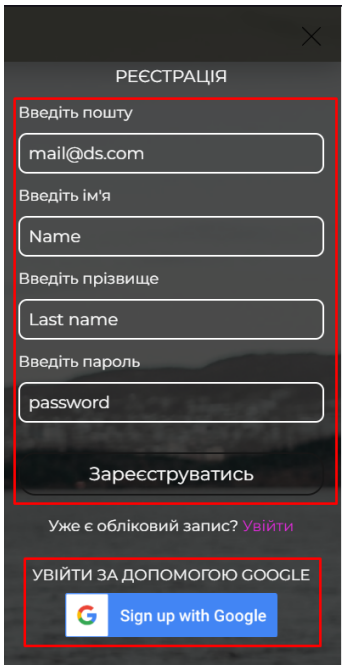


Рисунок 2.8 – Варіанти входу / реєстрації

Після аналізу аналогів таких як Airbnb і Booking.com було прийняте рішення реалізувати як традиційний варіант реєстрації / входу, через пошту і пароль, так і через обліковий запис Google.

4) Швидкість:

- Оптимізація часу реакції програми для забезпечення швидкого бронювання та надання відповідей.

Також розглянемо вимоги до продуктивності, вони включають в себе такі аспекти:

1) Максимальна пропускна здатність для одночасних користувачів.

- Система має бути здатна підтримувати оптимальну кількість одночасних користувачів для безперервної роботи навіть у найінтенсивніші періоди.

2) Швидкість реакції системи.

- Мінімізація часу, який система витрачає на обробку запитів користувачів, для поліпшення їхнього користувацького досвіду.

Ці вимоги є основними для успішної розробки програмного забезпечення для бронювання готелів, яке відповідає потребам та очікуванням користувачів.

Процес розробки вимог до програмного забезпечення є складним і відповідальним, вимагаючи уважного аналізу потреб користувачів, конкурентного середовища та технічних можливостей. Головною метою створення вимог є забезпечення того, щоб розроблений продукт відповідав вимогам ринку, був зручним у використанні для користувачів та мав високу якість та ефективність роботи.

Функціональні вимоги визначають основні можливості системи та функції, які вона повинна виконувати. Нефункціональні вимоги, у свою чергу, визначають якісні аспекти роботи системи, такі як швидкодія, безпека та доступність. Вони є так само важливими, оскільки впливають на загальний досвід користувачів та надійність роботи програми.

Отже, розробка вимог до програмного забезпечення є ключовим етапом у процесі створення продукту, який задовольняє потреби користувачів та вимоги ринку. Чітко визначені функціональні та нефункціональні вимоги дозволяють створити програмне забезпечення, яке є надійним, ефективним та зручним у використанні. Врахування цих вимог забезпечить успішне впровадження проекту та задоволення потреб користувачів.

2.2 Розробка архітектури системи

Розробка архітектури програмної системи є важливим етапом у її створенні. На цьому етапі формується загальна структура системи, визначаються її складові частини та взаємозв'язки між ними. Основною метою є створення концептуального та технічного плану, що визначає основні принципи та підходи до подальшої розробки програмного забезпечення.

Для створення веб-системи було прийняте рішення обрати трьохрівневу архітектуру, що складається з клієнтського додатку, який з'єднується з сервером додатку, а цей, у свою чергу, взаємодіє з сервером бази даних.

Структурну схему системи зображено на рисунку 2.9.

Рисунок 2.9 – Структурна схема системи

Клієнтський додаток (Frontend): Це інтерфейс, з яким користувач взаємодіє. Він реалізований за допомогою HTML, CSS, та JavaScript. Клієнтський додаток відправляє запити на сервер додатків та отримує відповіді.

Сервер додатку (Backend): Це сервер, який обробляє запити від клієнтського додатку. Він включає в себе логіку додатку та доступ до бази даних. Firebase виступає як сервер додатків, який надає API для взаємодії з клієнтським додатком та базою даних.

Сервер баз даних (Database Server): Це сервер, на якому знаходяться бази даних. На цьому сервері розташовано Firestore. Клієнтський додаток та сервер додатків взаємодіють з цим сервером для отримання та зберігання даних.

Веб-система ґрунтується на трьохрівневій архітектурі, що включає фронтенд (клієнтський додаток), бекенд (сервер додатку) та сервер баз даних. Клієнтський додаток відповідає за інтерфейс користувача, сервер додатку - за логіку додатку та взаємодію з базою даних, а сервер баз даних - за зберігання та обробку даних. За допомогою цієї архітектури можна ефективно розробляти та масштабувати веб-застосунки з великим обсягом функцій та данних.

Під час візуалізації кроків, які програма виконує для бронювання житла, було розроблено блок-схему алгоритму роботи програми. Блок-схема надає загальний огляд основних кроків та логіки роботи програми.

Блок-схема алгоритму роботи програми зображена на рисунку 2.10.

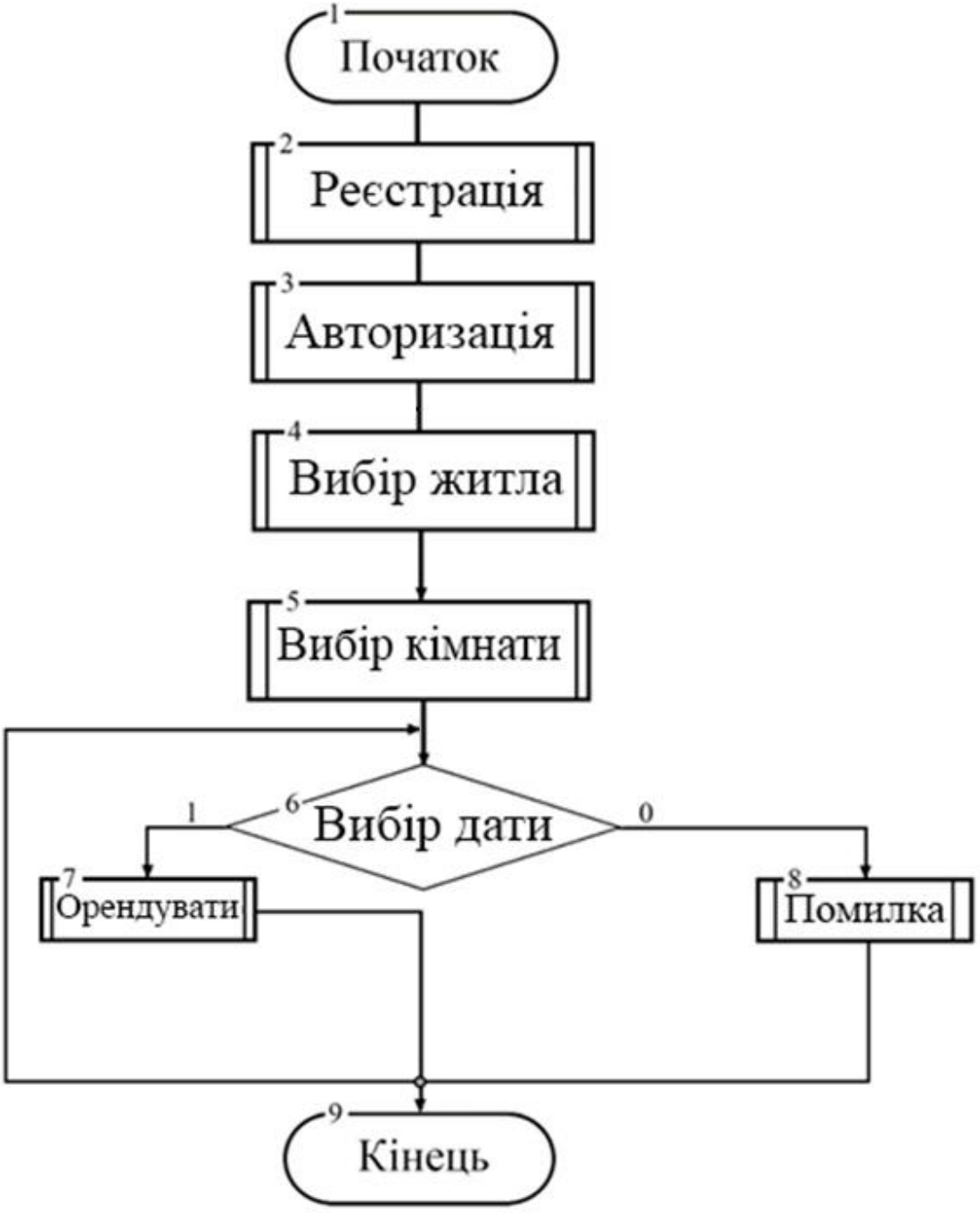


Рисунок 2.10 – Блок-схема алгоритму роботи програми

Кроки виконання програми:

1) Початок.
2) Спершу користувач повинен зареєструватись.
3) Наступним кроком є авторизація користувача.
4) Далі клієнт повинен обрати житло, в якому б він хотів заселитися.
5) Після вибору житла – користувач повинен обрати кімнату, яка його зацікавила.
6) Після вибору кімнати – йому буде доступний вибір дати (рис. 2.11).

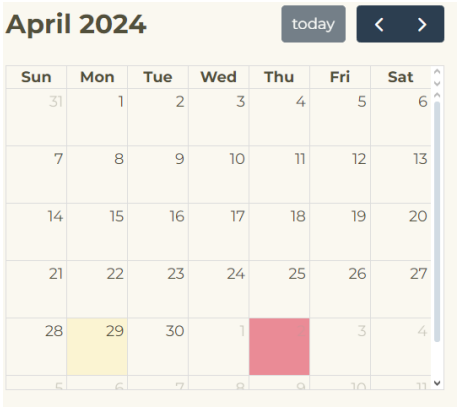


Рисунок 2.11 – Вибір дати оренди кімнати

Після цього користувач зможе знову обрати дату оренди.

7) Якщо клієнт вибере вільну дату – з'явиться наступне повідомлення (рис. 2.12)

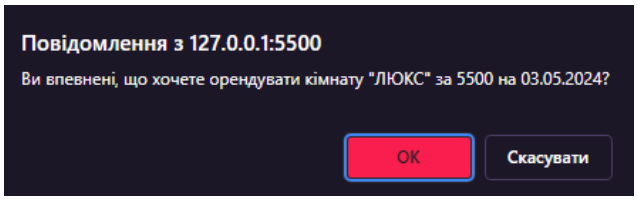


Рисунок 2.12 – Повідомлення з підтвердженням оренди

Після успішної оренди кімнати – клієнт може орендувати як і цю ж кімнату на інші дні – так і інші кімнати.

8) Якщо ж клієнт обере вже зайняту дату – з’явиться помилка, яка повідомить його про те, що ця дата вже зайнята (рисунок 2.12)

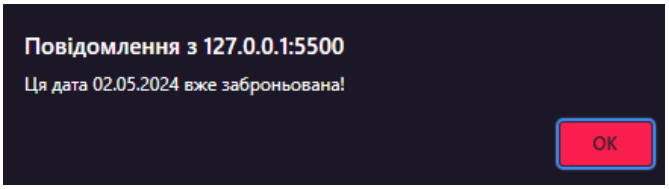


Рисунок 2.12 – Повідомлення про те, що дата вже зайнята

9) Завершення програми.

Блок-схема алгоритму роботи програми надає загальний огляд основних кроків та логіки роботи програми для бронювання житла. Вона допомагає візуалізувати послідовність дій користувача та реакцію програми на їхні дії.

2.3 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача є ключовим аспектом в розробці програмного забезпечення, оскільки саме через інтерфейс користувач взаємодіє з системою. На цьому етапі визначаються вигляд, організація та функціонал інтерфейсу з урахуванням потреб та зручності для користувача. Головною метою є створення інтуїтивно зрозумілого, зручного та привабливого інтерфейсу, який сприятиме комфортному користуванню програмою. Важливими аспектами при проєктуванні інтерфейсу є візуальна привабливість, легкість навігації, наявність зрозумілих інструкцій та швидкий доступ до необхідних функцій. Такий підхід дозволяє підвищити задоволення користувачів від використання програмного продукту та забезпечити його успішну інтеграцію на ринку.

При проєктуванні інтерфейсу користувача для програми бронювання житла, враховувалися наступні аспекти:

1) Інтуїтивність і простота використання: Інтерфейс повинен бути легким у використанні. Це включає в себе зрозумілу навігацію, послідовність дій та зрозумілість зображень і кнопок.

2) Мінімалізм: Візуальний дизайн має бути мінімалістичним, щоб не перенавантажувати користувача зайвою інформацією. Простота форм та елементів допомагає зосередитися на основній меті програми.

3) Привабливий дизайн: Естетичний вигляд інтерфейсу важливий для створення позитивного враження від користування програмою. Використання приємних кольорів, зручного шрифту та привабливих зображень забезпечує привабливість інтерфейсу.

4) Функціональність: Інтерфейс повинен надавати всі необхідні функції для бронювання житла без зайвих складнощів. Це включає в себе можливість перегляду різних варіантів проживання, вибір дат бронювання.

Враховуючи ці аспекти, проєктування інтерфейсу користувача для програми бронювання житла спрямоване на створення зручного, ефективного та привабливого інструменту для користувачів, що дозволить їм легко та швидко знаходити та бронювати житло за допомогою програми.

Для початку розглянемо макет головної сторінки застосунку (рис.2.14).

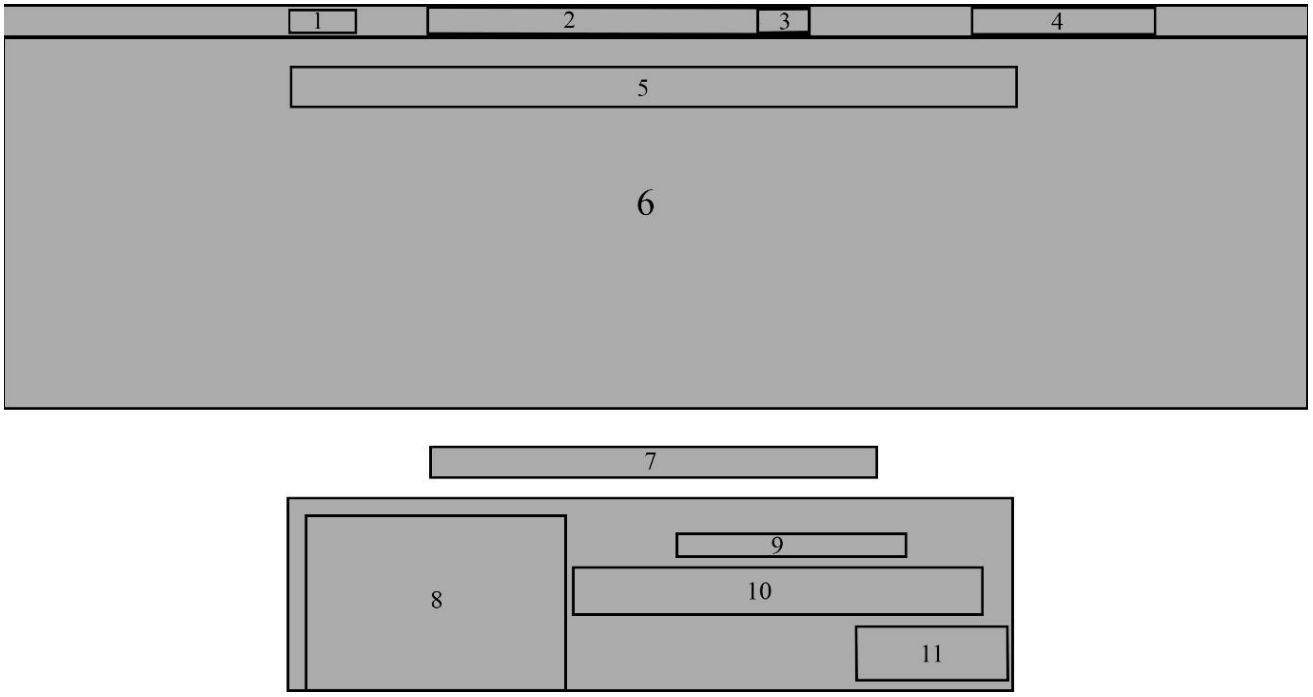


Рисунок 2.14 – Макет головної сторінки застосунку

Елементи головної сторінки:

1) Логотип;
2) Ім’я користувача;
3) Аватар користувача;
4) Кнопка (вхід/вихід/реєстрація);
5) Ключова фраза;
6) Головне фото;
7) Список доступних місць проживання;
8) Фото місця проживання;
9) Назва місця проживання;
10) Короткий опис місця проживання;
11) Кнопка перегляду кімнат в обраному місці.

На головній сторінці додатку – користувач спершу зустрічає ключову фразу, головне фото, та верхню частину сторінки. Ключову фразу було використано для залучення уваги, створення емоційного зв'язку з аудиторією та підкреслення переваг продукту. Прогорнувши сторінку трішки донизу – клієнт може побачити список, та короткі описи доступних для вибору місць проживання, які він може переглянути, натиснувши «Подивитись кімнати» навпроти житла, яке його зацікавило. Далі розглянемо сторінку житла (рис.2.15).

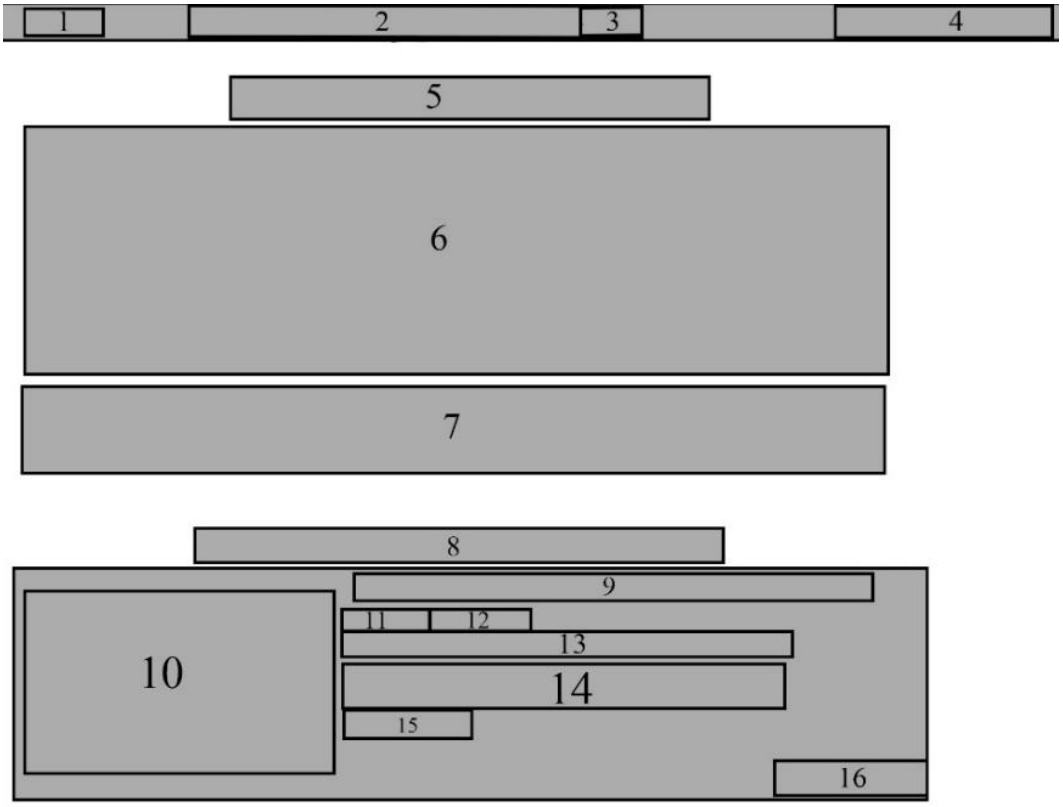


Рисунок 2.15 – Макет сторінки житла

Елементи сторінки житла:

1) Логотип;
2) Ім’я користувача;
3) Аватар користувача;
4) Кнопка (вхід/вихід/реєстрація);
5) Назва місця проживання;
6) Фото місця проживання;
7) Повний опис місця проживання;
8) Список доступних для бронювання кімнат;
9) Тип кімнати;
10) Фото кімнати;
11) Площа кімнати;
12) Кількість місць для проживання в кімнаті;
13) Доступні послуги, які надаються з обраною кімнатою;
14) Опис кімнати;
15) Ціна оренди;
16) Кнопка оренди.

На сторінці житла користувач може побачити назву, фото, та опис місця, яке він обрав, нижче він може переглянути список доступних для оренди кімнат, прочитати інформацію про кожну з них, інформація про кімнату включає в себе тип кімнати (економ, люкс, тощо), площу кімнати, кількість людей, які можуть проживати в конкретній кімнаті, короткий опис кімнати, ціну оренди і фото кімнати. Також навпроти кожної з кімнат розміщена кнопка «Орендувати», вона доступна лише для авторизованих користувачів, для не авторизованих буде напис «Щоб почати орендувати, увійдіть в аккаунт». При натисканні кнопки «Орендувати» перед користувачем з’являється календар в якому він може обрати дату, на яку б він хотів забронювати обрану кімнату.

Далі розглянемо макет сторінки адміністратора. Для входу на сторінку адміністратора нам знадобиться код доступу до цієї сторінки. Використовуючи код "213134421", ми матимемо доступ до панелі адміністратора. Макет цієї сторінки зображено на рисунку 2.16. На макеті можна побачити розміщення основних елементів управління, таких як кнопки додавання та видалення кімнат та інші важливі функції адміністрування.

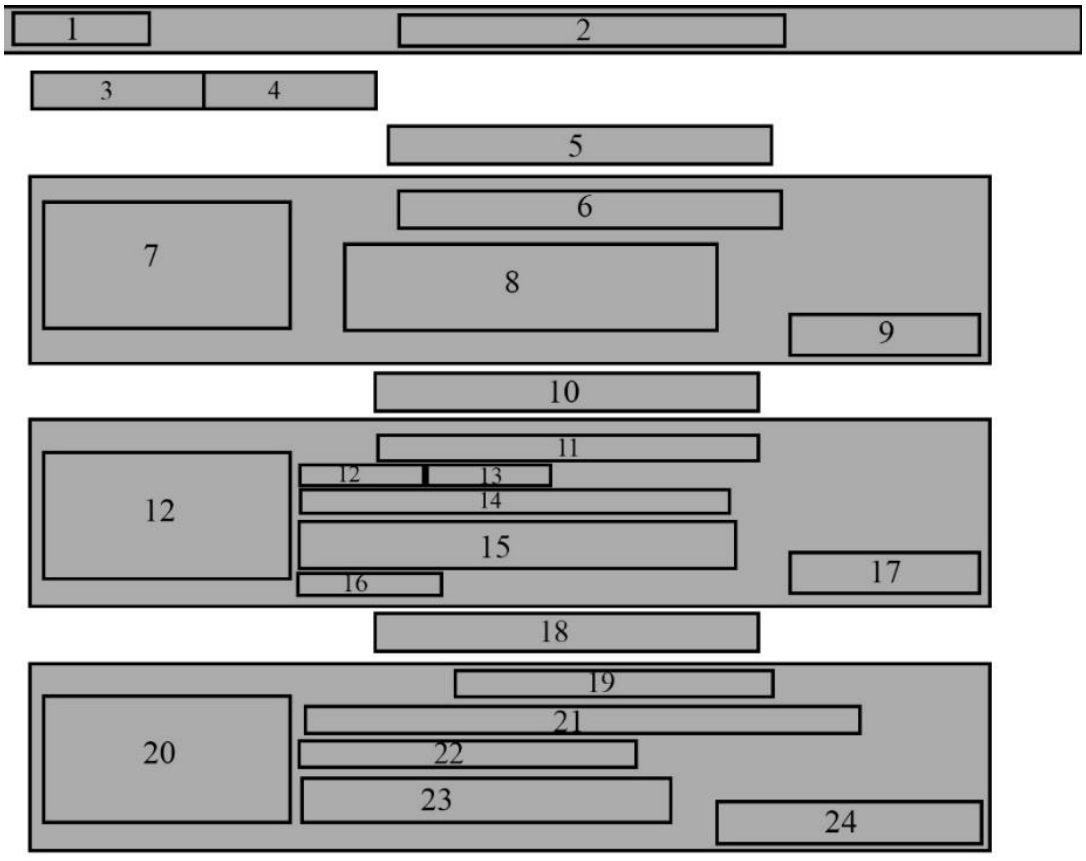


Рисунок 2.16 – Макет сторінки адміністратора

Елементи сторінки адміністратора:

1) Логотип;
2) Панель адміністратора;
3) Кнопка додавання житла;
4) Кнопка додавання кімнати;
5) Список місць проживання;
6) Назва місця проживання;
7) Фото місця проживання;
8) Опис місця проживання;
9) Кнопка видалення місця проживання;
10) Список доступних кімнат;
11) Тип кімнати;
12) Площа кімнати;
13) Кількість місць для проживання в кімнаті;
14) Доступні послуги, які надаються з обраною кімнатою;
15) Опис кімнати;
16) Ціна оренди;
17) Кнопка видалення кімнати;
18) Список орендованих кімнат;
19) Тип орендованої кімнати;
20) Фото орендованої кімнати;
21) Дані користувача, який орендував кімнату (ім’я, електронна пошта);
22) Дата бронювання та ціна;
23) Опис орендованої кімнати;
24) Кнопка відміни оренди.

Спершу адміністратор може побачити список місць проживання, які розміщенні на головній сторінці додатку, їх опис, та кнопку, яка відповідає за видалення місця навпроти якого вона знаходиться. Прогорнувши сторінку нижче – адміністратор може побачити список всіх доступних кімнат, повну інформацію про них (площа, опис, ціна, список послуг, тощо.), навпроти кожної кімнати також розміщена кнопка, яка відповідає за видалення відповідної кімнати. Прогорнувши ще нижче – можна побачити список кімнат, які були орендовані

користувачами, та інформацію яка містить фото, та опис кімнати, а також дані клієнта, такі як ім’я, та пошту, для зв’язку з клієнтом, та подальшого вирішення питань з оренди. Також навпроти кожної з орендованих кімнат є кнопка, яка відповідає за відміну оренди. При будь-якому видаленні потрібно буде ще раз підтвердити видалення, натиснувши «Ок» у випливаючому повідомленні, це зроблено для того, щоб адміністратор не міг щось видалити випадково. Одними з основних функцій адміністратора є додавання житла, та додавання кімнат. Кнопки, які відповідають за ці функції розміщені у верхній частині адміністративної панелі. Розглянемо функцію додавання на прикладі додавання кімнати (рис 2.17).

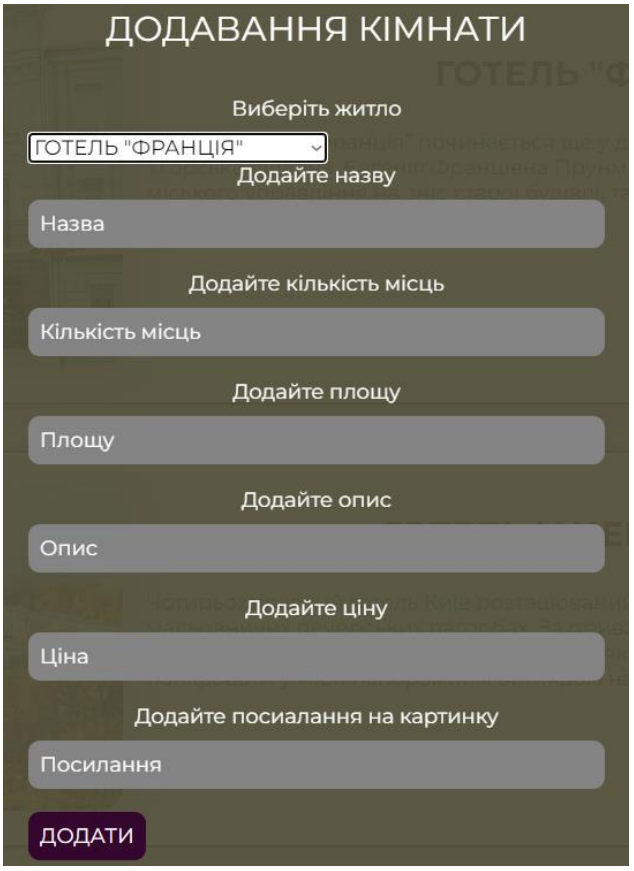


Рисунок 2.17 – Додавання кімнати

Для додавання кімнати – першим кроком обираємо житло в яке і будемо додавати кімнату, далі вказуємо назву (тип) кімнати, вказуємо кількість місць (кількість людей, які можуть жити в даній кімнаті), площу кімнати, додаємо опис, вказуємо ціну, та додаємо посилання на фото кімнати, натискаємо «Додати», та для підтвердження вводимо код адміністратора.

Проєктування інтерфейсу користувача є ключовим етапом в розробці програмного забезпечення, оскільки він визначає спосіб взаємодії користувача з системою. Для веб-додатку для бронювання житла враховано різноманітні аспекти, включаючи інтуїтивність використання, мінімалізм у дизайні, адаптивність до різних браузерів, привабливий вигляд та функціональність. Головна мета полягає у створенні зручного, ефективного та привабливого інтерфейсу, який сприятиме комфортному користуванню застосунком. Ретельне проєктування дозволяє створити інструмент, який задовольняє потреби користувачів та сприяє їх задоволенню від використання.

У процесі проєктування інтерфейсу користувача важливо врахувати потреби та очікування цільової аудиторії. Це може включати розміщення основних функціональних елементів на видимих місцях, використання зрозумілих піктограм та символів, а також інтуїтивність взаємодії з додатком.

Проєктування інтерфейсу також передбачає вивчення та впровадження кращих практик веб-дизайну, які включають в себе використання чіткого та консистентного шрифту для забезпечення зрозумілості текстового контенту. Крім того, доцільно враховувати колірну палітру, яка не лише приваблює, а й сприяє візуальній зручності та збереженню уніфікованого стилю. Правильне розміщення елементів на сторінці також має велике значення, оскільки воно визначає логічність та зручність навігації для користувача.

Загалом, ретельне проєктування інтерфейсу користувача відіграє важливу роль у створенні програмного забезпечення, яке не лише ефективно виконує свої функції, але й приносить задоволення користувачам від взаємодії з ним. Це включає в себе створення інтуїтивно зрозумілого та легкого в управлінні інтерфейсу, який сприяє зручному та приємному користуванню програмою. Такий підхід допомагає покращити загальний досвід користувача та забезпечити позитивне сприйняття програми.

2.4 Висновки

Під час проєктування системи було проведено детальний аналіз та розробка ключових аспектів програмного забезпечення для бронювання житла. Починаючи з визначення архітектури системи, було обрано трьохрівневу

архітектуру, яка включає фронтенд, бекенд та сервер баз даних. Ця архітектура забезпечує ефективну розробку та масштабування програми з великим обсягом функцій та даних. Було також розроблено блок-схему алгоритму роботи програми для бронювання житла, що допомагає візуалізувати послідовність дій користувача та реакцію програми на їхні дії.

Проєктування інтерфейсу користувача було здійснено з урахуванням різноманітних аспектів, таких як інтуїтивність використання, мінімалізм у дизайні, адаптивність до різних пристроїв, привабливий вигляд та функціональність. Головною метою було створення зручного, ефективного та привабливого інтерфейсу, який сприятиме комфортному користуванню програмою. Для цього було розроблено макети головної сторінки, сторінки житла та сторінки адміністратора, які детально розглянули всі елементи взаємодії користувачів, та адміністраторів з контентом та забезпечили зручну та інтуїтивно зрозумілу навігацію.

У процесі проєктування системи було розроблено концептуальний та технічний плани програмного забезпечення для бронювання житла. Ці плани враховують потреби користувачів та забезпечують їм зручне та ефективне використання програми. Результатом цього процесу є програмне забезпечення, яке відповідає вимогам та очікуванням користувачів, дозволяючи їм з легкістю користуватися всіма його можливостями.

3. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Аналіз технологій та методів для розробки програмного забезпечення

Розглянемо технології та методи, що були використані в процесі розробки програмного забезпечення для бронювання житла, та порівняємо їх з їх можливими аналогами.

1) Створення структури.

- Обраний метод: HTML (Hypertext Markup Language): HTML використовується для створення структури та контенту веб-сторінок. У контексті розробки програмного забезпечення для бронювання житла, HTML використовується для створення користувацького інтерфейсу, включаючи форми бронювання та відображення результатів.
- Можливим аналогом HTML може бути Markdown - це легкий текстовий формат для написання веб-сторінок. Він зазвичай використовується для написання документації, блогів, коментарів у вихідних файлах Git і багатьох інших сценаріїв. Markdown є простим у використанні, має читабельний синтаксис і може бути легко конвертований в HTML для відображення веб-сторінок.

Порівняння HTML і Markdown зображено в таблиці 3.1.

Таблиця 3.1 – Порівняння HTML і Markdown

Критерій	HTML	Markdown
Використання	Веб-розробка	Створення документів
Функціональність	Багато можливостей	Обмежений функціонал
Розширюваність	Можливість використання CSS та JavaScript для покращення стилів та функціональності	Обмежена можливість розширення

2) Препроцесор.

- Обраний метод: SCSS (Sass CSS): SCSS є розширенням CSS, яке дозволяє використовувати змінні, вкладені стилі та інші покращення для зручності розробки стилів веб-сторінок. Використання SCSS дозволяє ефективно організувати та керувати стилями програмного забезпечення для бронювання житла, забезпечуючи їхню консистентність та легкість зміни.
- Можливим аналогом SCSS може бути Less - це препроцесор CSS, що дозволяє зробити CSS код більш організованим та легшим у керуванні. Він додає функціонал, такий як змінні, міксіни, вкладеність та інші можливості до звичайного CSS, що полегшує розробку та підтримку стилів.

Порівняння SCSS і Less зображено в таблиці 3.2.

Таблиця 3.2 – Порівняння SCSS і Less

Критерій	SCSS	Less
Структура коду	Підтримка вкладеності, змінних та міксінів для покращення читабельності та управління стилями	Підтримка вкладеності, але менше можливостей для роботи зі змінними та міксінами
Сумісність	Підтримується більшістю браузерів та інтерпретаторами CSS	Потребує компіляції у CSS перед використанням для більшості браузерів
Розширюваність	Є можливість використання плагінів та розширень для роботи зі стилями	Обмежена можливість розширення за допомогою плагінів та розширень

3) Динамічне оновлення вмісту.

- Обраний метод: JavaScript (JS): використовується для реалізації динамічної

поведінки веб-сторінок, включаючи обробку подій, взаємодію з користувачем та маніпулювання вмістом сторінки. У програмному забезпеченні для бронювання житла, JavaScript використовується для валідації форм, асинхронного завантаження даних та взаємодії з користувачем.

- Можливим аналогом JavaScript може бути TypeScript - це сучасна мова програмування, що розширює JavaScript додаванням строго типізації та інших функцій, що полегшують розробку великих та складних проектів. TypeScript дозволяє виявляти та усувати помилки під час розробки, підвищуючи при цьому безпеку та стабільність програм.

Порівняння JavaScript і TypeScript зображено в таблиці 3.3.

Таблиця 3.3 – Порівняння JavaScript і TypeScript

Критерій	JavaScript	TypeScript
Типізація	Не строго типізована	Строго типізована
Код-компіляція	Виконується безпосередньо в браузері	Потребує компіляції до JavaScript перед виконанням
Функціональність	Можливість використання асинхронного програмування, маніпуляція DOM, обробка подій тощо	Розширює функціональність JavaScript, додаючи статичну типізацію, інтерфейси, перечислення тощо
Підтримка	Широко підтримується у браузерах та середовищах виконання	Потребує додаткової обробки для сумісності з усіма браузерами та середовищами виконання

4) База даних.

- Обраний метод: Firebase - це платформа Google для розробки мобільних та

веб-додатків, яка надає різноманітні сервіси, включаючи базу даних в реальному часі, аутентифікацію користувачів, зберігання файлів та інші. У контексті програмного забезпечення для бронювання житла, Firebase може використовуватися для зберігання та управління даними про користувачів, готелі, бронювання та інші аспекти додатка.

- Можливим аналогом Firebase може бути AWS Amplify - це сервіс від Amazon Web Services, який дозволяє розробникам швидко розгортати та масштабувати веб-застосунки та мобільні додатки. Він надає інструменти для автоматизації процесів розробки, такі як розгортання коду, керування інфраструктурою та інтеграція з іншими сервісами AWS.

Порівняння Firebase і AWS Amplify зображено в таблиці 3.4.

Таблиця 3.4 - Порівняння Firebase і AWS Amplify

Критерій	Firebase	AWS Amplify
Швидкість	Швидке розгортання та робота з даними в реальному часі	Швидке розгортання та висока швидкість роботи
Складність	Легке налаштування та використання	Може виглядати складніше у порівнянні з Firebase
Функціональність	Забезпечує бази даних в реальному часі, аутентифікацію, зберігання файлів тощо	Надає широкий спектр сервісів для розробки та розгортання додатків
Вартість	Безкоштовний тарифний план та різноманітні тарифи для підприємств	Тарифи можуть бути більш вигідними для великих проектів

Аналіз технологій та методів для розробки програмного забезпечення в сфері бронювань помешкань демонструє, що використання HTML, SCSS, JS та Firebase є ефективними та раціональними в контексті розробки програмного

забезпечення для бронювання житла. HTML та SCSS дозволяють створювати структуру та візуальний дизайн веб-сторінок з великою гнучкістю та можливістю керування, тоді як JS забезпечує динаміку та інтерактивність, необхідну для забезпечення зручності та ефективності користувачів. Firebase, з своїми різноманітними сервісами, допомагає зосередитися на функціональності додатку, не витрачаючи час на налаштування та управління серверною інфраструктурою. Такий підхід сприяє швидкому впровадженню та підтримці програмного забезпечення, що відповідає сучасним вимогам користувачів.

3.2 Розробка алгоритмів реєстрації та авторизації

Розглянемо процес розробки алгоритмів реєстрації та авторизації користувачів для програмного додатку. Алгоритм реєстрації включає в себе створення нового облікового запису користувача, а авторизація - перевірку прав доступу до додатку.

У першу чергу, для розробки алгоритмів реєстрації та авторизації необхідно визначити процедури, які будуть виконуватися при взаємодії користувача з додатком. Алгоритм реєстрації включає в себе такі кроки, як введення користувачем даних (наприклад, електронної пошти, пароля тощо), перевірку їх на валідність, додавання нового облікового запису в базу даних.

Під час розробки алгоритмів необхідно також враховувати вимоги до безпеки, щоб запобігти можливим атакам на облікові записи користувачів, таким як злам пароля чи використання недозволених методів авторизації.

Розглянемо блок-схеми алгоритмів реєстрації та авторизації. Спершу розглянемо блок-схему алгоритму реєстрації (рис. 3.1).

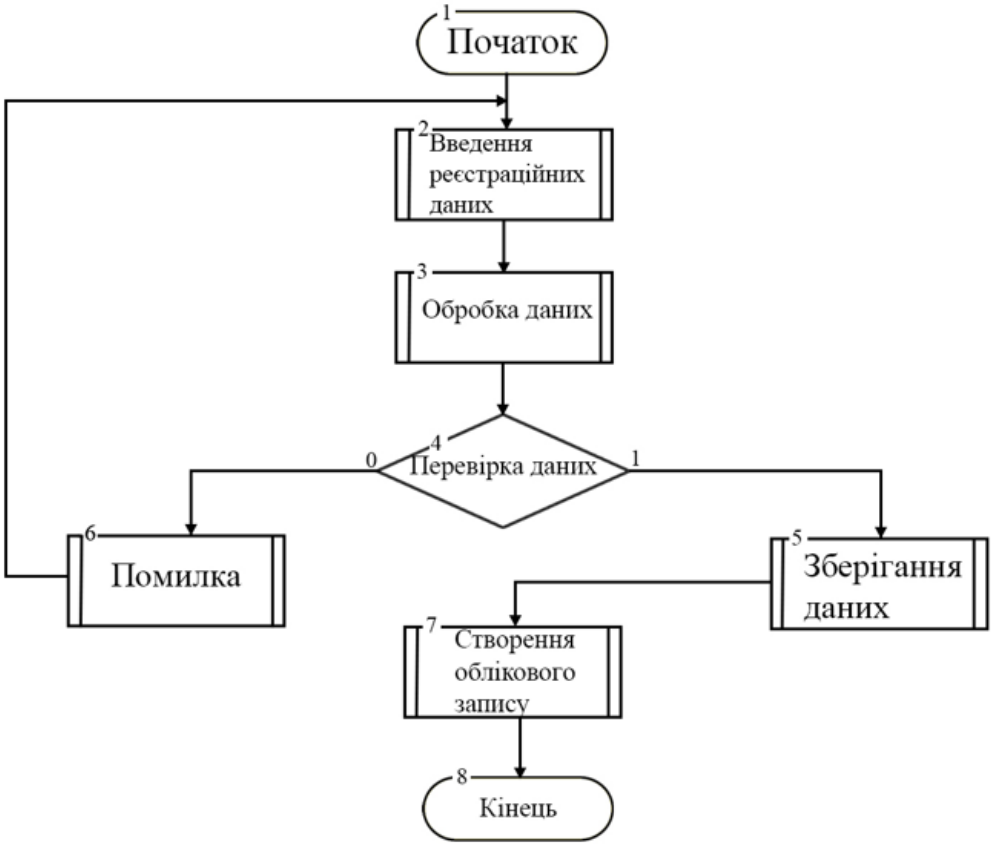


Рисунок 3.1 – блок-схема алгоритму реєстрації

1) Для початку реєстрації – користувач повинен відкрити вікно входу/реєстрації, що знаходиться в правій верхній частині інтерфейсу.
2) Далі, користувач повинен заповнити реєстраційні дані і натиснути «Зареєструватись» (рис. 3.2).

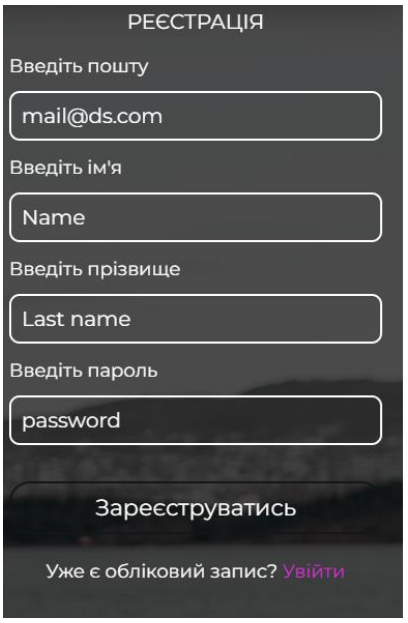


Рисунок 3.2 – Вікно реєстрації

3) Потім відбувається обробка введених користувачем даних.
4) Наступним кроком – програма перевіряє коректність введених даних, а також, чи такого користувача немає в базі даних.
5) Якщо ж результат перевірки позитивний – програма зберігає введені користувачем дані до бази даних з користувачами (рис 3.3).

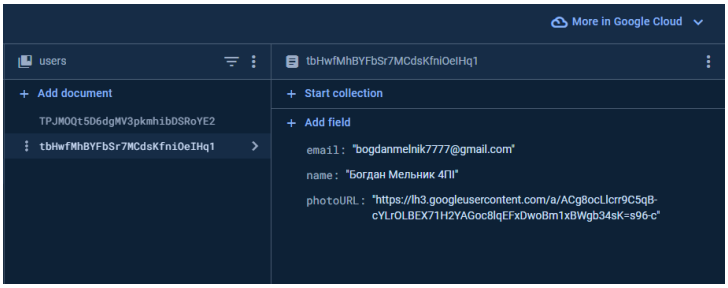


Рисунок 3.3 – База даних з користувачами

6) Якщо результат перевірки негативний, то користувач повинен перевірити введені дані, та ввести їх коректно.
7) Після успішної перевірки даних, та зберігання їх в базі даних – програма створює новий обліковий запис користувача.
8) Завершення процесу реєстрації.

Далі розглянемо блок-схему алгоритму авторизіції, вона зображена на рисунку 3.4.

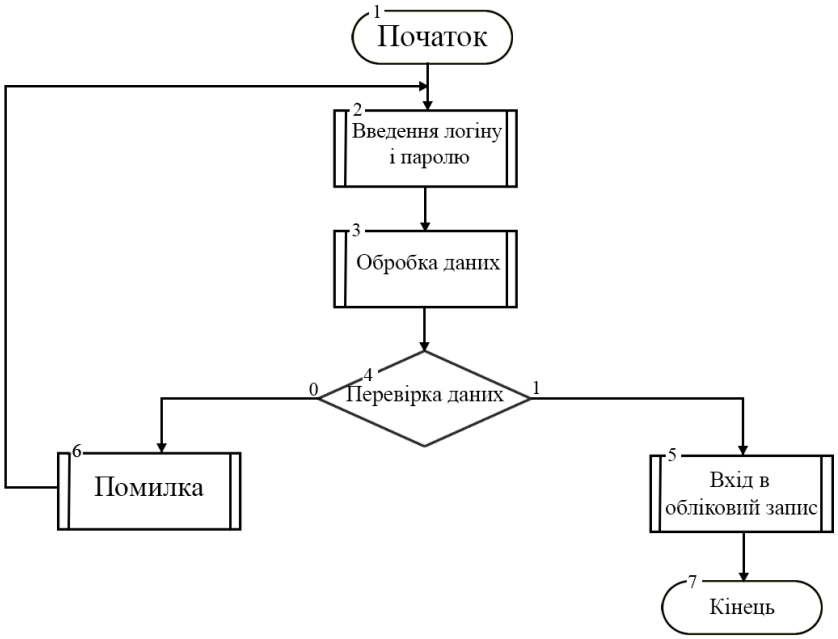


Рисунок 3.4 – блок-схема алгоритму авторизації

1) Для початку входу в обліковий запис – користувач повинен відкрити вікно входу/реєстрації, що знаходиться в правій верхній частині інтерфейсу, та натиснути «Увійти», в нижній частині вікна реєстрації.
2) Далі, користувач повинен заповнити дані для входу і натиснути «Увійти» (рис. 3.5).

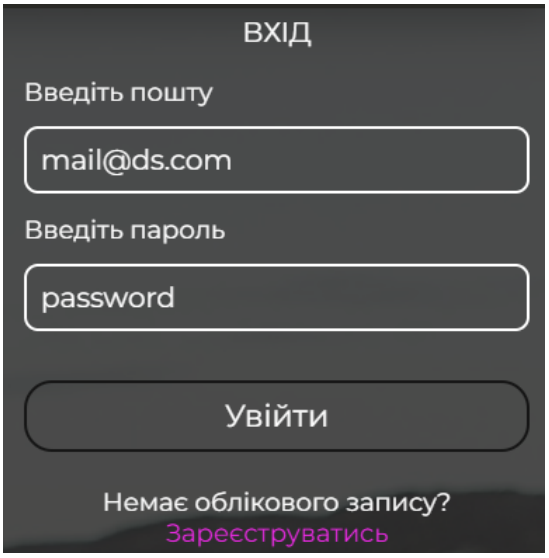


Рисунок 3.5 – Вікно входу до системи

3) Потім відбувається обробка введених користувачем даних.
4) Наступним кроком – програма перевіряє коректність введених даних.
5) Якщо результат перевірки позитивний – програма виконує вхід до облікового запису користувача.
6) Якщо ж результат перевірки негативний, то користувач повинен перевірити введені дані, виправити їх та повторити спробу входу.
7) Завершення процесу авторизації.

Реалізація алгоритмів реєстрації та авторизації користувачів є важливим етапом у розробці програмного додатку. Завдання алгоритму реєстрації полягає у створенні нового облікового запису користувача, в той час як авторизація включає перевірку прав доступу до додатку.

Першим кроком у реєстрації є введення користувачем необхідних даних та їх подання на обробку. Далі програма перевіряє правильність введених даних та перевіряє, чи вже існує користувач з такими даними в базі даних. Якщо дані введені коректно та обліковий запис ще не створений, програма зберігає новий

обліковий запис у базу даних. У разі вдалого входу до системи, користувач має доступ до функціоналу додатку.

3.3 Розробка алгоритму додавання кімнат

Розробка алгоритму додавання кімнат включає в себе процес створення та додавання нових приміщень до системи за допомогою програмного додатку. Цей алгоритм визначає послідовність кроків, необхідних для успішного додавання нової кімнати та відображення її у програмі.

Блок-схема алгоритму додавання кімнати зображена на рисунку 3.6.

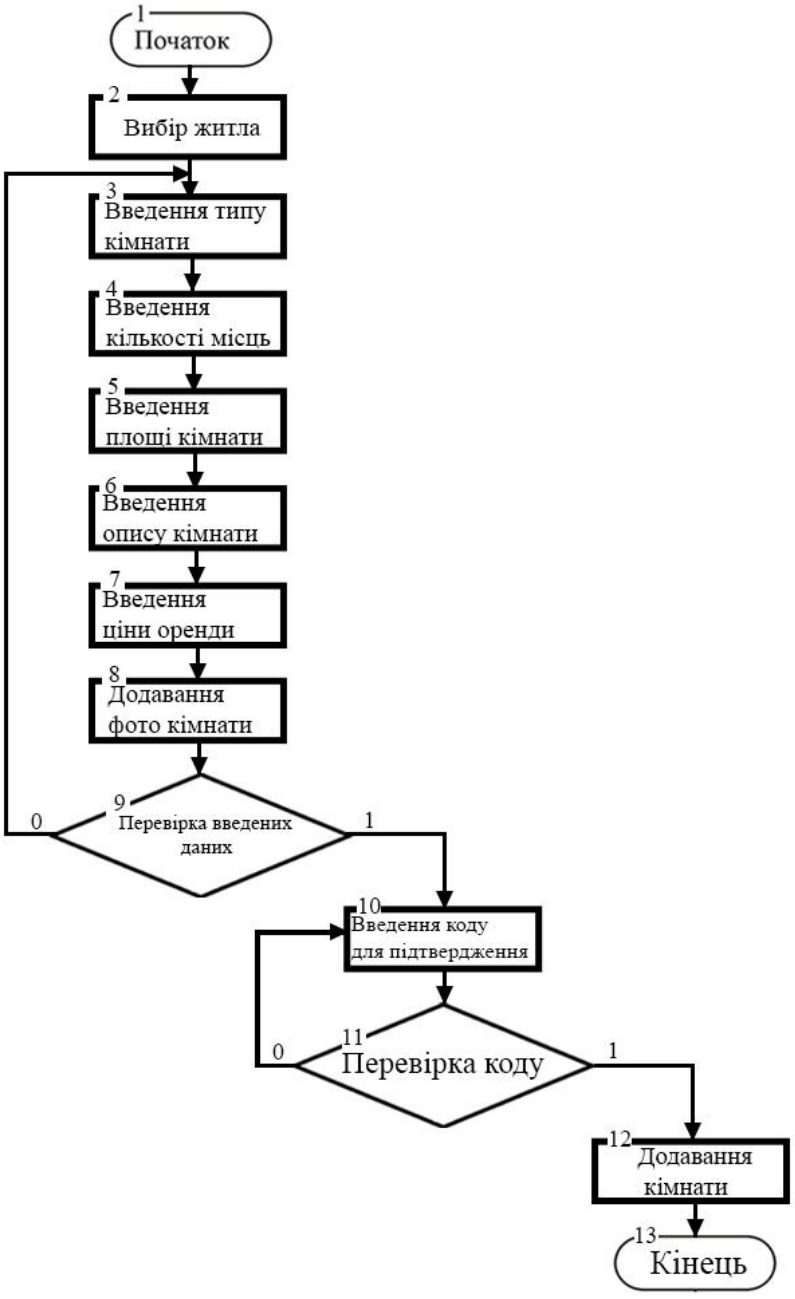


Рисунок 3.6 – Блок схема алгоритму додавання кімнати

1) Початок.
2) Для початку – адміністратор повинен обрати житло, в яке потрібно додати кімнату.
3) Ввести тип кімнати (економ, люкс, тощо).
4) Ввести кількість місць передбачену для проживання в кімнаті.
5) Ввести площу кімнати.
6) Добавити опис кімнати.
7) Ввести ціну оренди кімнати.
8) Завантажити фото кімнати.
9) Далі програма перевіряє коректність введених даних, якщо данні введені не вірно, або якесь з полів не заповнено – адміністратор повинен це виправити.
10) Після успішної перевірки потрібно підтвердити додавання кімнати, ввівши код адміністратора в полі, яке з’явиться (рис. 3.7).

Рисунок 3.7 – Поле вводу коду доступу

11) Далі відбувається перевірка коду. Якщо код введено не вірно – його потрібно буде ввести повторно.
12) Після успішної перевірки коду відбувається додавання кімнати.
13) Завершення процесу додавання кімнати.

Блок-схема алгоритму відображає процес додавання кімнати, розкривши кожен етап, починаючи з обрання житла, введення характеристик кімнати, завантаження фотографії та завершення підтвердженням додавання.

Важливим елементом цього процесу є контроль коректності введених даних та відповідність коду доступу, що забезпечує безпеку та відповідність дій адміністратора.

Завершення процесу додавання кімнати підтверджує успішне додавання

нового приміщення до системи, що дозволяє користувачам програмного додатку здійснювати бронювання вибраних приміщень.

3.4 Висновки

Під час розробки програмного додатку було зосереджено увагу на ключових аспектах створення програмного забезпечення для бронювання житла, включаючи аналіз технологій та методів, розробку алгоритмів реєстрації та авторизації, а також алгоритму додавання кімнат.

Аналіз технологій та методів розкриває використання HTML, SCSS, JavaScript та Firebase для створення користувацького інтерфейсу, забезпечення динамічного оновлення вмісту та управління базою даних. Ці технології вибрані з урахуванням їхньої ефективності та можливостей і відображають сучасні підходи до розробки веб-додатків.

Розроблені алгоритми реєстрації та авторизації забезпечують безпеку та доступ користувачів до функціоналу додатку.

Алгоритм додавання кімнат розкриває процес створення та додавання нових приміщень до системи. Він включає всі необхідні кроки, починаючи від введення характеристик кімнати до підтвердження додавання за допомогою коду доступу, що забезпечує простоту використання та ефективне управління приміщеннями.

Використання сучасних інструментів та ретельна розробка алгоритмів дозволяє створювати функціональний та безпечний додаток, що задовольняє потреби користувачів.

4. ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Аналіз методів тестування веб-додатку

Планування тестування є ключовим етапом у забезпеченні якості програмного забезпечення. Воно включає визначення цілей, методологій, ресурсів і графіку тестування, що забезпечують систематичний підхід до виявлення та виправлення дефектів у додатку.

Мета тестування.

Основна мета тестування програмного додатку полягає у виявленні можливих помилок та недоліків до того, як додаток буде впроваджений у реальне середовище використання. Тестування також допомагає переконатися, що додаток відповідає функціональним та нефункціональним вимогам, визначеним на етапі розробки.

1) Функціональне тестування:
   - Тестування користувацького інтерфейсу (UI Testing): Перевірка взаємодії користувача з елементами інтерфейсу, такими як кнопки, форми, меню, для забезпечення їх правильної роботи.
   - Тестування форм: Перевірка коректності роботи форм, включаючи валідацію введених даних та правильність обробки форм.
   - Тестування навігації: Перевірка правильності навігації між сторінками та розділами додатку.
2) Інтеграційне тестування:
   - Тестування взаємодії між компонентами: Перевірка коректності взаємодії між різними модулями та компонентами додатку.
   - Тестування бази даних: Перевірка взаємодії додатку з базою даних, включаючи запити, цілісність даних та коректність збереження і відновлення даних.
3) Тестування продуктивності:
   - Навантажувальне тестування (Load Testing): Перевірка здатності додатку справлятися з певною кількістю одночасних користувачів або запитів.
   - Стрес-тестування (Stress Testing): Перевірка поведінки додатку під екстремальним навантаженням, з метою визначення, при яких умовах

додаток почне некоректно працювати або виходити з ладу.

- Тестування масштабованості (Scalability Testing): Оцінка здатності додатку збільшувати свою продуктивність при збільшенні ресурсів або навантаження.

4) Тестування безпеки:
   - Виявлення уразливостей (Vulnerability Scanning): Використання інструментів для автоматизованого пошуку відомих уразливостей у додатку.
   - Тестування автентифікації та авторизації: Перевірка механізмів входу до системи та управління правами доступу.
   - Тестування захисту даних (Data Security Testing): Перевірка методів шифрування, захисту даних під час передачі та зберігання.
5) Крос-браузерне та крос-платформне тестування:
   - Тестування сумісності з різними браузерами: Перевірка коректності роботи додатку в різних браузерах (Chrome, Firefox, Safari, Edge тощо).
   - Тестування на різних платформах: Перевірка роботи додатку на різних операційних системах (Windows, macOS, Linux) та мобільних пристроях (Android, iOS).
6) Тестування зручності використання (Usability Testing):
   - Тестування з реальними користувачами: Оцінка додатку на групі реальних користувачів для визначення зручності його використання.
7) Тестування адаптивності (Responsive Testing):
   - Тестування на різних розмірах екранів: Перевірка коректності відображення додатку на різних розмірах екранів та пристроях з різними роздільними здатностями.
8) Тестування дотримання стандартів:
   - Тестування відповідності до веб-стандартів: Перевірка відповідності додатку встановленим стандартам розробки веб- додатків.

Також, популярними методами тестування програмного забезпечення, які використовуються для забезпечення якості та надійності додатків є наступні методи:

1) Тестування "чорної скриньки" (Black Box Testing):

- Опис: Цей метод тестування зосереджується на функціональності додатку без знання його внутрішньої структури або коду. Тестувальники оцінюють додаток на основі вхідних даних та очікуваних вихідних результатів.
- Приклади: Тестування користувацьких інтерфейсів, функціональності форм, навігації, поведінки системи під час виконання різних дій.
- Переваги: Не вимагає знання коду; тестування може бути виконане з точки зору користувача; допомагає виявляти недоліки у функціональності та використанні додатку.
- Недоліки: Може пропустити внутрішні помилки та неефективність коду; складно повністю покрити всі можливі сценарії.

2) Тестування "білої скриньки" (White Box Testing):
   - Опис: Цей метод тестування вимагає знання внутрішньої структури та коду додатку. Тестувальники мають доступ до коду і тестують логіку, контрольні структури, умови, цикли та шляхи виконання.
   - Приклади: Юніт-тестування, тестування потоків даних, тестування логіки умов і циклів, покриття коду.
   - Переваги: Дозволяє виявляти приховані помилки та уразливості в коді; забезпечує високе покриття тестами; допомагає оптимізувати та поліпшити код.
   - Недоліки: Вимагає знання програмування; може бути складним для великих та комплексних систем; не виявляє проблем з використанням додатку з боку користувачів.
3) Тестування "сірої скриньки" (Gray Box Testing):
   - Опис: Цей метод тестування є комбінацією тестування "чорної" та "білої" скриньок. Тестувальники мають обмежене знання внутрішньої структури додатку, що дозволяє їм тестувати систему з обох точок зору: функціональності та внутрішньої логіки.
   - Приклади: Тестування баз даних з розумінням структури таблиць, але без повного знання SQL запитів; тестування API з обмеженим доступом до коду.
   - Переваги: Поєднує переваги обох підходів; дозволяє виявляти недоліки

як у функціональності, так і у внутрішній логіці; ефективний для складних систем.

- Недоліки: Може бути складним у плануванні та виконанні; вимагає як технічних знань, так і розуміння користувацьких вимог; може пропустити деякі внутрішні або функціональні помилки.

Під час аналізу методів тестування веб-додатку було розглянуто різноманітні підходи до забезпечення якості програмного забезпечення. Ці методи дозволяють виявити можливі помилки та недоліки у додатку до його впровадження в реальне середовище використання. Функціональне тестування, інтеграційне тестування, тестування продуктивності, безпеки, крос-браузерне та крос-платформенне тестування, тестування зручності використання - це лише деякі з методів, які можуть бути використані для забезпечення якості веб-додатку.

Застосування цих методів дозволяє виявити та виправити потенційні проблеми ще на етапі розробки, забезпечуючи високу якість та надійність продукту. Аналіз цих методів є важливою передумовою для успішної реалізації стратегії тестування та підвищення конкурентоспроможності веб-додатку на ринку.

4.2 Тестування користувацької частини

Тестування користувацької частини веб-додатку є важливим етапом у забезпеченні задоволення потреб користувачів та оптимального функціонування програми. Цей процес включає перевірку всіх аспектів взаємодії користувача з додатком, починаючи від реєстрації та авторизації, і закінчуючи процесом оренди житла. Для досягнення високої якості продукту необхідно виконати серію тестів, які охоплюють всі можливі сценарії використання додатку.

Одним з ключових аспектів тестування є перевірка функціональності основних елементів інтерфейсу та їх відповідність очікуванням користувачів. Наприклад, перевірка коректності відображення контенту на головній сторінці, робота з формами реєстрації та авторизації, а також перевірка процесу бронювання.

Першим кроком протестуємо коректність відображення інформації на

головній сторінці (табл. 4.1).

Таблиця 4.1 – Тест 1

Тестування головної сторінки		
Дії	Очікуваний результат	Дійсний результат
Відкрити головну сторінку веб-додатку	Весь контент (кнопки, графічна, та текстова інформація) відображається коректно	Контент відображається коректно
Переглянути місця проживання	Користувач може перейти до перегляду кімнат	Перехід здійснено успішно

На головній сторінці користувач може побачити кнопку авторизації та реєстрації, що розміщена у верхній частині сторінки, а також перейти до перегляду кімнат місця проживання, яке його зацікавило.

Далі протестуємо реєстрацію, та авторизацію. Створимо новий обліковий запис в системі (табл.4.2).

Таблиця 4.2 – Тест 2

Тестування реєстрації нового користувача		
Дії	Очікуваний результат	Дійсний результат
Відкрити вікно реєстрації	Відображення форми для введення даних нового користувача	Форма відображається коректно
Ввести дані та натиснути кнопку "Реєстрація"	Новий користувач створюється, дані зберігаються в базу даних	Акаунт користувача створено успішно

Наступним кроком у процесі тестування буде перевірка процедури авторизації для раніше створеного облікового запису, щоб гарантувати, що користувачі можуть успішно входити в систему. Це включає перевірку коректного відображення форми для введення даних, успішність авторизації при введенні правильних даних та надання відповідних прав доступу користувачу після входу. Це допоможе переконатися, що процес авторизації є надійним та безпечним для всіх користувачів. Тестування процесу авторизації користувача відображено в таблиці 4.3.

Таблиця 4.3 – Тест 3

Тестування авторизації користувача		
Дії	Очікуваний результат	Дійсний результат
Відкрити сторінку авторизації	Відображення форми для введення даних для авторизації	Форма відображається коректно
Ввести дані та натиснути кнопку "Увійти"	Користувач успішно авторизується, користувачеві надається можливість оренди	Авторизація пройдена успішно, можливість оренди доступна

Наступним кроком протестуємо коректність роботи оренди, що є ключовим функціоналом веб-додатку. Це забезпечує користувачам можливість перегляду доступних кімнат, перевірки їх наявності та здійснення бронювання в зручний та безпечний спосіб. Під час тестування перевіримо, чи відображається інформація про житло та кімнати коректно, чи правильно працює календар, та чи відображаються зайняті й вільні дати. Окрім того, переконаємося, що після обрання бажаної дати система коректно оновлює статус кімнати в календарі. Тестування функції оренди відображено в таблиці 4.4.

Таблиця 4.4 – Тест 4

Тестування роботи оренди		
Дії	Очікуваний результат	Дійсний результат
Відкрити сторінку житла, перейти до списку кімнат	Інформація про житло, та кімнати відображається коректно	Інформація відображається коректно
Навпроти бажаної кімнати та натиснути кнопку «Орендувати»	Відкриття календаря, та відображення зайнятих і вільних дат	Успішне відображення інформації в календарі
Обрати бажану вільну дату, та натиснути «Орендувати»	Обрана дата в календарі відображається як зайнята, та з’являється повідомлення про успішну оренду	Оренда пройшла успішно

Тестування функціоналу оренди показало, що користувачі можуть безперешкодно переглядати інформацію про житло, перевіряти доступність кімнат на обрані дати та здійснювати бронювання. Усі перевірки підтвердили, що система працює коректно, забезпечуючи надійність і зручність для кінцевих користувачів.

Тестування користувацької частини веб-додатку є важливим етапом у забезпеченні його якості та конкурентоспроможності на ринку. Під час тестування були перевірені всі основні функціональні можливості, включаючи реєстрацію, авторизацію та оренду житла. При тестуванні користувацької частини помилок виявлено не було, що свідчить про високу якість розробки та стабільність роботи додатку. Тестування проводилось з урахуванням реальних сценаріїв використання, що дозволило максимально наблизити умови до експлуатаційних.

4.3 Тестування адміністративної частини

Тестування адміністративної частини веб-додатку є важливим етапом для забезпечення його надійності, безпеки та зручності використання.

Адміністративна частина включає в себе функціонал для управління контентом, налаштуваннями системи та іншими критичними аспектами, тому її тестування потребує особливої уваги.

Адміністративна частина веб-додатку відіграє ключову роль у забезпеченні ефективного функціонування системи та задоволення потреб адміністраторів. Вона надає можливості для управління контентом, модерації користувачів, аналізу даних та іншого.

Проведення тестування дозволить виявити та виправити потенційні проблеми адміністративної частини, що можуть виникнути під час реального використання веб-додатку. Такий підхід допоможе підвищити надійність, безпеку та зручність користування системою як для адміністраторів, так і для звичайних користувачів.

Першим етапом у процесі тестування буде перевірка коректності виконання процедури входу в панель адміністратора веб-додатку. (табл. 4.5)

Таблиця 4.5 – Тест 5

Тестування входу в панель адміністратора		
Дії	Очікуваний результат	Дійсний результат
Відкрити сторінку входу в панель адміністратора	Відображення поля вводу коду адміністратора	Поле відображається коректно
Ввести неправильний код доступу	Відображається повідомлення про неправильний код доступу	Повідомлення відображається коректно
Ввести правильний код доступу	Відображається повідомлення про успішний вхід, та відкриття панелі адміністратора	Успішний вхід в панель адміністратора

Наступним кроком буде перевірка правильності та точності відображення всієї необхідної інформації на сторінці адміністратора, включаючи графічні елементи та функціональні кнопки. Цей етап включатиме перевірку візуального вигляду сторінки, переконання, що всі елементи розташовані та відображаються належним чином, а також перевірку взаємодії з кнопками та іншими елементами управління, яка має бути безперебійною і інтуїтивно зрозумілою для адміністратора. Під час цього етапу також буде звернута увага на забезпечення консистентності дизайну та забезпечення відповідності стандартам веб-розробки і дизайну програмного інтерфейсу користувача. Тестування сторінки адміністратора відображено в таблиці 4.6.

Таблиця 4.6 – Тест 6

Тестування сторінки адміністратора		
Дії	Очікуваний результат	Дійсний результат
Виконати вхід в панель адміністратора	Успішний вхід	Вхід виконано успішно
Переглянути контент панелі адміністратора	Уся графічна інформація і кнопки відображаються коректно	Вся інформація відображається коректно

Наступним етапом тестування буде перевірка функціональності, а також коректної роботи опції додавання нової кімнати, щоб переконатися в її належному виконанні. Цей процес включатиме перевірку відображення нової кімнати на веб-сторінці для користувачів та у відповідному розділі в панелі адміністратора. Тестування додавання кімнати відображено в таблиці 4.7.

Таблиця 4.7 – Тест 7

Тестування функції додавання кімнати		
Дії	Очікуваний результат	Дійсний результат
Натиснути кнопку «Додати кімнату»	Відкриття вікна додавання кімнати	Вікно успішно відкрито
Заповнити всі поля у вікні додавання кімнати, та натиснути «Додати»	Відображення поля введеня коду адміністратора, для підтвердження дії	Вся інформація відображається коректно
Ввести код адміністратора	Кімнату додано	Кімнату успішно додано

Після завершення тестування функції додавання нової кімнати, перейдемо до перевірки коректності роботи функції видалення кімнати, щоб гарантувати її правильне функціонування. Це включатиме перевірку відображення змін на сторінках адміністратора та користувача, а також перевірку оновлення списку доступних кімнат у реальному часі. Тестування функції видалення кімнати наведено в таблиці 4.8.

Таблиця 4.8 – Тест 8

Тестування функції видалення кімнати		
Дії	Очікуваний результат	Дійсний результат
Натиснути кнопку «Видалити» навпроти кімнати, яку потрібно видалити	Видалення кімнати	Кімнату успішно видалено
Перевірити, чи кімнату видалено зі сторінки житла	Кімната більше не відображається на сторінці житла	Кімнату видалено успішно

Далі проведемо тестування, щоб перевірити коректність роботи та ефективність функції видалення оренди, забезпечуючи її належне

функціонування. Це включатиме перевірку, чи відображаються зміни на сторінці адміністратора та користувача, а також чи звільняються відповідні дати в календарі. Переконаємося, що система правильно обробляє запити на видалення та оновлює інтерфейс у реальному часі. Тестування функції видалення оренди відображено в таблиці 4.9.

Таблиця 4.9 – Тест 9

Тестування функції видалення оренди		
Дії	Очікуваний результат	Дійсний результат
Натиснути кнопку «Видалити» навпроти оренду, яку потрібно видалити	Видалення оренди	Оренду успішно видалено
Перевірити, дата видаленої оренди звільнилась в календарі на сторінці орендикімнати	Дата в календарі звільнилась	Оренду успішно видалено

Успішне тестування адміністративної частини веб-додатку забезпечує її надійну та безпечну роботу, зручність використання та відповідність функціональним вимогам, що в кінцевому рахунку підвищує якість всього програмного продукту. При тестуванні адміністративної частини помилок виявлено не було, що підтверджує її стабільність та готовність до використання в реальних умовах.

4.4 Висновки

У даному розділі було розглянули методи тестування веб-додатку, акцентуючи увагу на плануванні, цілях тестування, функціональному та нефункціональному тестуванні. Мета тестування веб-додатку полягає у виявленні та усуненні помилок, забезпеченні відповідності функціональним і нефункціональним вимогам, а також підвищенні загальної якості та надійності програмного продукту.

Було перевірено коректність відображення інформації на головній сторінці, реєстрація, авторизація користувачів, а також функціонування оренди кімнат. Тестування користувацької частини допомогло переконатися у відсутності помилок та збоїв у роботі додатку.

Також було протестовано функціонал адміністративної панелі, включаючи вхід, відображення інформації, додавання та видалення кімнат, а також управління орендою. Це забезпечило надійність, безпеку та зручність використання адміністративної частини додатку.

Загальний аналіз методів тестування веб-додатку показав важливість комплексного підходу до забезпечення якості програмного забезпечення. Використання різноманітних методів тестування дозволяє виявити та усунути потенційні проблеми ще на етапі розробки, що в кінцевому результаті підвищує якість, надійність та конкурентоспроможність веб-додатку на ринку.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено веб-додаток для бронювання житла. На початку роботи здійснено огляд індустрії оренди житла, зокрема зростання популярності онлайн-бронювань. Визначено основні цілі та вимоги до програмного забезпечення і проведено порівняльний аналіз аналогів, таких як Airbnb і Booking.com, для виявлення їхніх переваг і недоліків. Також було обґрунтовано вибір середовища розробки, та проведено аналіз обраного середовища Visual Studio Code з його аналогами, такими, як IntelliJ IDEA і Atom.

Під час проєктування системи обрано трьохрівневу архітектуру (фронтенд, бекенд та сервер баз даних), що забезпечує ефективну розробку та масштабування. Особливу увагу приділено інтуїтивному, мінімалістичному дизайну інтерфейсу, адаптивному до різних пристроїв, для зручного користування програмою.

Розробка додатку включала аналіз технологій, розробку алгоритмів реєстрації, авторизації та додавання кімнат. Використання сучасних інструментів, таких як HTML, SCSS, JavaScript та Firebase, дозволило створити функціональний і безпечний додаток, що задовольняє потреби користувачів. Розроблені алгоритми забезпечують безпеку та доступ користувачів до функціоналу.

Тестування додатку включало перевірку коректності відображення інформації, реєстрації та авторизації користувачів, функціонування оренди кімнат і адміністративної панелі. Це дозволило виявити та усунути потенційні проблеми на етапі розробки, забезпечуючи високу якість продукту.

Загалом, комплексний підхід до розробки та тестування забезпечив створення надійного, безпечного і зручного веб-додатку для бронювання житла, що відповідає потребам сучасних користувачів і має потенціал для успішної реалізації на ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Booking.com [Електронний ресурс] – Режим доступу до ресурсу: https://www.booking.com

2. Airbnb [Електронний ресурс] – Режим доступу до ресурсу: https://www.airbnb.com.ua

3. "HTML and CSS " – Джон Дакетт [Електронний ресурс] – Режим доступу до ресурсу:https://wtf.tw/ref/duckett.pdf

4. "Dynamic CSS Secrets Takeaways " - Леа Веро [Електронний ресурс] – Режим доступу до ресурсу: https://forsethingvild.medium.com/lea-verous-dynamic

5. Firebase Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs

6. Firebase як бекенд для будь-яких застосунків, та як використовувати Firebase-сервіси [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/44058/

7. "Guide to HTML, CSS, Graphics" – Дженніфер Роббінс [Електронний ресурс] – Режим доступу до ресурсу: https://www.academia.edu/Learning_Web_Design_A_Beginners_Guide

8. "Responsive Web Design with HTML5 and CSS3" - Бен Фрінк [Електронний ресурс] – Режим доступу до ресурсу: https://perpus.univpancasila.ac.id/repository/EBUPT200673.pdf

9. "HTML – A Programming Course " – Аннегрет М.Гросс [Електронний ресурс] – Режим доступу до ресурсу: https://www.academia.edu/HTML_A_Programming_Course_for_Beginners

10. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук / Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. [Електронний ресурс] – Режим доступу до ресурсу: https://pdf.lib.vntu.edu.ua/books/2015/Romanyuk_2003_216.pdf

11. "CSS Mastery" – Енді Баддітт [Електронний ресурс] – Режим доступу до ресурсу: https://www.academia.edu/CSS_Mastery

12. Angular JS Fungsi Dasar Angular JS[Електронний ресурс] Режим доступу до ресурсу: https://www.academia.edu/Angular_JS_Fungsi_Dasar_Angular_JS

13. HTML #html [Електронний ресурс] Режим доступу до ресурсу: https://www.academia.edu/39985243/HTML_html
14. Сучасний підручник з JavaScript [Електронний ресурс] Режим доступу до ресурсу: https://uk.javascript.info
15. Український веб довідник [Електронний ресурс] Режим доступу до ресурсу: https://css.in.ua
16. Хмарні технології на прикладі Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/52183/3/17059-60421-1-PB.pdf
17. Формат файлу SCSS – каскадна таблиця стилів [Електронний ресурс] Режим доступу до ресурсу: https://docs.fileformat.com
18. Основи веб-дизайну [Електронний ресурс] Режим доступу до ресурсу: https://school1k24.at.ua/WEB/OsnovyWebDis.pdf
19. Веб-дизайн і комп'ютерна графіка - О. Н. Романюк, Д. І. Катєльніков, О. П. Косовець [Електронний ресурс] – Режим доступу до ресурсу: https://ir.lib.vntu.edu.ua/bitstream/handle (дата звернення:07.04.2024)
20. JavaScript Підручник. Основи вебпрограмування [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/js/index.html

ДОДАТКИ

Додаток А. Технічне завдання

67

Міністерство освіти і науки України
Вінницький національний технічний університет
факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедрою ПЗ
д.т.н., проф.
_____О. Н. Романюк
" 12 " березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного застосунку для бронювання житла» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____к.т.н., доцент О.М. Хошаба
" 12 " березня 2024 р.
Виконав:
_____студент гр. 4ПІ-20Б Б.В. Мельник
" 12 " березня 2024 р.

Вінниця – 2024 року

1. **Найменування та галузь застосування**

Бакалаврська дипломна робота: «Розробка програмного застосунку для бронювання житла».

Галузь застосування – онлайн бронювання.

2. **Підстава для розробки.**

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора по ВНТУ про закріплення тем БДР.

3. **Мета та призначення розробки.**

Метою цієї роботи є створення веб-додатку для бронювання житла, який забезпечує зручний, інтуїтивно зрозумілий та ефективний процес пошуку, резервування та управління житлом для різних типів користувачів. Програмне забезпечення використовує сучасні технології, такі як HTML, SCSS, JavaScript та Firebase, і базується на трьохрівневій архітектурі для забезпечення безпеки, масштабованості та надійності. Проєкт враховує потреби користувачів через адаптивний дизайн інтерфейсу, автоматизацію процесів реєстрації та бронювання, а також забезпечує конкурентоспроможність продукту через порівняльний аналіз існуючих рішень. Завдяки ретельному тестуванню функціональних та нефункціональних аспектів, додаток гарантує високу якість, безпеку та зручність використання.

4. **Вихідні дані для проведення НДР**

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. "Guide to HTML, CSS, Graphics" – Дженніфер Роббінс [Електронний ресурс] – Режим доступу до ресурсу: https://www.academia.edu/Learning_Web_Design_A_Beginners_Guide
2. Firebase Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs

5. Технічні вимоги

Вихідні дані до роботи: список місць проживання; інформація про вільні кімнати у вибраних місцях; система управління базами даних – Firebase; мова запитів – Firestore Query Language; вхідні дані – база даних місць проживання та база даних з клієнтами; середовище розробки – Visual Studio Code ;мова програмування – JavaScript.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандар там України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1) пояснювальна записка до бакалаврської дипломної роботи;
2) технічне завдання;
3) лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка задачі	15.03.24 – 23.03.24	
2	Аналіз аналогів та аналіз вимог	15.03.24 – 23.03.24	
3	Проєктування системи	24.03.24 – 18.04.24	
4	Вибір середовища та мови розробки	25.04.24 – 26.04.24	
5	Розробка системних модулів	27.04.24 – 15.05.24	
6	Тестування програми	17.05.24 – 26.05.24	
7	Оформлення матеріалів до захисту БДР	27.05.24 – 05.06.24	

10. Порядок контролю та прийняття

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

71

Додаток Б. Перевірка на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ
РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного застосунку для бронювання житла

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Хошаба О.М.

Оригінальність	91,9 %
Схожість	8.1 %

Аналіз звіту подібності

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Мельник Б.В.

Керівник роботи Хошаба О.М.

Додаток В. Лістинг програми

Index.html:

```
<!DOCTYPE html>
<html lang="ru">
<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>Booking hotels</title>
  <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=Montserrat:ital,wght@0,100..900;1,100..900&display=swap"
   rel="stylesheet">
  <!-- Firebase JavaScript SDK без модульности -->
  <script src="https://www.gstatic.com/firebasejs/10.9.0/firebase-app-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/10.9.0/firebase-auth-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.6.0/firebase-firestore-compat.js"></script>
  <!-- Другие модули Firebase, если они нужны -->
  <link rel="stylesheet" href="css/style.min.css">
</head>
<body>
  <header class="header">
    <div class="container">
      <div class="header__inner">
        <a href="#" class="header__logo">
          <img src="./images/logo.png" alt="logo" class="header__logo-img">
        </a>
        <div class="header__hi-box">
          <p id="user-info" class="header__hi">
            Вітаємо:)
          </p>
          <img id="user-photo" src="" alt="">
        </div>
        <nav class="menu">
          <ul class="menu__list">
            <li id="buttonregister" class="menu__item">
              <a href="#" id="openRegister" class="menu__link">
                Увійти\Зареєструватись
              </a>
            </li>
            <li id="logout" class="menu__item none">
```

```
          <a id="sign-out" href="#" class="menu__link">
           Вийти
          </a>
         </li>
        </ul>
       </nav>
      </div>
     </div>
    </header>
    <main class="main">
     <section class="top">
      <div class="container">
       <div class="top__inner">
        <img src="./images/top_section.png" alt="" class="top__img">
        <p class="top__text">
         "Відкрийте світ відпочинку: швидко, зручно, надійно – бронюйте житло легко!"
        </p>
        <img class="top__mouse" src="./images/mous__top.svg" alt="">
       </div>
      </div>
     </section>
     <section class="hotels">
      <div class="container">
       <div class="hotels__inner">
        <p class="hotels__title">
         Мережа наших місць проживання
        </p>
        <ul id="get-hotels" class="hotels__list">
         <!-- <li class="hotels__item">
          <img src="./images/hotel-vinnitsa-page.jpg" alt="" class="hotels__item-img">
          <div class="hotels__box">
           <h2 class="hotels__name">Name name</h2>
           <p class="hotels__descr">
            Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ea, id totam eveniet, quae animi
            necessitatibus saepe autem explicabo facere porro qui. Aliquam blanditiis magni ab fugit officia,
            pariatur temporibus necessitatibus?
           </p>
           <div class="hotels__link-to-hotel-box">
            <a href="./vinnitsa-hotel.html" class="hotels__link-to-hotel">
             Подивитись кімнати
            </a>
           </div>
          </div>
```

```
        </li>
        <li class="hotels__item">
          <img src="./images/hotel2.jpg" alt="" class="hotels__item-img">
          <div class="hotels__box">
            <h2 class="hotels__name">Name2 name2</h2>
            <p class="hotels__descr">
              Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ea, id totam eveniet, quae animi
              necessit
            </p>
            <div class="hotels__link-to-hotel-box">
              <a href="#" class="hotels__link-to-hotel">
                Подивитись кімнати
              </a>
            </div>
          </div>
        </li> -->
      </ul>
    </div>
  </div>
 </section>
</main>
<footer class="footer"></footer>
<div class="register-modal">
  <a href="#" class="register-modal-close">
    <img src="./images/close.svg" alt="">
  </a>
  <div class="register-box">
    <p class="register__title">
      Реєстрація
    </p>
    <form class="register-form">
      <p class="register-form-titles">Введіть пошту</p>
      <input placeholder="mail@ds.com" type="email" class="register-input" id="email">
      <p class="register-form-titles">Введіть ім'я</p>
      <input placeholder="Name" type="text" class="register-input" id="firstName">
      <p class="register-form-titles">Введіть прізвище</p>
      <input placeholder="Last name" type="text" class="register-input" id="lastName">
      <p class="register-form-titles">Введіть пароль</p>
      <input placeholder="password" type="password" class="register-input" id="password">
      <button type="button" class="register-btn" id="register-btn" >Зареєструватись</button>
      <p class="register__descr">
        Уже є обліковий запис? <a href="#" id="toLogin" class="register-link-to-login">Увійти</a>
      </p>
```

```
        <button id="google-login">
          <p>Увійти за допомогою Google</p>
          <img src="./images/sign.png" alt="">
        </button>
      </form>
    </div>
    <div class="login-box none">
      <p class="register__title">
        Вхід
      </p>
      <form class="register-form">
        <p class="register-form-titles">Введіть пошту</p>
        <input id="loginEmail" placeholder="mail@ds.com" type="text" class="register-input">
        <p class="register-form-titles">Введіть пароль</p>
        <input id="loginPassword" placeholder="password" type="password" class="register-input">
        <button id="loginBtn" class="register-btn">
          Увійти
        </button>
        <p class="register__descr">
          Немає облікового запису? <a href="#" id="toRegister" class="register-link-to-login">Зареєструватись</a>
        </p>
        <button id="google-login">
          <p>Увійти за допомогою Google</p>
          <img src="./images/sign.png" alt="">
        </button>
      </form>
    </div>
  </div>
  </div>
  <div class="box-loader">
    <span class="loader"></span>
  </div>
  <script type="module" src="js/main.min.js"></script>
  <script type="module" src="js/modal.js"></script>
  <script type="module" src="js/get-hotels-index.js"></script>
</body>
</html>
```

Admin_panel.html:

```
<!DOCTYPE html>
<html lang="ru">
<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>Admin Panel</title>
  <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=Montserrat:ital,wght@0,100..900;1,100..900&display=swap"
   rel="stylesheet">
  <!-- Firebase JavaScript SDK без модульності -->
  <script src="https://www.gstatic.com/firebasejs/10.9.0/firebase-app-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/10.9.0/firebase-auth-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.6.0/firebase-firestore-compat.js"></script>
  <!-- Інші модулі Firebase, якщо вони потрібні -->
  <link rel="stylesheet" href="css/style.min.css">
</head>
<body>
  <header class="header">
    <div class="container">
      <div class="header__inner">
        <a href="./index.html" class="header__logo">
          <img src="./images/logo.png" alt="logo" class="header__logo-img">
        </a>
        <div class="header__hi-box">
          <p id="user-info" class="header__hi">
            Адміністративна панель
          </p>
        </div>
        <nav class="menu">
          <ul class="menu__list"></ul>
        </nav>
      </div>
    </div>
  </header>
  <main class="main">
    <section class="adminhotels">
      <div class="container">
        <div class="adminhotels__buttons">
          <button id="addhotelbtn" class="adminhotels__btn">
            Додати житло
          </button>
          <button id="addroombtn" class="adminhotels__btn">
            Додати кімнату
          </button>
```

```
</div>
<div class="adminhotels__inner">
  <p class="adminhotels__title">
    Список місць проживання
  </p>
  <ul class="adminhotels__list">
    <!-- <li class="adminhotels__item">
      <div class="hotels__item">
        <img src="./images/hotel2.jpg" alt="" class="hotels__item-img">
        <div class="hotels__box">
          <h2 class="hotels__name">Name2 name2</h2>
          <p class="hotels__descr">
            Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ea, id totam eveniet, quae animi
            necessit
          </p>
        </div>
      </div>
    </li> -->
  </ul>
</div>
<div class="adminrooms__inner">
  <p class="adminhotels__title">
    Список кімнат
  </p>
  <ul class="adminrooms__list">
    <!-- <li class="adminrooms__item">
      <div data-id="22" class="order__item">
        <div class="order__img-box">
          <img src="./images/hotel-vinnitsa-page.jpg" alt="" class="oreder__img">
        </div>
        <div class="order-content-box">
          <h4 class="order__item-title">Люкс 1 місце</h4>
          <div class="order-mini-box">
            <p class="order-square">
              40 м2
            </p>
            <p class="order-square">
              1 місце
            </p>
          </div>
          <ul class="order__functions-list">
            <li class="order__functions-item">
              <img class="order__functions-icon" src="./images/wifi.png" alt="">
```

```
            <p class="order__functions-descr">
              Wi-fi
            </p>
          </li>
          <li class="order__functions-item">
            <img class="order__functions-icon" src="./images/air.png" alt="">
            <p class="order__functions-descr">
             Кондиціонер
            </p>
          </li>
          <li class="order__functions-item">
            <img class="order__functions-icon" src="./images/seif.png" alt="">
            <p class="order__functions-descr">
             Сейф
            </p>
          </li>
          <li class="order__functions-item">
            <img class="order__functions-icon" src="./images/phone.png" alt="">
            <p class="order__functions-descr">
             Телефон
            </p>
          </li>
        </ul>
        <p class="order__description">
          Lorem ipsum, dolor sit amet consectetur adipisicing elit. Voluptatem perspiciatis possimus sunt non maxime deleniti assumenda magnam officia! Quas nihil recusandae consequatur consequuntur dolorem deleniti, quis vitae odio non aut.
        </p>
        <p class="order__description">
         Готель : назва готелю
        </p>
       </div>
     </div>
   </li> -->
  </ul>
  <p class="order__descr">
   Список орендованих кімнат
  </p>
  <ul id="orders-room-list" class="order__list">
   <li data-id="22" class="order__item">
    <div class="order__img-box">
     <img src="./images/hotel-vinnitsa-page.jpg" alt="" class="oreder__img">
    </div>
```

```
          <div class="order-content-box">
            <h4 class="order__item-title">Люкс 1 місце</h4>
            <div class="order-mini-box">
              <p>
                Користувач:
              </p>
              <p>Ім'я Фамілія</p>
             <a href="mailto:">post@g.com</a>
            </div>
            <div class="date">
              <p>
                Заброньовано на :
              </p>
              <p>
                23.04.2022
              </p>
              <p>
                <p>
                  Ціна
                </p>
                <p>
                5500
                </p>
              </p>
            </div>
            <p class="order__description">
              Lorem ipsum, dolor sit amet consectetur adipisicing elit. Voluptatem perspiciatis possimus sunt non maxime deleniti assumenda magnam officia! Quas nihil recusandae consequatur consequuntur dolorem deleniti, quis vitae odio non aut.
            </p>
          </div>
        </li>
      </ul>
    </div>
  </div>
 </section>
</main>
<footer class="footer"></footer>
<div class="security">
 <div class="container">
  <div class="security__inner">
   <div class="security__box">
    <p class="security__title">
```

```
        Для входу в панель адміністратора, введіть код доступу
       </p>
       <input class="security__input" value="" placeholder="Код доступу" type="number">
       <button class="security__login">
        Увійти
       </button>
      </div>
     </div>
    </div>
   </div>
   <div id="addhotelmodal" class="addhotelmodal">
    <div class="container">
     <div class="addhotelmodal__inner">
      <button id="closeaddhotelmodal" class="close-modal">
       <img src="./images/close.svg" alt="">
      </button>
      <p class="addhotelmodal-title">
       Додавання житла
      </p>
      <form class="addhotelmodal__form">
       <p class="addhotelmodal__descr">
        Додайте назву
       </p>
       <input required name="name" placeholder="Назва" type="text" class="addhotelmodal__input">
       <p class="addhotelmodal__descr">
        Додайте опис
       </p>
       <input required name="description" placeholder="Опис" type="text" class="addhotelmodal__input">
       <p class="addhotelmodal__descr">
        Додайте посилання на картинку
       </p>
       <input required name="imageURL" placeholder="Посилання" type="text" class="addhotelmodal__input">
       <p class="addhotelmodal__descr">
        Додайте посилання сторінку
       </p>
       <input required name="pageURL" placeholder="Посилання" type="text" class="addhotelmodal__input">
       <button class="addhotelmodal__add">
        Додати
       </button>
      </form>
     </div>
    </div>
   </div>
```

```
<div id="addroommodal" class="addhotelmodal">
  <div class="container">
    <div class="addhotelmodal__inner">
      <button id="closeaddroommodal" class="close-modal">
        <img src="./images/close.svg" alt="">
      </button>
      <p class="addhotelmodal-title">
       Додавання Кімнати
      </p>
      <form class="addhotelmodal__form">
        <p class="addhotelmodal__descr">
          Виберіть житло
        </p>
        <select  name="hotels" id="lang"></select>
        <p class="addhotelmodal__descr">
          Додайте назву
        </p>
        <input id="roomname" required name="name"  placeholder="Назва" type="text"
      class="addhotelmodal__input">
        <p class="addhotelmodal__descr">
          Додайте кількість місць
        </p>
        <input id="roomcapacity" required name="capacity" placeholder="Кількість місць" type="number"
      class="addhotelmodal__input">
        <p class="addhotelmodal__descr">
          Додайте площу
        </p>
        <input id="roomarea" required name="area" placeholder="Площу" type="number"
      class="addhotelmodal__input">
        <p class="addhotelmodal__descr">
          Додайте опис
        </p>
        <input id="roomdescription" required name="description" placeholder="Опис" type="text"
      class="addhotelmodal__input">
        <p class="addhotelmodal__descr">
          Додайте ціну
        </p>
        <input id="roomprice" required name="price" placeholder="Ціна" type="text" class="addhotelmodal__input">
        <p class="addhotelmodal__descr">
          Додайте посиалання на картинку
        </p>
        <input id="roomimageURL" required name="imageURL" placeholder="Посилання" type="text"
      class="addhotelmodal__input">
```

```
        <button id="addroomtohotel" class="addhotelmodal__add">
          Додати
        </button>
       </form>
     </div>
   </div>
  </div>
  <div class="box-loader">
    <span class="loader"></span>
  </div>
  <script src="./js/admin.js"></script>
</body>
      </html>
```

Main.js:

```
const userPhotoElement = document.querySelector('#user-photo');
userPhotoElement.classList.add('none');
restoreAuthState()
/**auth */
let registerModal = document.querySelector('.register-modal');
const firebaseConfig = {
   apiKey: "AIzaSyDAh988QKboXhU_z8-YSKsNF77gls72Xck",
   authDomain: "booking-app-d9f5e.firebaseapp.com",
   projectId: "booking-app-d9f5e",
   storageBucket: "booking-app-d9f5e.appspot.com",
   messagingSenderId: "704630127315",
   appId: "1:704630127315:web:6ae461720be49c9bc1f226",
   measurementId: "G-QLVT7KX31E"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
// Функція для оновлення інформації про користувача на сторінці
function updateUserUI(user) {
   // Отримуємо елемент для відображення інформації про користувача
   const userInfoElement = document.getElementById('user-info');
   const buttonRegister = document.querySelector('#buttonregister');
   const logoutBtn = document.querySelector('#logout');
   buttonRegister.classList.add('none');
   logoutBtn.classList.remove('none');
   // Оновлюємо вміст з ім'ям користувача
   userInfoElement.textContent = `Ласкаво просимо, ${user.displayName}!`;
   if (user.photoURL) {
```

```
        userPhotoElement.style.display = 'block';
        userPhotoElement.src = user.photoURL; // Встановлюємо URL зображення профілю
        // Показуємо зображення
    }else  {
        userPhotoElement.style.display = 'none';
        console.log('----');// Приховуємо зображення, якщо URL відсутній
    }
}
// Функція для збереження інформації про користувача в базі даних
function saveUserInfo(user) {
    // Отримуємо посилання на базу даних Firestore
    const db = firebase.firestore();
    // Створюємо посилання на документ користувача у колекції "users" (замініть на свою колекцію)
    const userRef = db.collection('users').doc(user.uid);
    // Записуємо інформацію про користувача у документ
    return userRef.set({
        name: user.displayName,
        email: user.email,
        photoURL: user.photoURL // Сохраняем URL изображения профиля
    }, { merge: true }) // Використовуємо опцію merge, щоб зберегти лише нові поля і не перезаписувати існуючі
        .then(() => {
            console.log("Інформація про користувача успішно збережена в базі даних");
        })
        .catch((error) => {
            console.error("Помилка при збереженні інформації про користувача:", error);
        });
}
// Функція для перевірки стану автентифікації користувача
function checkAuthState() {
    firebase.auth().onAuthStateChanged((user) => {
        if (user) {
            // Якщо користувач автентифікований, оновлюємо інформацію на сторінці
            updateUserUI(user);
        } else {
            // Якщо користувач не автентифікований, можна виконати необхідні дії, наприклад, показати форму
       входу
            // Або перенаправити користувача на сторінку входу
            // Наприклад:
            // window.location.href = '/login.html';
        }
    });
}
// Функція для збереження стану автентифікації в локальному сховищі
```

```
function saveAuthState(user) {
    localStorage.setItem('user', JSON.stringify(user));
}
// Функція для відновлення стану автентифікації з локального сховища
function restoreAuthState() {
    const user = localStorage.getItem('user');
    if (user) {
        updateUserUI(JSON.parse(user));
    }
}
const registerBtnWithMail = document.querySelector('#register-btn');

registerBtnWithMail.addEventListener('click', (e) => {
    e.preventDefault();
    registerUser()
})
function registerUser() {
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const firstName = document.getElementById('firstName').value;
    const lastName = document.getElementById('lastName').value;
    firebase.auth().createUserWithEmailAndPassword(email, password)
        .then((userCredential) => {
            // Успішна реєстрація
            const user = userCredential.user;
            // Оновлюємо профіль користувача з інформацією про ім'я
            user.updateProfile({
                displayName: firstName + ' ' + lastName
            }).then(() => {
                // Ім'я користувача оновлено
                console.log('Ім\'я користувача оновлено');
                // Зберігаємо додаткові дані про користувача в базі даних
                firebase.firestore().collection('users').doc(user.uid).set({
                    name: firstName + ' ' + lastName,
                    email: email,
                    photoURL: './images/user.png' // Поки що встановлюємо як null, можна змінити на значення URL
        фото користувача
                }).then(() => {
                    console.log('Дані про користувача збережено в базі даних');
                    console.log(user);
                    saveAuthState(user);
                    registerModal.classList.remove('open-modal');
                    // Save user info to database
```

```
                // Update user info on the page
                updateUserUI(user);
            }).catch((error) => {
                console.error('Помилка при збереженні даних користувача в базі даних:', error);
            });
        }).catch((error) => {
            // Помилка при оновленні імені користувача
            console.error('Помилка при оновленні імені користувача:', error);
        });
    })
    .catch((error) => {
        // Помилка при реєстрації
        const errorCode = error.code;
        const errorMessage = error.message;
        console.error('Помилка при реєстрації:', errorMessage);
    });
}
logimWidthMail()
function logimWidthMail() {
    // Отримуємо посилання на форму входу, електронну пошту та пароль
    const loginBtn = document.getElementById('loginBtn');
    const loginEmail = document.getElementById('loginEmail');
    const loginPassword = document.getElementById('loginPassword');
    // Додаємо обробник подій для форми входу
    loginBtn.addEventListener('click', (event) => {
        event.preventDefault(); // Зупиняємо стандартну поведінку форми
        const email = loginEmail.value;
        const password = loginPassword.value;
        // Виконуємо логін користувача з введеними даними
        firebase.auth().signInWithEmailAndPassword(email, password)
            .then((userCredential) => {
                // Успішний вхід
                const user = userCredential.user;
                console.log('Успішний вхід:', user);
                saveAuthState(user);
                registerModal.classList.remove('open-modal');
                // Save user info to database
                saveUserInfo(user);
                // Update user info on the page
                updateUserUI(user);
                // Оновлюємо сторінку або виконуємо інші дії
            })
            .catch((error) => {
```

```
            // Помилка під час входу
            const errorCode = error.code;
            const errorMessage = error.message;
            console.error('Помилка під час входу:', errorMessage);
        });
    });
}
// Function to sign in with Google
function signInWithGoogle() {
    const provider = new firebase.auth.GoogleAuthProvider();
    firebase.auth().signInWithPopup(provider)
        .then((result) => {
            // User successfully authenticated
            const user = result.user;
            saveAuthState(user);
            registerModal.classList.remove('open-modal');
            // Save user info to database
            saveUserInfo(user);
            // Update user info on the page
            updateUserUI(user);
        })
        .catch((error) => {
            // Authentication error
            console.error('Authentication error:', error);
        });
}
// Event listener for Google login button
const googleLogin = document.querySelector('#google-login');
googleLogin.addEventListener('click', (e) => {
    e.preventDefault();
    signInWithGoogle();
});
function clearAuthState() {
    localStorage.removeItem('user');
}
// Функція для перезавантаження сторінки
function reloadPage() {
    location.reload();
}
// Функція для виходу з аккаунту
function signOut() {
    firebase.auth().signOut()
        .then(() => {
```

```
            // Вихід успішний
            console.log('Користувач вийшов з облікового запису');
            clearAuthState();
            reloadPage();
            userPhotoElement.classList.add('none');
            // Після виходу ви можете перенаправити користувача на сторінку входу або виконати інші дії
            // Наприклад:
            // window.location.href = '/login.html';
        })
        .catch((error) => {
            // Помилка виходу
            console.error('Помилка виходу з облікового запису:', error);
        });
}
// Прослуховувач події для кнопки виходу
const signOutButton = document.querySelector('#sign-out');
signOutButton.addEventListener('click', (e) => {
    e.preventDefault();
    signOut();
});
```

Admin.js:

```
const securityCode = '213134421';
const buttonLoginNode = document.querySelector('.security__login');
const inputValue = document.querySelector('.security__input');
const modalNode = document.querySelector('.security');
buttonLoginNode.addEventListener('click', (e) => {
    e.preventDefault();
    let code = inputValue.value;
    if (code == securityCode) {
        modalNode.classList.add('none');
        alert('Код дійсний')
    } else {
        modalNode.classList.remove('none');
        alert('Код доступу введено неправильно')
    }
});
function showLoader() {
    const loader = document.querySelector('.box-loader');
    loader.classList.add('show-loader');
    document.body.style.overflow = 'hidden';
    const scrollBarWidth = window.innerWidth - document.documentElement.clientWidth;
```

```
    document.body.style.paddingRight = scrollBarWidth + 'px';
    document.body.style.overflow = 'hidden';
}
function hideLoader() {
    const loader = document.querySelector('.box-loader');
    loader.classList.remove('show-loader');
    document.body.style.paddingRight = '';
    document.body.style.overflow = '';
    document.body.style.overflow = '';
}
const addHotelBtn = document.querySelector('#addhotelbtn');
const addHotelModal = document.querySelector('#addhotelmodal');
const closeAddHotelModal = document.querySelector('#closeaddhotelmodal');
addHotelBtn.addEventListener('click', () => {
    addHotelModal.classList.add('open-modal');
})
closeAddHotelModal.addEventListener('click', () => {
    addHotelModal.classList.remove('open-modal');
})
const addRoomBtn = document.querySelector('#addroombtn');
const addRoomNode = document.querySelector('#addroommodal');
const closeAddRoomBtn = document.querySelector('#closeaddroommodal');
addRoomBtn.addEventListener('click', () => {
    addRoomNode.classList.add('open-modal');
})
closeAddRoomBtn.addEventListener('click', () => {
    addRoomNode.classList.remove('open-modal');
})
const firebaseConfig = {
    apiKey: "AIzaSyDAh988QKboXhU_z8-YSKsNF77gls72Xck",
    authDomain: "booking-app-d9f5e.firebaseapp.com",
    projectId: "booking-app-d9f5e",
    storageBucket: "booking-app-d9f5e.appspot.com",
    messagingSenderId: "704630127315",
    appId: "1:704630127315:web:6ae461720be49c9bc1f226",
    measurementId: "G-QLVT7KX31E"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
const db = firebase.firestore();
//add hotel
// Функція для обробки події відправлення форми
document.querySelector('.addhotelmodal__form').addEventListener('submit', function (event) {
```

```
    event.preventDefault(); // Відміняємо стандартну поведінку відправки форми
    // Отримуємо значення з полів вводу
    const name = document.querySelector('.addhotelmodal__input[name="name"]').value;
    const description = document.querySelector('.addhotelmodal__input[name="description"]').value;
    const imageURL = document.querySelector('.addhotelmodal__input[name="imageURL"]').value;
    const pageURL = document.querySelector('.addhotelmodal__input[name="pageURL"]').value;
    // Створюємо об'єкт з даними про готель
    const hotelData = {
        name: name,
        description: description,
        imageURL: imageURL,
        pageURL: pageURL
    };
    // Добавляємо дані про готель в колекцію "hotels"
    db.collection("hotels").add(hotelData)
        .then((docRef) => {
            console.log("Дані  готелю успішно добавлені:", docRef.id);
            // Очищуємо поля вводу після успішного добавлення
            document.querySelector('.addhotelmodal__form').reset();
            location.reload();
        })
        .catch((error) => {
            console.error("Помилка при добавлені даних про готель:", error);
        });
});
//get hotels from db
// Получаємо елемент, в який будемо вставляти готелі
const hotelsList = document.querySelector('.adminhotels__list');
getHotels()
function getHotels() {
    showLoader();
    // Запит до колекції "hotels" для отримання всіх готелів
    db.collection("hotels").get()
        .then((querySnapshot) => {
            let html = ''; // Стрічка для зберігання HTML-коду всіх готелів
            // Перебираємо всі документи в коллекції "hotels"
            querySnapshot.forEach((doc) => {
                // Получаємо дані про готелі з документу
                const hotelData = doc.data();
                const hotelId = doc.id;
                // Створюємо HTML-код для відображення готелю
                html += `
                <li class="adminhotels__item" data-id="${hotelId}">
```

```
        <div class="hotels__item">
          <img src="${hotelData.imageURL}" alt="" class="hotels__item-img">
          <div class="hotels__box">
            <h2 class="hotels__name">${hotelData.name}</h2>
            <p class="hotels__descr">${hotelData.description}</p>
            <button class="delete-hotel-btn delete-btn">
            <p>Видалити</p>
                    <img src="./images/delete.png" alt="">
                    </button>
          </div>
        </div>

    </li>
`;
});
// Встановлюємо HTML-код в елемент списку готелів
hotelsList.innerHTML = html;
// Отримуємо всі кнопки видалення
const deleteHotelButtons = document.querySelectorAll('.delete-hotel-btn');
// Додаємо обробник подій для кожної кнопки видалення
deleteHotelButtons.forEach((button) => {
    button.addEventListener('click', (event) => {
        // Отримуємо ID готелю, який потрібно видалити
        event.preventDefault();
        const hotelId = event.target.closest('.adminhotels__item').dataset.id;
        // Отримуємо назву готелю для повідомлення
        const hotelName = event.target.closest('.adminhotels__item').querySelector('.hotels__name').textContent;
        // Формуємо повідомлення для підтвердження
        const confirmationMessage = `Ви впевнені, що хочете видалити готель "${hotelName}"?`;
        // Показуємо підтвердження
        const isConfirmed = confirm(confirmationMessage);
        // Якщо користувач підтвердив, видаляємо готель
        if (isConfirmed) {
            // Викликаємо функцію для видалення готеля з бази даних
            deleteHotel(hotelId);
        }
    });
});
// Функція для видалення готеля з бази даних
function deleteHotel(hotelId) {
    // Видаляємо готель з колекції "hotels" за його ID
    db.collection("hotels").doc(hotelId).delete()
        .then(() => {
```

```
                console.log("Готель успішно видалений");
                location.reload();
                // Якщо потрібно, оновлюємо інтерфейс
                // Наприклад, можна видалити елемент зі списку готелів
              })
              .catch((error) => {
                console.error("Помилка при видаленні готеля:", error);
              });
          }
          hideLoader();
        })
        .catch((error) => {
          console.error("Ошибка при получении гостиниц из базы данных:", error);
        });
}
///get hotels and inner to select
// Получаємо елемент <select>, в який будемо добавляти готелі
const selectElement = document.getElementById('lang');
// Запит до колекції "hotels" для отримання всіх готелів
db.collection("hotels").get()
    .then((querySnapshot) => {
        // Перебираємо всі документи в коллекції "hotels"
        querySnapshot.forEach((doc) => {
            // Получаємо дані про готелі з документу
            const hotelData = doc.data();
            const hotelId = doc.id;
            // Створюємо елемент <option> для готелю
            const optionElement = document.createElement('option');
            optionElement.value = hotelId; // Встановлюємо значення як ідентифікатор готелю
            optionElement.textContent = hotelData.name; // Встановлюємо текстовий вміст як назву готелю
            // Добавляємо елемент <option> в <select>
            selectElement.appendChild(optionElement);
        });
    })
    .catch((error) => {
        console.error("Помилка при отриманні готелю з бази даних:", error);
    });
////add room to bd
function addRoomToSelectedHotel() {
    // Отримуємо вибраний елемент <select>
    const selectElement = document.getElementById('lang');
    // Получаємо вибраний ідентифікатор готелю з атрибута value вибраного елементу <select>
    const hotelId = selectElement.value;
```

```
    // Получаємо значення полів форми
    const roomName = document.querySelector('#roomname').value;
    const roomDescription = document.querySelector('#roomdescription').value;
    const roomCapacity = document.querySelector('#roomcapacity').value;
    const roomArea = document.querySelector('#roomarea').value;
    const roomImageUrl = document.querySelector('#roomimageURL').value;
    const roomprice = document.querySelector('#roomprice').value;
    // Перевіряємо, чи вибрано готель
    if (hotelId) {
        // Получаємо посилання на підколекцію "rooms" для вибраного готелю
        const roomsRef = db.collection("hotels").doc(hotelId).collection("rooms");
        // Добавляємо нову кімнату в підколекцію "rooms" для вибраного готелю
        roomsRef.add({
            name: roomName,
            description: roomDescription,
            capacity: roomCapacity,
            area: roomArea,
            imageURL: roomImageUrl,
            price: roomprice
        })
            .then((docRef) => {
                console.log("Кімната успішно добавлена в готель з ID:", hotelId);
                location.reload();
            })
            .catch((error) => {
                console.error("Помилка при додаванні кімнати:", error);
            });
    } else {
        console.error("Готель не вибраний.");
    }
}
// Получаємо форму
const addroomtohotel = document.querySelector('#addroomtohotel');
// Добавляємо обробник події на відправку форми
addroomtohotel.addEventListener('click', (e) => {
    e.preventDefault(); // Запобігаємо стандартній поведінці форми
    // Викликаємо функцію для додавання кімнати в вибраний готель
    addRoomToSelectedHotel();
});
/////get rooms
// Отримкємо елемент <ul>, в який будемо добавляти кімнати
const roomsList = document.querySelector('.adminrooms__list');
// Запит до коллекції "hotels" для отримання всіх готелів
```

```
db.collection("hotels").get()
    .then((querySnapshot) => {
        // Перебираємо всі документи в колекції "hotels"
        querySnapshot.forEach((hotelDoc) => {
            // Получаємо ідентифікатор готелю
            const hotelId = hotelDoc.id;
            // Получаємо дані про готель
            const hotelData = hotelDoc.data();
            // Запит до підколекції "rooms" для отримання всіх кімнат поточного готелю
            hotelDoc.ref.collection("rooms").get()
                .then((roomsQuerySnapshot) => {
                    // Перебираємо всі документи в колекції "rooms"
                    roomsQuerySnapshot.forEach((roomDoc) => {
                        // Получаємо дані про кімнату з документу
                        const roomData = roomDoc.data();
                        // Створюємо HTML-розмітку для кімнати
                        const roomHTML = `
                            <li data-id="${roomDoc.id}" data-hotel="${hotelId}" class="adminrooms__item">
                                <div data-id="${roomDoc.id}" class="order__item">
                                    <div class="order__img-box">
                                        <img src="${roomData.imageURL}" alt="" class="oreder__img">
                                    </div>
                                    <div class="order-content-box">
                                        <h4 class="order__item-title">${roomData.name}</h4>
                                        <div class="order-mini-box">
                                            <p class="order-square">
                                                ${roomData.area} м2
                                            </p>
                                            <p class="order-square">
                                                ${roomData.capacity} мест
                                            </p>
                                        </div>
                                        <ul class="order__functions-list">
                                            <li class="order__functions-item">
                                                <img class="order__functions-icon" src="./images/wifi.png" alt="">
                                                <p class="order__functions-descr">Wi-fi</p>
                                            </li>
                                            <li class="order__functions-item">
                                                <img class="order__functions-icon" src="./images/air.png" alt="">
                                                <p class="order__functions-descr">Кондиціонер</p>
                                            </li>
                                            <li class="order__functions-item">
                                                <img class="order__functions-icon" src="./images/seif.png" alt="">
```

```
                        <p class="order__functions-descr">Сейф</p>
                    </li>
                    <li class="order__functions-item">
                        <img class="order__functions-icon" src="./images/phone.png" alt="">
                        <p class="order__functions-descr">Телефон</p>
                    </li>
                </ul>
                <p class="order__description">${roomData.description}</p>
                <p class="order__description">Готель: ${hotelData.name}</p>
                <p class="order__description">Ціна: ${roomData.price} Грн</p>
                <button class="delete-room-btn delete-btn" data-roomid="${roomDoc.id}" data-
hotelid="${hotelId}">
                <p>Видалити</p>
                <img src="./images/delete.png" alt="">
                </button>
              </div>
            </div>
          </li>
        `;
        // Добавляємо HTML-розмітку кімнати в список кімнат
        roomsList.insertAdjacentHTML('beforeend', roomHTML);
        // Добавляємо обробник подій для кнопок видалення кімнат
        const deleteRoomBtn = document.querySelector(`.delete-room-btn[data-
roomid="${roomDoc.id}"][data-hotelid="${hotelId}"]`);
        deleteRoomBtn.addEventListener('click', () => {
          // Отримуємо назву кімнати і готелю для повідомлення
          const roomName = roomData.name;
          const hotelName = hotelData.name;
          // Формуємо повідомлення для підтвердження
          const confirmationMessage = `Ви впевнені, що хочете видалити кімнату "${roomName}" з
готелю "${hotelName}"?`;
          // Показуємо підтвердження
          const isConfirmed = confirm(confirmationMessage);
          // Якщо користувач підтвердив, видаляємо кімнату
          if (isConfirmed) {
            const roomId = deleteRoomBtn.getAttribute('data-roomid');
            const hotelId = deleteRoomBtn.getAttribute('data-hotelid');
            // Виконуємо запит на видалення кімнати
            db.collection("hotels").doc(hotelId).collection("rooms").doc(roomId).delete()
              .then(() => {
                // Видалення успішне, оновлюємо інтерфейс або виконуємо інші дії, які вам потрібно
                console.log("Кімната успішно видалена");
                location.reload();
```

```
                        // Оновлення інтерфейсу або інші дії
                    })
                    .catch((error) => {
                        console.error("Помилка при видаленні кімнати:", error);
                    });
                }
            });
        });
    })
    .catch((error) => {
        console.error("Помилка при отриманні комнат з готелю:", error);
    });
  });
})
.catch((error) => {
  console.error("Помилка при отриманні готелів:", error);
});
/**get orders rooms */
async function displayBookedRooms() {
  const ordersRoomList = document.getElementById('orders-room-list');
  ordersRoomList.innerHTML = ''; // Очистити список перед оновленням
  try {
    const orendRoomsQuerySnapshot = await firebase.firestore().collection("orendRooms").get();
    orendRoomsQuerySnapshot.forEach((doc) => {
      const roomData = doc.data();
        // Отримуємо інформацію про готель
      const user = JSON.parse(localStorage.getItem('user')); // Отримати дані користувача з локального сховища
      const listItem = `
          <li data-id="${doc.id}" class="order__item">
            <div class="order__img-box">
              <img src="${roomData.roomImageURL}" alt="" class="oreder__img">
            </div>
            <div class="order-content-box">
              <h4 class="order__item-title">${roomData.roomName}</h4>
              <div class="order-mini-box">
                <p>
                  Користувач:
                </p>
                <p>${roomData.user.displayName}</p>
                <a href="mailto:${roomData.user.email}">${roomData.user.email}</a>
              </div>
              <div class="date">
                <p>
```

```
                    Заброньовано на :
                  </p>
                  <p>${roomData.selectedDate}</p>
                  <p>
                    <p>
                      Ціна
                    </p>
                    <p>
                      ${roomData.roomPrice}
                    </p>
                  </p>
                </div>
                <p class="order__description">
                  ${roomData.roomDescription}
                </p>
                <button class="delete-button delete-style" data-id="${doc.id}"> <p>Видалити оренду </p><img
    src="./images/delete.png" alt=""></button>
              </div>
            </li>
          `;
        ordersRoomList.insertAdjacentHTML('beforeend', listItem);
      });
    } catch (error) {
      console.error("Помилка при отриманні орендованих кімнат:", error);
    }
}
document.addEventListener('click', async function (event) {
    if (event.target.classList.contains('delete-button')) {
      const orderId = event.target.dataset.id;
      const confirmed = confirm("Ви впевнені, що хочете видалити цю оренду?");
      if (confirmed) {
        try {
          // Отримуємо дані про орендовану кімнату
          const orderDoc = await firebase.firestore().collection("orendRooms").doc(orderId).get();
          const orderData = orderDoc.data();
          // Перевіряємо чи існує hotelId в orderData перед використанням
          if (orderData && orderData.hotelId) {
            const hotelId = orderData.hotelId;
            const roomId = orderData.roomId;
            const selectedDate = orderData.selectedDate;
            // Видаляємо орендовану дату з кімнати в готелі
            const roomDoc = await
      firebase.firestore().collection("hotels").doc(hotelId).collection("rooms").doc(roomId).get();
```

```
                const roomData = roomDoc.data();
                const updatedOrendDates = roomData.orendDates.filter(date => date !== selectedDate);
                // Оновлюємо кімнату з видаленою орендованою датою
                await firebase.firestore().collection("hotels").doc(hotelId).collection("rooms").doc(roomId).update({
      orendDates: updatedOrendDates });
              } else {
                console.error("Помилка: Поле hotelId відсутнє у даних про орендовану кімнату.");
              }
              // Видаляємо оренду
              await firebase.firestore().collection("orendRooms").doc(orderId).delete();

              // Оновлюємо список після видалення
              displayBookedRooms();
            } catch (error) {
              console.error("Помилка при видаленні оренди:", error);
            }
          }
        }
});
// Викликати функцію для відображення списку орендованих кімнат
displayBookedRooms();
```

Додаток Г. Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ БРОНЮВАННЯ ЖИТЛА

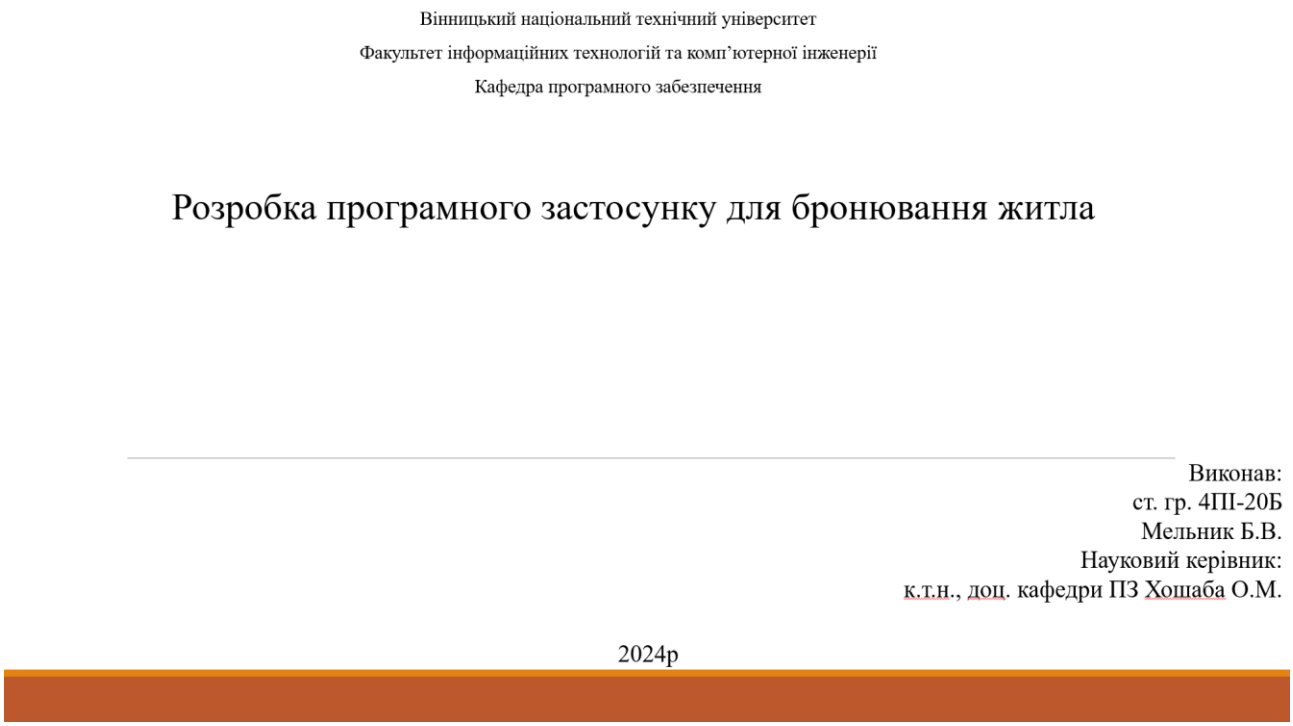


Рисунок Г.1 – Титульний слайд

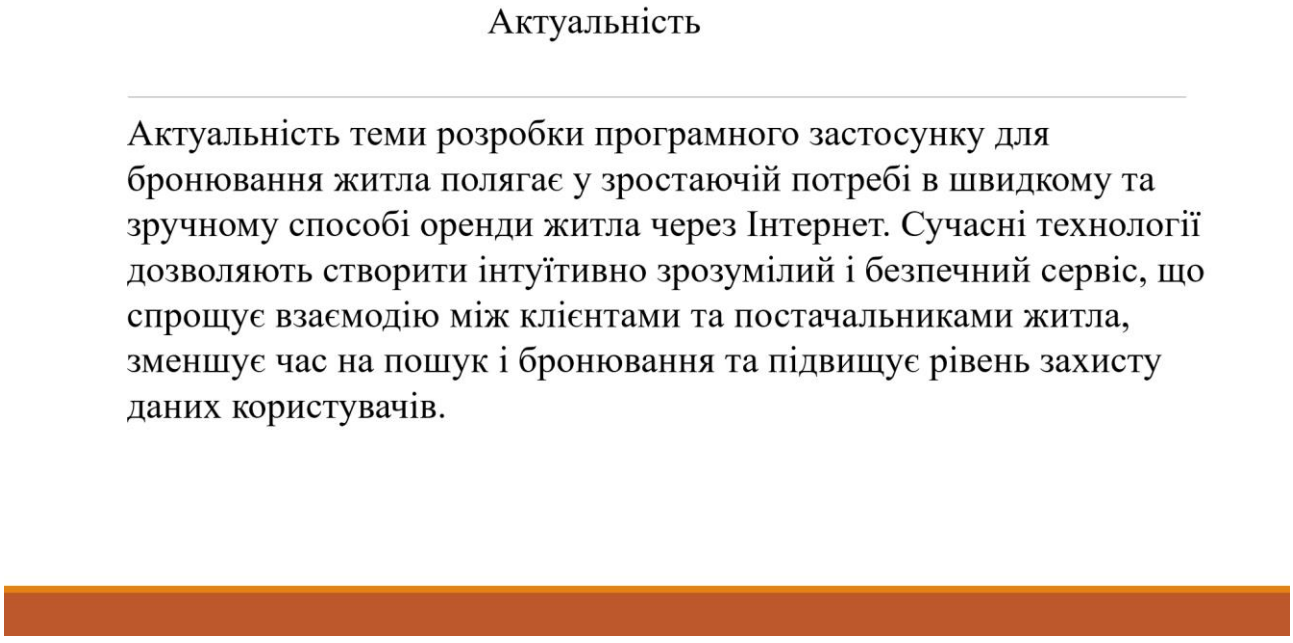


Рисунок Г.2 – Актуальність

❖ **Мета роботи:** надання користувачам зручного та ефективного інструменту для бронювання житла через Інтернет, сприяючи підвищенню якості обслуговування та зручності для користувачів.

❖ **Об'єкт дослідження:** процес бронювання житла через Інтернет та програмний застосунок, розроблений для забезпечення цього процесу.

❖ **Предмет дослідження:** програмний застосунок для бронювання житла, його функціональність, ефективність та зручність для користувачів.

Рисунок Г.3 – Мета, об'єкт і предмет дослідження

Новизна отриманих результатів:

1) Запропоновано новий підхід до розробки програмного забезпечення для бронювання житла, особливість якого полягає у зручному та інтуїтивно зрозумілому інтерфейсі для користувачів та адміністраторів, що дозволяє підвищити ефективність взаємодії з системою за рахунок оптимізації процесів навігації та управління бронюванням.

2) Подальшого розвитку отримав механізм управління житлом, у якому, на відміну від існуючих рішень, виконано інтегровані функції додавання та видалення орендних пропозицій, що дозволило зменшити кількість необхідних дій для оновлення інформації про наявні місця проживання і, як наслідок, підвищити зручність адміністрування та знизити ризик помилок в управлінні пропозиціями.

Рисунок Г.4 – Новизна отриманих результатів

Практична цінність

Практична цінність розробленого застосунку полягає у спрощенні процесу бронювання житла для користувачів та покращенні обслуговування клієнтів для бізнесу. Застосунок забезпечує зручний інтерфейс, швидкий доступ до пропозицій, та ефективну взаємодію між постачальниками житла і клієнтами. Це підвищує конкурентоспроможність на ринку нерухомості та подорожей, задовольняючи потреби користувачів.

Рисунок Г.5 – Практична цінність

Завдання дослідження:

1) Провести аналіз існуючих систем бронювання житла та визначити їх переваги і недоліки.
2) Розробка вимог до програмного застосунку, враховуючи потреби користувачів та власників житла.
3) Розробка архітектури програмного застосунку та його інтерфейсу.
4) Розробити програмний застосунок для бронювання житла, який включатиме у себе зручний та інтуїтивно зрозумілий інтерфейс для користувачів та адміністраторів.
5) Реалізувати алгоритми зручного та безпечного бронювання, та додавання житла.
6) Провести тестування програмного застосунку.

Рисунок Г.6 – Завдання дослідження

Порівняння аналогів

Критерій	Airbnb	Booking.com	Власна розробка
Дизайн	+/-	+/-	+
Інтуїтивність	-	-	+
Клієнтська підтримка	+/-	+	+
Швидкість роботи	+/-	+/-	+
Результат	1,5	2	4

Рисунок Г.7 – Порівняння аналогів

Вимоги до програмного застосунку

Категорія	Вимоги
Функціональні	Швидке та зручне бронювання житла
	Відміна бронювання
	Перегляд інформації про житло
	Додавання житла
Нефункціональні	Інтуїтивний інтерфейс
	Безперебійна робота
	Захист персональних даних
	Оптимальна швидкість роботи

Рисунок Г.8 – Вимоги до програмного застосунку

Блок-схема алгоритму роботи програми

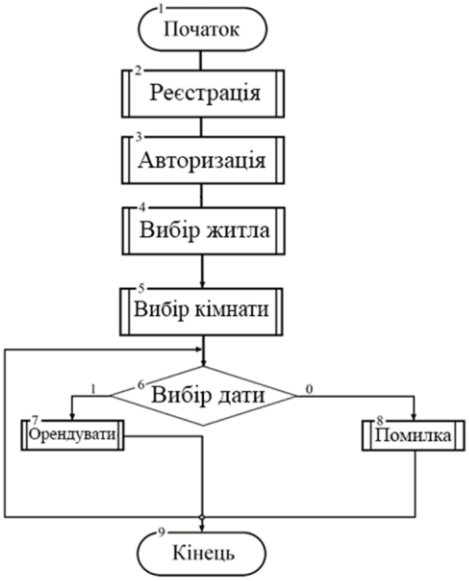


Рисунок Г.9 – Блок-схема алгоритму роботи програми

Обрані засоби розробки програмного застосунку

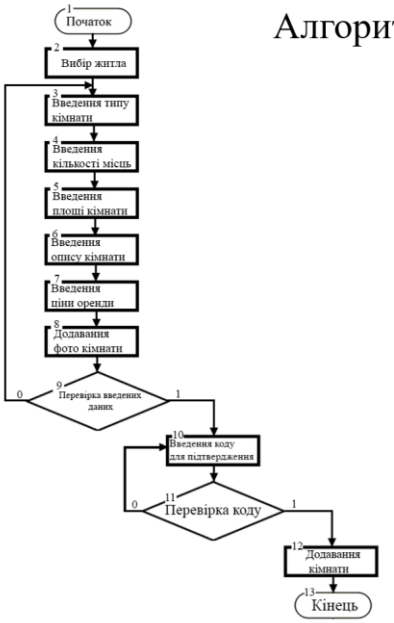
❖**Створення структури:** HTML

❖**Препроцесор:** SCSS

❖**Динамічне оновлення вмісту:** JavaScript

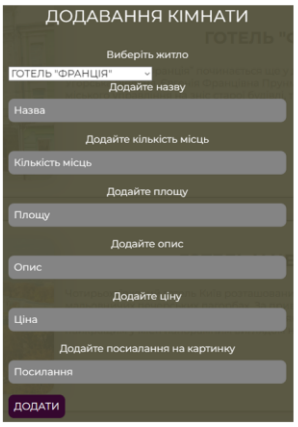
❖**База даних:** Firebase

Рисунок Г.10 – Обрані засоби розробки програмного застосунку



Блок-схема

Алгоритм додавання кімнати



Вигляд вікна додавання

Поле вводу коду доступу

Рисунок Г.11 – Алгоритм додавання кімнати

Тестування програмного застосунку

№ тесту	Назва тесту	Опис	Результат
1	Головна сторінка	Перевірка відображення контенту на головній сторінці веб-додатку	Контент відображається коректно
2	Реєстрація нового користувача	Перевірка роботи форми реєстрації	Обліковий запис користувача створено успішно
3	Авторизація користувача	Перевірка входу до системи з існуючими даними	Авторизація пройшла успішно
4	Робота оренди	Тестування оренди кімнати	Оренда пройшла успішно
5	Вхід в панель адміністратора	Перевірка входу до панелі адміністратора	Вхід виконано успішно
6	Сторінка адміністратора	Перевірка відображення контенту на сторінці адміністратора	Весь контент відображається коректно
7	Додавання кімнати	Перевірка додавання нової кімнати	Кімната успішно додана
8	Видалення кімнати	Перевірка видалення існуючої кімнати	Кімната успішно видалена
9	Видалення оренди	Перевірка видалення оренди	Оренда успішно видалена

Рисунок Г.12 – Тестування програмного застосунку

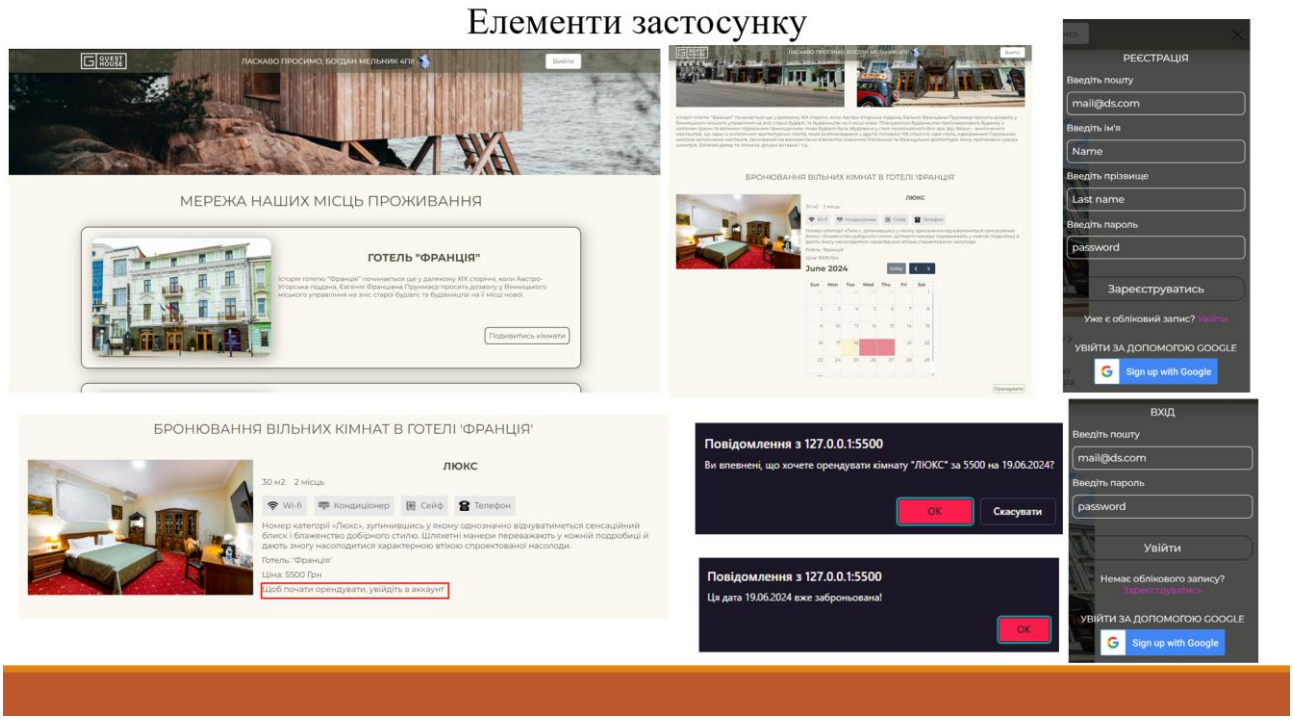


Рисунок Г.13 – Елементи застосунку

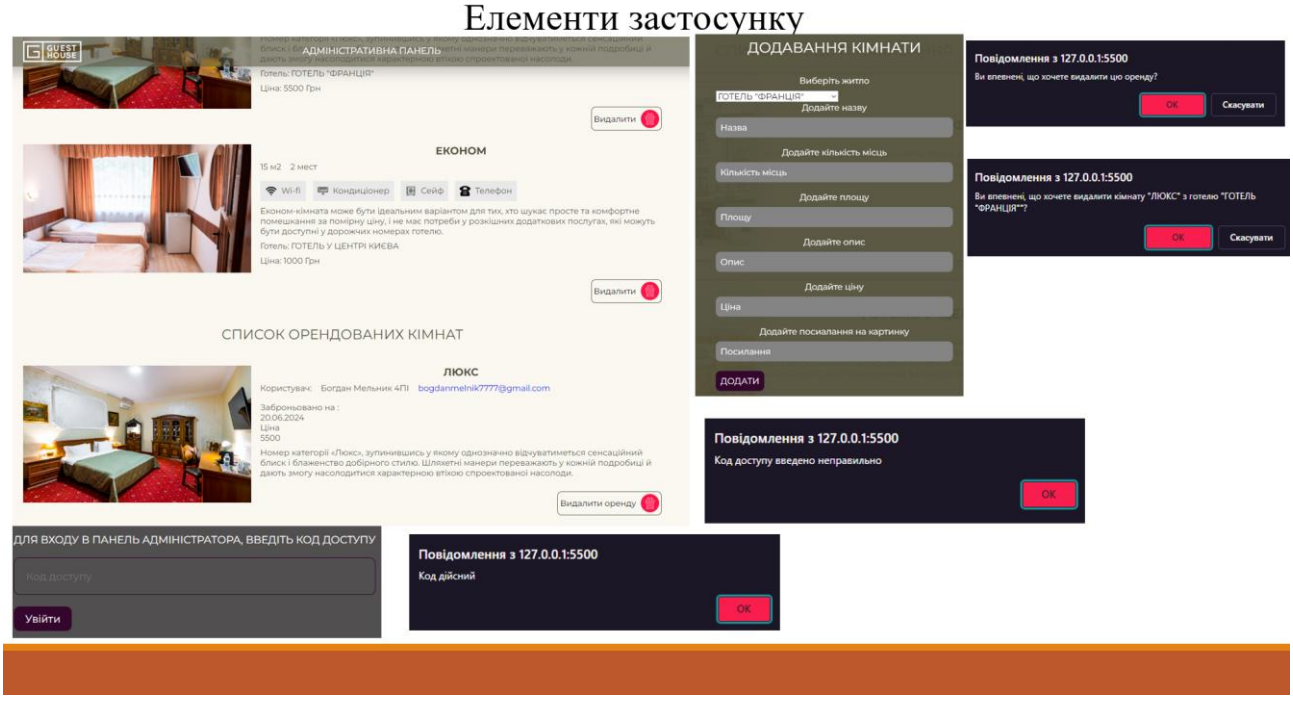


Рисунок Г.14 – Елементи застосунку

Висновок

У результаті роботи над проектом було виконано наступні завдання:

- ❖ Проведено аналіз існуючих систем бронювання житла, виявлено їх переваги та недоліки.
- ❖ Розроблено функціональні та нефункціональні вимоги до програмного забезпечення.
- ❖ Розроблено програмний застосунок з інтуїтивно зрозумілим інтерфейсом для користувачів та адміністраторів.
- ❖ Реалізовано алгоритми зручного та безпечного бронювання і додавання житла.
- ❖ Проведено тестування програмного застосунку

Рисунок Г.15 – Висновок

Дякую за увагу

Рисунок Г.16 – Фінальний слайд