

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

" _____ " _____ 2025 р.

Зав. кафедри, к.т.н., доцент

_____ Лішук Р.І.

**КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА**

Інформаційна система для бронювання житла

Виконав – студент 4 курсу, групи 41-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки

Роїк Юлія Ігорівна

Науковий керівник – д.т.н., професор

Кучерук Володимир Юрійович

Умань – 2025

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	7
ВСТУП	8
РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ, ПОСТАНОВКА ЗАДАЧІ, ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ ТА АНАЛІЗ ВИМОГ	11
1.1 Огляд предметної області та постановка задачі	11
1.2 Порівняльний аналіз існуючих систем бронювання житла	13
1.3 Визначення типів користувачів та ключових вимог	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ	25
2.1 Створення вимог до програмного забезпечення	25
2.2 Розробка архітектури системи	30
2.3 Проєктування інтерфейсу користувача	34
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ	43
3.1 Аналіз технологій та методів для розробки програмного забезпечення	43
3.2 Розробка алгоритмів реєстрації, авторизації та додавання кімнат	46
3.3 Тестування програмного додатку	52
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	62
Додаток А. Лістинг програми.....	64

Перелік умовних позначень, скорочень і термінів

VS Code	Visual Studio Code
OC	Операційна система
SSL	Secure Sockets Layer
TLS	Transport Layer Security
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
SCSS	Sass Cascading Style Sheets
AWS	Amazon Web Services
HTML	HyperText Markup Language

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі, де мобільність і швидкість прийняття рішень набувають особливої ваги, проблема онлайн-бронювання житла стає все актуальнішою. Зростання популярності цифрових сервісів свідчить про нагальну потребу у зручних, ефективних і безпечних інструментах для резервування житла через Інтернет. Попит на подібні застосунки зумовлений збільшенням кількості користувачів, які віддають перевагу цифровим платформам при плануванні подорожей або короткострокового проживання.

Сучасні технології розробки програмного забезпечення відкривають широкі можливості для створення інноваційних рішень у сфері онлайн-бронювання. Удосконалення цього процесу не лише полегшує життя споживачів, а й забезпечує конкурентні переваги для бізнесу, зокрема власників житла, які зацікавлені в ефективному просуванні своїх об'єктів.

Обґрунтування теми дослідження базується на зростаючих потребах ринку, де критично важливими факторами є швидкість, зручність та надійність сервісів. Актуальність обраної теми підтверджується стабільним зростанням туристичного сектору та високим попитом на сервіси онлайн-резервування. Дослідження цієї проблематики сприяє вирішенню практичних завдань у сфері цифрової трансформації послуг та розширює можливості як для користувачів, так і для постачальників житла.

Мета та завдання дослідження. Метою бакалаврської кваліфікаційної роботи є створення веб-застосунку для бронювання житла, який забезпечить споживачам зручний, безпечний і ефективний інструмент взаємодії з сервісом, сприятиме підвищенню якості обслуговування та задоволенню ринкового попиту.

Основні завдання дослідження.

- Провести аналіз існуючих систем онлайн-бронювання житла, визначити їхні переваги та недоліки.
- Сформулювати функціональні вимоги до програмного продукту з

урахуванням потреб споживачів та власників житла.

- Спроекувати архітектуру застосунку та інтерфейс користувача.
- Реалізувати веб-застосунок із зручним, інтуїтивно зрозумілим інтерфейсом для споживачів та адміністраторів.
- Розробити механізми реєстрації та авторизації користувачів.
- Реалізувати функціонал безпечного бронювання житла та можливість додавання нових об'єктів.
- Створити адміністративну панель для управління оголошеннями та контентом.
- Провести повноцінне тестування застосунку, оцінити його функціональність та стабільність у роботі.

Об'єкт і предмет дослідження – процес онлайн-бронювання житла та розроблений веб-застосунок для його реалізації.

Предмет дослідження – функціональність, ефективність і зручність використання веб-застосунку для резервування житла.

Методи дослідження

У процесі дослідження застосовано такі методи:

- Методи оптимізації – для вдосконалення алгоритмів і покращення продуктивності системи.
- Методи імітації та моделювання – для аналізу різних сценаріїв функціонування застосунку в умовах змінного навантаження.
- Емпіричні методи – для збору та аналізу фактичних даних, отриманих у ході тестування застосунку.
- Методи розподілених обчислень – для забезпечення масштабованості та ефективної роботи під час обробки великої кількості одночасних запитів.

Практична цінність отриманих результатів. Розроблений веб-застосунок спрощує процес бронювання житла для кінцевих користувачів, забезпечуючи швидкий доступ до актуальних пропозицій та зручну взаємодію з власниками житла. Для бізнесу застосунок є інструментом ефективного управління об'єктами оренди, що дозволяє підвищити якість обслуговування та

конкурентоспроможність на ринку туристичних і житлових послуг. Таким чином, результати дослідження мають вагому прикладну значущість як для кінцевих споживачів, так і для постачальників послуг.

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

1.1 Огляд предметної області та постановка задачі

Індустрія оренди житла є однією з найдинамічніших сфер сучасної економіки, що постійно трансформується під впливом технологічних інновацій, змін у поведінці споживачів і глобальних соціокультурних тенденцій. Особливо помітною стала роль цифрових технологій, які суттєво змінили підходи до пошуку, бронювання та управління орендованим житлом. У цьому контексті важливим є дослідження та проєктування ефективних систем резервування, здатних задовольняти потреби як орендарів, так і власників житла.

Зростання популярності онлайн-бронювання та цифрових платформ, таких як Airbnb, Booking.com, HomeAway, спричинило зміну парадигми взаємодії між споживачем і постачальником послуг. Вони пропонують широкий спектр функціональних можливостей: від алгоритмів персоналізації до інструментів управління попитом, аналітики та штучного інтелекту для покращення клієнтського досвіду. Це підвищує очікування користувачів та зумовлює потребу в постійному вдосконаленні сервісів.

В умовах високої конкуренції компанії зобов'язані впроваджувати інноваційні підходи, що включають не лише якісну технічну реалізацію, але й врахування факторів зручності, безпеки та доступності. Розробка нового веб-застосунку для бронювання житла повинна відповідати цим вимогам та враховувати практичні потреби обох сторін - споживачів і орендодавців.

Мета розробки програмного забезпечення полягає у створенні інтуїтивно зрозумілого, безпечного та функціонального веб-застосунку для резервування житла, який буде відповідати сучасним технологічним стандартам і потребам користувачів.

Основні завдання розробки включають:

- Аналіз існуючих систем бронювання: Дослідити провідні цифрові платформи (Airbnb, Booking.com, HomeAway) з метою виявлення їхніх переваг,

недоліків, функціональних можливостей, рівня захисту даних та зручності для користувачів. Провести порівняльний аналіз, щоб визначити сильні сторони, які можна інтегрувати, та слабкі місця, яких слід уникати при розробці власного застосунку.

- Формування вимог до системи: Зібрати та систематизувати вимоги різних категорій користувачів: орендарів (зручність пошуку, безпечні платежі, швидке бронювання) та орендодавців (простота додавання житла, керування замовленнями).

- Сформулювати чіткі функціональні та нефункціональні вимоги (надійність, масштабованість, UX-дизайн, безпека).

- Проєктування архітектури системи та користувацького інтерфейсу:

- Розробити багаторівневу архітектуру системи з розділенням на серверну логіку, клієнтський інтерфейс, базу даних і API.

Дизайн інтерфейсу повинен враховувати специфіку роботи з різними типами користувачів (орендарі, орендодавці, адміністратори).

Розробка веб-застосунку:

- Реалізувати базовий функціонал: пошук житла, фільтрація пропозицій, перегляд опису, реєстрація/авторизація, здійснення та скасування бронювань.

- Створити адміністративний інтерфейс для управління користувачами, об'єктами житла та моніторингу активності.

Розробка алгоритмів безпечного бронювання та управління житлом:

- Впровадити механізми підтвердження замовлень у реальному часі, обробки оплат та захисту особистих даних.

- Забезпечити можливість для власників житла легко керувати інформацією про об'єкти та бронювання.

Тестування системи:

- Провести повноцінне функціональне тестування користувацької частини: перевірка реєстрації, резервування, відображення об'єктів.

- Провести тестування адміністративної частини: управління контентом, контроль доступу, валідація функцій.

- Оцінити безпеку системи в умовах реального навантаження.

Таким чином, створення ефективного веб-застосунку для резервування житла є актуальним завданням, яке потребує комплексного підходу, включаючи ретельний аналіз предметної області, постановку технічного завдання, проєктування архітектури системи та реалізацію функціоналу відповідно до очікувань користувачів. Враховуючи досвід провідних платформ і динаміку розвитку ринку, запропонований продукт має потенціал для забезпечення якісного обслуговування, задоволення потреб споживачів і підвищення конкурентоспроможності у сфері цифрової оренди житла.

1.2 Порівняльний аналіз існуючих систем бронювання житла

Тенденції розвитку програмного забезпечення для онлайн-бронювання житла

Сфера розробки програмних рішень для онлайн-бронювання житла зазнає суттєвих трансформацій під впливом двох ключових чинників - зростання попиту на подорожі та стрімкого розвитку цифрових технологій. У таких умовах значення програмного забезпечення зростає, оскільки воно забезпечує можливість дистанційного бронювання, ефективне управління пропозиціями житла, а також автоматизацію процесів оренди для підприємств, що працюють у цій галузі.

Сучасний IT-ринок пропонує широкий спектр програмних рішень, орієнтованих на різні потреби користувачів - від глобальних мультиплатформ до локалізованих сервісів, призначених для обслуговування невеликих регіонів. Функціонал таких систем варіюється від базових інструментів для резервування до складних аналітичних і управлінських модулів, що дозволяють адаптувати платформу під конкретні бізнес-завдання.

Ключові характеристики конкурентоспроможного програмного забезпечення для бронювання житла:

- Функціональна повнота - здатність охоплювати як базові, так і розширені потреби учасників ринку оренди.

- Юзабіліті (зручність користування) - інтуїтивно зрозумілий інтерфейс як для кінцевих користувачів, так і для адміністраторів.

- Інтеграційна сумісність - можливість взаємодії із зовнішніми системами, зокрема бухгалтерськими, CRM або платіжними сервісами.

- Захищеність даних - відповідність стандартам кібербезпеки, захист персональної інформації користувачів і власників житла.

На ринку домінують великі гравці, зокрема Airbnb і Booking.com, які завдяки масштабності, впізнаваності та гнучкості змогли охопити різноманітні сегменти споживачів. У той же час існує значна кількість нішевих платформ, орієнтованих на специфічні потреби окремих груп - наприклад, туристів, які шукають еко-житло, або власників, які здають квартири в межах одного міста.

Порівняльний аналіз провідних платформ

Оцінювання особливостей функціонування провідних сервісів бронювання дозволяє виділити переваги й обмеження кожної з них, що є корисним при розробці власних рішень або виборі готових продуктів.

Airbnb - одна з найпопулярніших міжнародних платформ, заснована у 2008 році в Сан-Франциско, функціонує як посередник між орендодавцями та подорожуючими. Платформа дозволяє легко організувати як короткострокове, так і тривале перебування, забезпечуючи при цьому високий рівень персоналізації досвіду.

Основні переваги Airbnb:

- Широкий вибір житла - від класичних квартир до унікальних об'єктів (будинки на деревах, замки, човни тощо).

- Аутентичний досвід - проживання в умовах, наближених до реального життя місцевих мешканців.

- Прозора система перевірки - верифікація користувачів, система рейтингів і відкритих відгуків.

- Гнучкі умови оренди - можливість бронювання на будь-який термін із варіативними політиками скасування.

- Міжнародне покриття - платформа доступна у більшості країн світу та

має розгалужену систему локальної підтримки.

Airbnb активно охоплює різні цільові аудиторії - від туристів-одинаків до сімей і бізнес-мандрівників - і пропонує індивідуалізований підхід до обслуговування. Завдяки цьому, у поєднанні з якісним дизайном і зручною навігацією, сервіс здобув провідні позиції у своїй галузі.

Таким чином, розвиток програмного забезпечення для бронювання житла тісно пов'язаний із глобальними змінами в туристичній галузі та ІТ-секторі. Попит на гнучкі, безпечні та зручні цифрові сервіси стимулює появу нових рішень і вдосконалення існуючих. Аналіз провідних платформ, таких як Airbnb, дозволяє зрозуміти ключові фактори успіху на ринку та сформулювати стратегічні орієнтири для власних програмних розробок у цій сфері.



Рисунок 1.1. Інтерфейс головної сторінки «Airbnb»

Аналіз недоліків платформи Airbnb та характеристика Booking.com

Попри численні переваги, платформа Airbnb має низку недоліків, які варто враховувати при плануванні поїздок і виборі житла. Основні з них:

Ризик виникнення непередбачуваних ситуацій. Мандрівники можуть стикнутися з проблемами, зокрема раптовим скасуванням бронювання орендодавцем або невідповідністю реальних умов проживання задекларованим.

Варіативність якості об'єктів. Відсутність єдиних стандартів призводить до нерівномірного рівня комфорту житла, що іноді негативно позначається на

досвіді користувачів.

Додаткові сервісні збори. Платформа стягує комісію як із гостей, так і з орендодавців, що може значно підвищити кінцеву вартість проживання порівняно з іншими варіантами.

Відсутність гарантованої якості послуг. На відміну від готелів, Airbnb не забезпечує уніфікованого рівня сервісу, що створює ризики для користувачів у разі незадовільного обслуговування.

Юридичні обмеження. У деяких країнах та містах існують законодавчі норми, які регулюють або обмежують короткострокову оренду, що може створювати правові труднощі для обох сторін - орендодавців і гостей.

Попри вказані обмеження, Airbnb залишається однією з найпопулярніших платформ у світі завдяки великому вибору об'єктів, можливості отримати унікальний досвід проживання, гнучкості умов оренди та підтримці локального контексту. Щоб зменшити ризики, користувачам рекомендується заздалегідь ознайомитися з відгуками, умовами проживання та репутацією орендодавця.

Booking.com: універсальна платформа для онлайн-бронювання

Booking.com є одним із провідних світових сервісів для онлайн-бронювання житла. Заснована у 1996 році, платформа входить до складу корпорації Booking Holdings Inc. і охоплює широкий спектр послуг у сфері туризму.

Платформа пропонує великий вибір об'єктів розміщення - від готелів і гостьових будинків до апартаментів і вілл - у різних цінових категоріях. Крім житла, Booking.com дозволяє замовити:

- трансфер з аеропорту або вокзалу;
- оренду автомобіля;
- екскурсії та розваги в місці призначення.

Серед ключових переваг сервісу:

- зручний інтерфейс користувача, що адаптований до мобільних і десктопних пристроїв;
- розширені можливості пошуку з великою кількістю фільтрів;

- детальна база відгуків та оцінок, яка допомагає обрати найбільш підходящий варіант житла;
- програма лояльності (Genius), яка надає знижки та інші привілеї постійним користувачам.

Завдяки високій надійності, широкому функціоналу та зручності користування, Booking.com продовжує утримувати лідируючі позиції серед онлайн-сервісів з бронювання житла у світі.

Інтерфейс головної сторінки сервісу Booking.com представлено на рис. 1.2.

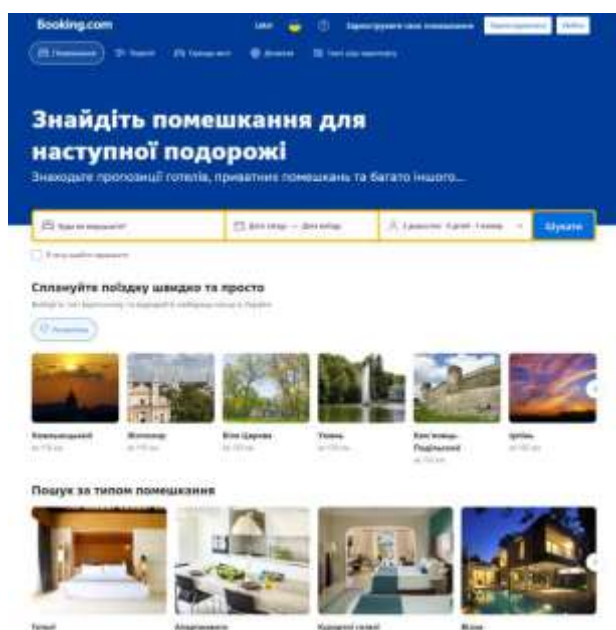


Рисунок 1.2. Інтерфейс головної сторінки «Booking.com»

Платформа Booking.com має низку суттєвих переваг, які забезпечують її популярність серед мандрівників у всьому світі. До основних сильних сторін сервісу належать:

Широкий вибір житла. Сервіс пропонує різноманітні варіанти розміщення - готелі, апартаменти, гостьові будинки та інші типи помешкань у великій кількості країн і міст.

Зручна система пошуку. Інтерфейс платформи є інтуїтивно зрозумілим і дає змогу швидко здійснювати фільтрацію пропозицій за ціною, рейтингом, місцезнаходженням, зручностями тощо.

Система відгуків і рейтингів. Користувачі мають доступ до великої бази

рецензій від попередніх гостей, що допомагає об'єктивно оцінити якість житла перед бронюванням.

Програма лояльності. Постійні клієнти можуть користуватися бонусною програмою Genius, яка забезпечує додаткові знижки та спеціальні пропозиції під час подальших бронювань.

Водночас сервіс має і певні обмеження, серед яких варто виокремити:

- Додаткові комісійні платежі. У деяких випадках до вартості проживання додаються сервісні збори, що робить бронювання через платформу дорожчим порівняно з прямим резервуванням на сайті об'єкта.

- Суб'єктивність відгуків. Хоча система рецензій є перевагою, окремі відгуки можуть бути неповними або необ'єктивними, що іноді призводить до хибних очікувань.

- Обмежена кількість доступних варіантів у пікові періоди. У високий туристичний сезон або під час масових заходів найкращі пропозиції можуть бути швидко заброньовані, що ускладнює вибір.

Незважаючи на вказані недоліки, Booking.com стабільно залишається одним із найбільш зручних і надійних інструментів для онлайн-бронювання. Потужна база помешкань, гнучкі фільтри пошуку, розвинена система зворотного зв'язку й приваблива бонусна програма формують позитивний користувацький досвід.

На тлі популярних платформ власноруч розроблений програмний застосунок для бронювання житла демонструє низку конкурентних переваг, орієнтованих на потреби цільової аудиторії.

Його інтерфейс спроектований для забезпечення:

- максимальної зручності навігації;
- естетичної привабливості;
- швидкого доступу до ключових функцій.

Користувачі мають можливість легко орієнтуватися у структурі застосунку, швидко знаходити необхідні об'єкти та здійснювати бронювання без зайвих складностей. Завдяки цьому підвищується ефективність роботи з

сервісом і покращується загальний досвід взаємодії.

Візуальне представлення інтерфейсу власного застосунку наведено на рис. 1.3.



Рисунок 1.3. Дизайн власного програмного додатку

Однак ключова особливість розробленого програмного рішення полягає не лише в його привабливому візуальному оформленні. Застосунок підтримує декілька альтернативних методів реєстрації та входу в систему, що істотно підвищує зручність користування. Така гнучкість дозволяє адаптувати процес авторизації до індивідуальних потреб та вподобань кожного користувача.

Персоналізований підхід до доступу сприяє формуванню позитивного досвіду взаємодії з сервісом, зміцнює довіру до продукту та підвищує рівень користувацької лояльності.

Порівняльний аналіз функціональних можливостей платформ Airbnb, Booking.com та створеного застосунку представлено у табл. 1.1.

Розробка власного програмного застосунку здійснювалася на основі всебічного аналізу сильних і слабких сторін провідних сервісів, зокрема Airbnb та Booking.com. У процесі створення було адаптовано найкращі практики, застосовувані на цих платформах, а також впроваджено вдосконалення в тих елементах, де було виявлено функціональні чи структурні недоліки.

Таблиця 1.1 – Порівняння аналогів з власною розробкою

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

До основних переваг створеного застосунку належать інтуїтивно зрозумілий і візуально привабливий інтерфейс, можливість обирати серед кількох способів реєстрації та входу в систему, висока швидкодія та наявність служби підтримки. Завдяки поєднанню цих характеристик застосунок демонструє високу конкурентоспроможність у сфері онлайн-бронювання житла та забезпечує користувачам зручний і приємний досвід взаємодії.

Таким чином, метою створення власного програмного продукту було розроблення сучасного, ефективного та зручного інструменту, який поєднує переваги провідних платформ, одночасно нівелюючи їх найтипівіші недоліки.

1.3 Визначення типів користувачів та ключових вимог

Для успішної реалізації програмного забезпечення для бронювання житла необхідно враховувати широкий спектр потреб і очікувань різних категорій користувачів. На початковому етапі важливо визначити ключові критерії, які формують структуру цільової аудиторії:

- Індивідуальні потреби та вподобання – залежно від мети подорожі, бюджету й особистих переваг, користувачі можуть обирати як бюджетне житло, так і преміум-варіанти.

- Технічні особливості – користувачі взаємодіють із сервісом через різні пристрої: смартфони, планшети або комп'ютери. Це зумовлює необхідність забезпечення адаптивного та зручного інтерфейсу.

- Аналіз поведінки – дані про попередні дії користувача, його місце перебування, демографічні характеристики допомагають персоналізувати

функціонал та підвищити релевантність пропозицій.

- Частота використання – доцільно розділяти постійних активних користувачів та тих, хто використовує сервіс епізодично, з метою налаштування відповідних сценаріїв взаємодії.

На основі цих параметрів можна виокремити основні типи користувачів:

- Туристи – зазвичай шукають комфортне житло поблизу туристичних місць, зручну навігацію по карті, фотогалереї об'єктів і просту систему пошуку.

- Сімейні мандрівники – віддають перевагу просторим квартирам із кухнею, кількома спальнями, Wi-Fi, умовами для дітей.

- Бізнес-користувачі – очікують стабільного інтернету, наявності робочої зони, можливості трансферу. Важливими є швидкість і надійність бронювання.

- Мандрівники з обмеженим бюджетом – шукають просте, доступне житло з базовими зручностями й вигідним розташуванням.

Урахування потреб кожного сегмента дозволяє формувати персоналізовані функціональні рішення та ефективні стратегії просування:

- Туристам доцільно пропонувати інтеграцію мапи, рекомендації локацій і рекламу через туристичні ресурси.

- Сім'ям – фільтри за кількістю кімнат, наявністю кухні чи дитячих зручностей, реклама через тематичні видання.

- Бізнес-клієнтам – просування через ділові платформи, підтримка швидкого бронювання й виставлення рахунків.

- Бюджетним мандрівникам – акцент на знижки, добірки вигідних пропозицій, реклама на спеціалізованих сайтах.

Індивідуальний підхід не лише приваблює нових користувачів, а й сприяє утриманню постійних завдяки зручності, адаптованості функціоналу та фокусуванню на реальні потреби.

Водночас важливо не ігнорувати менш поширені категорії, зокрема користувачів із тваринами, осіб з інвалідністю тощо. Інклюзивність функціоналу дозволяє розширити охоплення платформи та підвищити її соціальну значущість.

Зібраний аналітичний матеріал стає основою для подальшого вдосконалення системи. Наприклад, якщо серед користувачів переважають бізнес-туристи, доцільно впровадити функції групового бронювання чи автоматичного формування звітів. Для туристів – удосконалити систему рекомендацій та навігації по об’єктах.

Таким чином, розуміння структури цільової аудиторії та її очікувань дозволяє створити максимально ефективне, адаптивне й зручне програмне забезпечення, орієнтоване на реальні умови його використання.

Окрім цього, важливу роль у реалізації якісного програмного продукту відіграє вибір середовища розробки. Для створення цього застосунку було обрано Visual Studio Code, зображене на рис. 1.4.

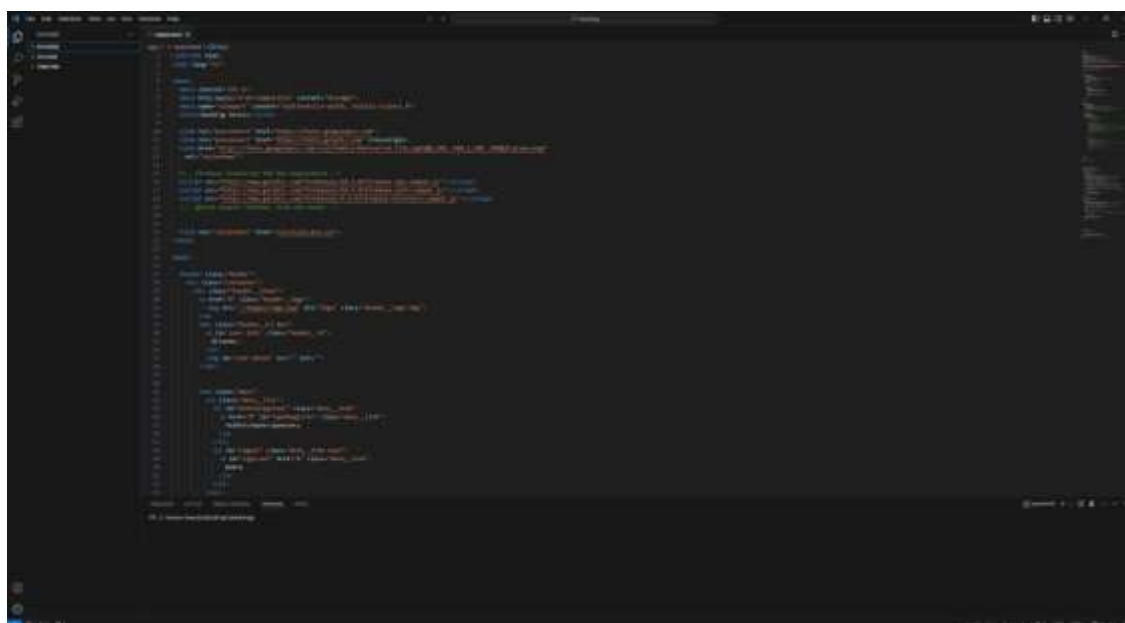


Рисунок 1.4. Інтерфейс середовища Visual Studio Code

Вибір середовища розробки Visual Studio Code (VS Code) для створення системи бронювання житла був зумовлений рядом вагомих переваг, які відповідають сучасним вимогам до гнучкості, продуктивності й безпеки програмних рішень:

- Розширена функціональність. VS Code надає повноцінний набір інструментів для ефективної розробки: інтелектуальне автодоповнення коду, інтегрована система контролю версій Git, зручне зневадження, підтримка

численних мов програмування тощо. Усе це сприяє пришвидшенню процесу написання, тестування та оптимізації коду.

- Гнучка система розширень. Завдяки великому каталогу розширень, середовище легко адаптується під специфіку будь-якого проєкту. Під час розробки застосунку використовувалися плагіни для роботи з JavaScript, Firestore, інструменти аналізу якості коду та автоматичного форматування, що підвищило ефективність роботи.

- Сумісність із сучасними технологіями. VS Code забезпечує глибоку інтеграцію з Firebase та Firestore - ключовими складовими застосунку. Це дозволяє швидко налаштовувати зв'язок із базою даних, ефективно здійснювати запити й обробку даних.

- Висока продуктивність. Незважаючи на багатий функціонал, середовище зберігає швидкодію та стабільність, навіть у роботі з великими проєктами, що позитивно впливає на продуктивність розробника.

- Платформна незалежність. VS Code підтримується на Windows, macOS і Linux, що дає змогу організувати зручну командну роботу незалежно від операційної системи кожного учасника.

- Активна спільнота. Потужна база користувачів і розробників сприяє швидкому виявленню та усуненню проблем, регулярним оновленням функцій, а також забезпечує доступ до великої кількості навчальних матеріалів, документації та шаблонів.

- Інтеграція з хмарною інфраструктурою. Середовище безперешкодно взаємодіє з хмарними сервісами, що особливо актуально для масштабованих вебзастосунків. У даному проєкті інтеграція з Firebase дозволила забезпечити зручне зберігання даних та високу доступність сервісу.

- Підтримка широкого спектра мов. VS Code сумісний не лише з JavaScript, а й з багатьма іншими мовами та фреймворками, що відкриває перспективи подальшого розвитку функціоналу системи.

- Безпека. Середовище підтримує сучасні засоби захисту, включаючи SSL/TLS-протоколи для безпечної взаємодії з віддаленими серверами, що є

критично важливим при роботі з персональними даними користувачів.

З огляду на ці переваги, Visual Studio Code став оптимальним вибором для реалізації програмного продукту. Його функціональна гнучкість, підтримка сучасних технологій, стабільна продуктивність і високий рівень безпеки створюють умови для розробки ефективного, масштабованого й конкурентоспроможного застосунку. У табл. 1.2 буде представлено порівняльний аналіз VS Code з іншими середовищами розробки.

Серед розглянутих рішень саме Visual Studio Code отримує найвищі оцінки завдяки поєднанню багатофункціональності, підтримки великої кількості мов програмування, гнучкої системи розширень та інтеграції з хмарними платформами. Такий набір характеристик робить його універсальним інструментом для розробників різного рівня та спеціалізації.

Таблиця 1.2 – Порівняння середовищ розробки

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

IntelliJ IDEA виділяється як потужне середовище розробки, особливо для проєктів на платформі JVM. Проте його функціонал зосереджений на обмеженому наборі мов програмування, що робить його менш гнучким порівняно з VS Code.

Atom, попри широкі можливості кастомізації, поступається VS Code у ключових аспектах, таких як продуктивність і інтеграція з хмарними технологіями, що знижує його привабливість для сучасних проєктів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Створення вимог до програмного забезпечення

Проектування архітектури системи є ключовим етапом створення програмного забезпечення для платформи бронювання житла. Цей розділ описує низку рішень і дій, спрямованих на побудову структурної основи системи, яка відповідатиме функціональним вимогам і забезпечуватиме її стабільну та ефективну роботу.

Перший етап передбачає визначення основних функціональних і нефункціональних вимог до застосунку. Перед цим рекомендується проаналізувати типові сценарії взаємодії з системою з позицій як користувачів, так і адміністраторів.

На рис. 2.1 зображена діаграма варіантів використання, яка відображає можливі дії користувача під час роботи з системою.



Рисунок 2.1. Діаграма варіантів використання системи користувачем

Далі розглянемо сценарії взаємодії з системою з позиції адміністратора. На рис. 2.2 зображена діаграма варіантів використання, яка ілюструє функціональні можливості адміністратора в межах системи.



Рисунок 2.2. Діаграма варіантів використання системи адміністратором

Визначення функціональних і нефункціональних вимог до системи бронювання житла

Формулювання вимог до програмного забезпечення платформи для бронювання житла - це критично важливий етап. Саме тут ми визначаємо основні можливості та обмеження системи, що безпосередньо впливають на її архітектуру, реалізацію та подальшу експлуатацію.

Розпочнімо з функціональних вимог. Вони описують основні операції, які система повинна виконувати для забезпечення повноцінної взаємодії з користувачем:

Функція бронювання: Користувачі повинні мати можливість швидко та інтуїтивно зрозуміло бронювати житло (рис. 2.3).

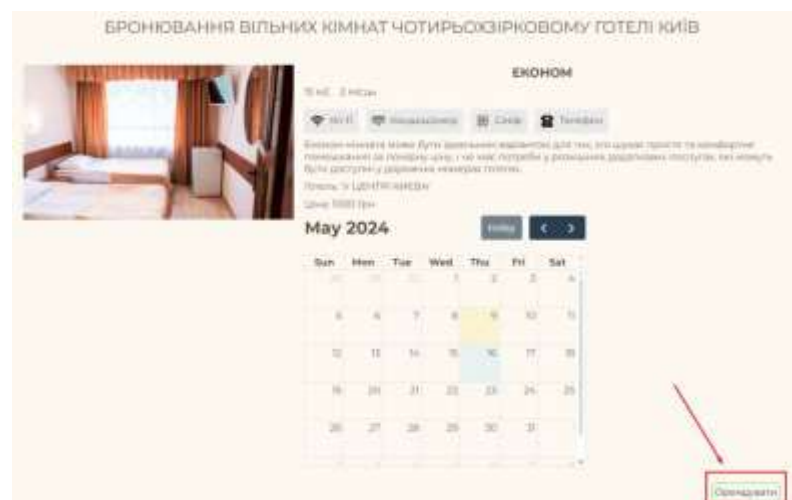


Рисунок 2.3. Оренда житла

Орендувати житло можуть тільки авторизовані користувачі.

- Можливість відмінити бронювання (рис. 2.4).



Рисунок 2.4. Відміна оренди житла

Управління бронюваннями та інформацією про житло

Після того, як користувач успішно забронював житло, адміністратор зв'язується з ним, щоб уточнити деталі та обговорити подальші кроки.

Якщо користувач вирішить скасувати бронювання, він повинен повідомити про це адміністратора. Адміністратор, у свою чергу, матиме можливість скасувати відповідну оренду в системі.

Крім того, система має надавати функціонал для перегляду повної інформації про вибране житло. Це включає фотографії та детальний опис характеристик об'єкта (рис. 2.5).



Рисунок 2.5. Інформація про місце проживання

Детальна інформація про об'єкт та адміністративне управління

Користувачі матимуть доступ до розширеної інформації як про об'єкт розміщення загалом, так і про конкретну кімнату, яка їх зацікавила. Це включає перегляд фотографій, детальний опис житла та його характеристики.

2. Управління системою

Для власників готелів або орендодавців передбачено адміністративний інтерфейс. Він дозволяє керувати процесом бронювання, контролювати доступність номерів, а також редагувати або видаляти об'єкти (рис. 2.6).

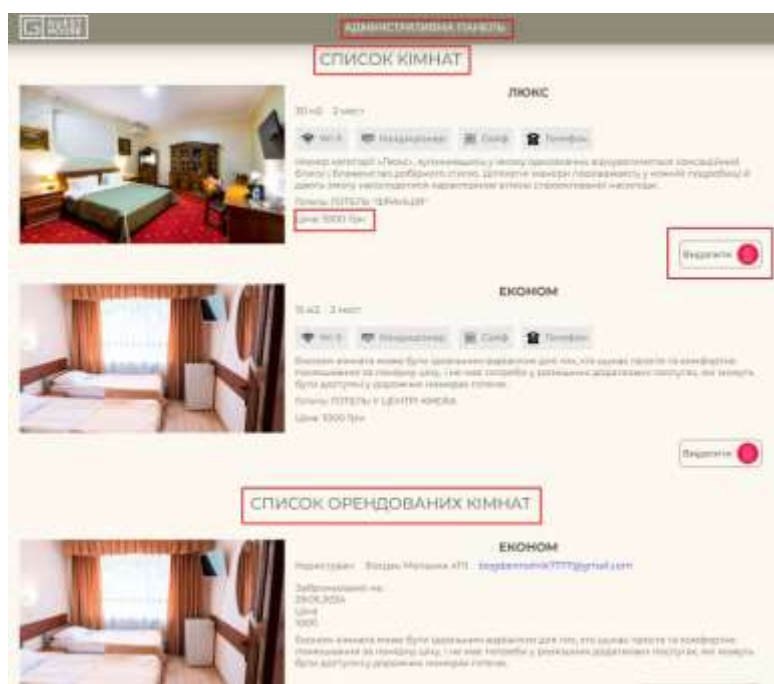


Рисунок 2.6. Панель адміністратора

Доступ до адміністративної панелі та додавання житла

Щоб увійти до адміністративної панелі, вам потрібно ввести "admin_panel" перед розширенням ".html" в адресному рядку. Після цього введіть код доступу та натисніть "увійти".

Система також надає функціонал для додавання нового житла (рис. 2.7).



Рисунок 2.7. Функціонал адміністратора

Нефункціональні вимоги до системи

Далі ми розглянемо нефункціональні вимоги, які є не менш важливими для успішної роботи системи:

Зручність використання: Інтерфейс системи повинен бути інтуїтивно зрозумілим та простим як для кінцевих користувачів, так і для адміністраторів.

Надійність: Система має забезпечувати безперебійну роботу, мінімізуючи ризики збоїв та відмов.

Безпека: Необхідно гарантувати надійний захист персональних даних користувачів.

Додатково, ми пропонуємо альтернативний варіант реєстрації та входу для користувачів (рис. 2.8).

Рисунок 2.8. Варіанти входу / реєстрації

Варіанти реєстрації та входу

Аналізуючи популярні платформи, такі як Airbnb та Booking.com, ми вирішили впровадити два варіанти реєстрації та входу в систему:

- Традиційний: за допомогою електронної пошти та пароля.
- Через обліковий запис Google.
- Вимоги до швидкості та продуктивності системи

Ми ставимо високі вимоги до швидкості роботи системи, прагнучи оптимізувати час відгуку програми. Це забезпечить швидке бронювання та оперативне надання відповідей користувачам.

Вимоги до продуктивності включають:

- Максимальна пропускна здатність для одночасних користувачів: Система повинна підтримувати оптимальну кількість одночасних користувачів, забезпечуючи безперебійну роботу навіть у пікові навантаження.

- Швидкість реакції системи: Мінімізація часу обробки запитів для покращення користувацького досвіду.

Ці вимоги є ключовими для успішної розробки програмного забезпечення для бронювання житла, яке відповідає сучасним потребам та очікуванням користувачів.

Формування вимог: комплексний підхід

Процес формування вимог - це складний і відповідальний етап, що вимагає ретельного аналізу потреб користувачів, конкурентного середовища та технічних можливостей. Наша основна мета - створити продукт, який максимально відповідає ринковим вимогам, є зручним у використанні, володіє високою якістю та ефективністю.

Функціональні вимоги визначають основні можливості та функції системи, тоді як нефункціональні вимоги описують її якісні характеристики, такі як швидкодія, безпека та доступність. Обидва типи вимог є однаково важливими, адже вони безпосередньо впливають на користувацький досвід та надійність роботи програмного забезпечення.

Отже, розробка чітких функціональних і нефункціональних вимог є критично важливим кроком для створення надійного, ефективного та зручного продукту. Врахування цих вимог сприятиме успішному впровадженню проєкту та задоволенню потреб користувачів.

2.2 Розробка архітектури системи

Розробка архітектури програмної системи - це ключовий етап у процесі її

створення. Саме тут ми визначаємо загальну структуру системи, її основні компоненти та взаємозв'язки між ними. Головна мета - сформулювати концептуальний та технічний план, який задасть основні принципи та підходи для подальшої розробки програмного забезпечення.

Для реалізації веб-системи ми обрали трирівневу архітектуру. Вона складається з трьох основних компонентів:

- Клієнтський додаток
- Сервер додатку
- Сервер бази даних

Клієнтський додаток взаємодіє з сервером додатку, який, у свою чергу, обробляє запити та спілкується із сервером бази даних.

Структурна схема системи наведена на рис. 2.9.



Рисунок 2.9. Структурна схема системи

Архітектура веб-системи

Наша веб-система побудована за трирівневою архітектурою, що складається з трьох ключових компонентів:

Клієнтський додаток (Frontend): Це інтерфейс, з яким безпосередньо взаємодіє користувач. Він розроблений з використанням HTML, CSS та JavaScript. Завдання фронтенду - відправляти запити до сервера додатку та відображати отриману інформацію.

Сервер додатку (Backend): Цей компонент відповідає за обробку запитів від клієнтського додатку. Він містить всю бізнес-логіку та забезпечує доступ до бази даних. У нашому проєкті роль сервера додатку виконує Firebase, який надає API для взаємодії як з клієнтом, так і з базою даних.

Сервер баз даних (Database Server): Це сервер, на якому розміщена база

даних Firestore. До нього звертаються як клієнтський додаток, так і сервер додатку для збереження та отримання даних.

Така архітектура дозволяє ефективно розробляти, масштабувати та підтримувати веб-застосунки з великим функціоналом та обсягом даних. Клієнтський додаток відповідає за користувацький інтерфейс, сервер додатку - за бізнес-логіку та обробку даних, а сервер баз даних - за зберігання й управління інформацією.

Візуалізація процесу бронювання

Для кращого розуміння процесу бронювання житла, ми розробили блок-схему алгоритму роботи програми. Вона наочно демонструє основні кроки та логіку функціонування системи.

Блок-схема алгоритму наведена на рис. 2.10.

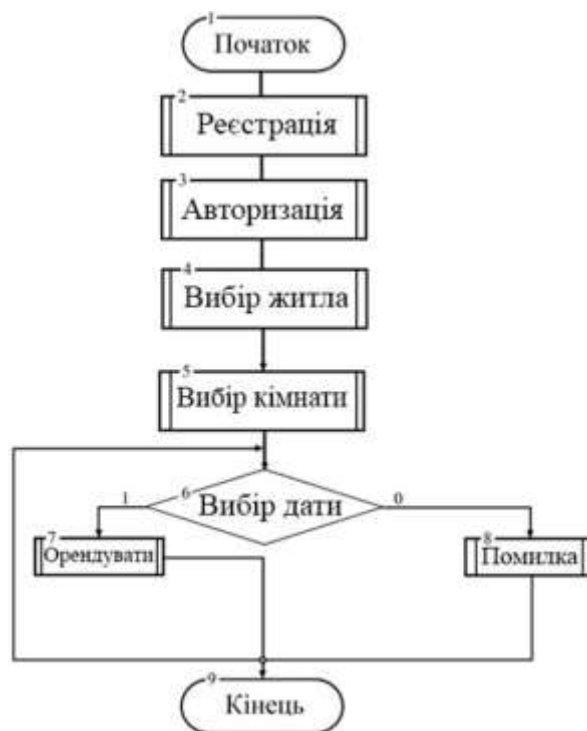


Рисунок 2.10. Блок-схема алгоритму роботи програми

Алгоритм роботи програми: Кроки користувача

Ось покроковий опис взаємодії користувача з програмою:

Початок.

Користувач спочатку має зареєструватися.

Наступним кроком є авторизація користувача.

Далі клієнт повинен обрати житло, в якому він хотів би заселитися.

Після вибору житла користувач має обрати кімнату, яка його зацікавила.

Після вибору кімнати йому буде доступний вибір дати (рис. 2.11).



Рисунок 2.11. Вибір дати оренди кімнати

Продовження алгоритму: Вибір дати та підтвердження

Після цих кроків користувач зможе знову обрати дату оренди.

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

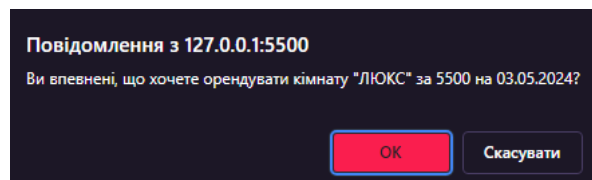


Рисунок 2.12. Повідомлення з підтвердженням оренди

Успішне бронювання та обробка зайнятих дат

Після успішної оренди кімнати клієнт має гнучкість: він може забронювати ту ж саму кімнату на інші дні або ж обрати інші кімнати.

Якщо ж клієнт обере вже зайняту дату, з'явиться повідомлення про помилку, яке інформує його про те, що ця дата вже недоступна (рис. 2.12).

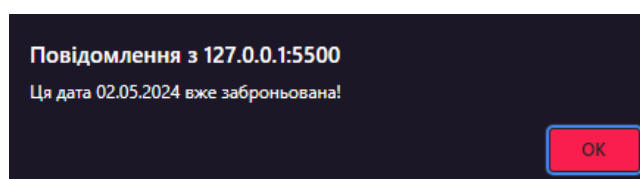


Рисунок 2.12. Повідомлення про те, що дата вже зайнята

Завершення програми.

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

2.3 Проєктування інтерфейсу користувача

Проєктування користувацького інтерфейсу (UI) - це один із ключових етапів у розробці програмного забезпечення. Саме через UI відбувається взаємодія користувача із системою. На цьому етапі ми визначаємо зовнішній вигляд, організацію та функціональність інтерфейсу, зважаючи на потреби користувачів та їхню зручність. Наша головна мета - створити інтуїтивно зрозумілий, зручний та привабливий інтерфейс, який забезпечить комфортне й ефективне використання програми.

Важливі критерії проєктування інтерфейсу:

- Візуальна привабливість: Естетичний дизайн, що створює позитивне враження.
- Легкість навігації: Простота переміщення між розділами та зрозумілість дій.
- Наявність чітких інструкцій: Підказки, які допомагають користувачу швидко освоїтися.
- Швидкий доступ до основних функцій: Мінімізація кількості кроків для виконання завдань.

Такий підхід підвищує задоволення користувачів і сприяє успішній інтеграції продукту на ринку.

При розробці інтерфейсу для нашої програми бронювання житла ми врахували наступні аспекти:

- Інтуїтивність і простота використання: Інтерфейс має бути зрозумілим і легким у користуванні, з логічною навігацією, послідовністю дій та чіткими позначеннями кнопок і елементів.

- **Мінімалізм:** Дизайн виконано в мінімалістичному стилі, щоб уникнути інформаційного перевантаження. Прості форми та елементи допомагають користувачу зосередитись на основних завданнях.

- **Привабливий дизайн:** Естетична привабливість інтерфейсу досягається завдяки гармонійному поєднанню кольорів, зручного шрифту та якісних зображень.

- **Функціональність:** Інтерфейс повинен забезпечувати всі необхідні можливості для бронювання житла, включно з переглядом варіантів проживання та вибором дат бронювання, без зайвих ускладнень.

Враховуючи ці принципи, проектування інтерфейсу користувача спрямоване на створення зручного, ефективного та привабливого інструменту, що дозволить користувачам легко знаходити та бронювати житло.

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

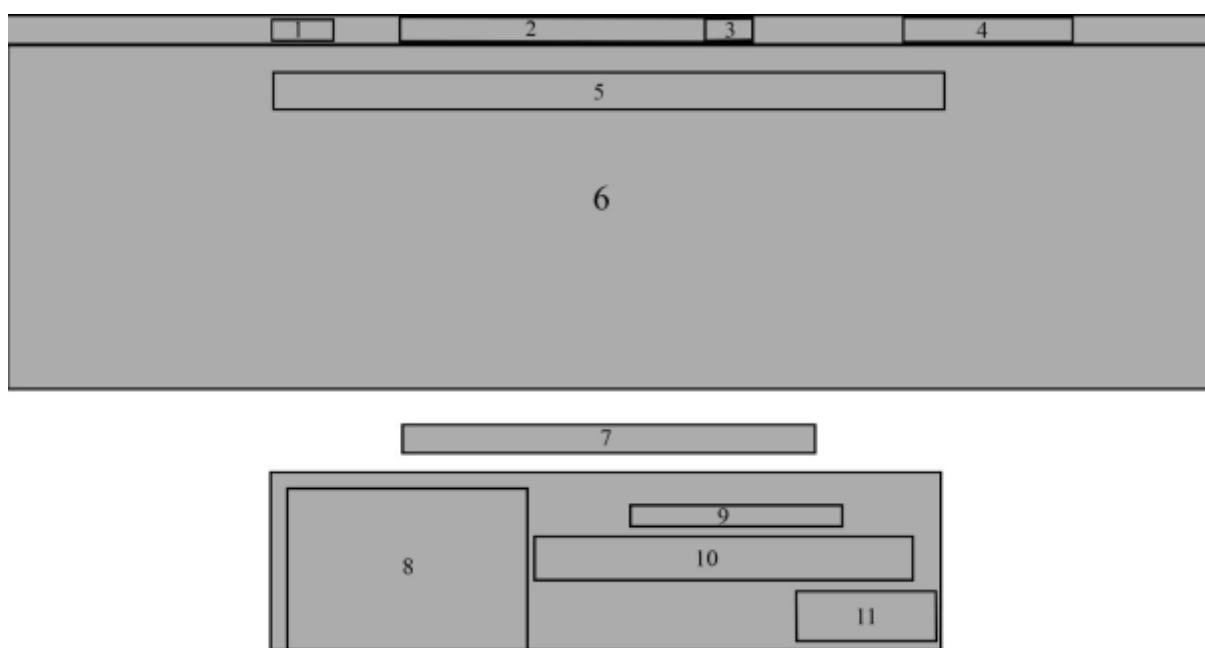


Рисунок 2.14. Макет головної сторінки застосунку

Елементи головної сторінки та її функціонал

Головна сторінка нашого застосунку містить наступні ключові елементи:

Логотип: Бренд компанії.

Ім'я користувача: Відображає ім'я поточного користувача.

Аватар користувача: Візуальне представлення користувача.

Кнопка для входу, виходу або реєстрації: Забезпечує доступ до системи.

Ключова фраза: Привертає увагу та підкреслює переваги.

Головне фото: Основне візуальне зображення.

Список доступних варіантів проживання: Огляд пропозицій.

Фото кожного місця проживання: Візуалізація об'єктів.

Назва місця проживання: Ідентифікатор об'єкта.

Короткий опис житла: Стисла інформація про об'єкт.

Кнопка «Переглянути кімнати»: Для детального ознайомлення з кімнатами обраного місця.

При першому відкритті застосунку, користувач бачить ключову фразу, головне фото та верхню частину інтерфейсу. Ключова фраза призначена для привернення уваги, створення емоційного зв'язку з аудиторією та підкреслення основних переваг продукту.

Прогорнувши сторінку вниз, користувач може ознайомитися зі списком доступних варіантів проживання та їхніми короткими описами. Щоб детально переглянути доступні кімнати, достатньо натиснути кнопку «Подивитись кімнати» поруч із обраним житлом.

Наступним кроком ми розглянемо сторінку конкретного житла (рис. 2.15).

Сторінка конкретного житла: Детальний огляд

Сторінка конкретного житла надає користувачам вичерпну інформацію про обраний об'єкт. Вона містить такі елементи:

Логотип: Для навігації та ідентифікації бренду.

Ім'я користувача: Відображення активного користувача.

Аватар користувача: Персональне зображення користувача.

Кнопка для входу, виходу або реєстрації: Універсальна кнопка для керування сесією.

Назва місця проживання: Чітка ідентифікація об'єкта.

Фото місця проживання: Візуальне представлення житла.

Повний опис житла: Детальна інформація про об'єкт.

Список доступних для бронювання кімнат: Перелік усіх пропозицій.

Тип кімнати: Категорія кімнати (наприклад, "економ", "люкс").

Фото кімнати: Зображення конкретної кімнати.

Площа кімнати: Розмір кімнати.

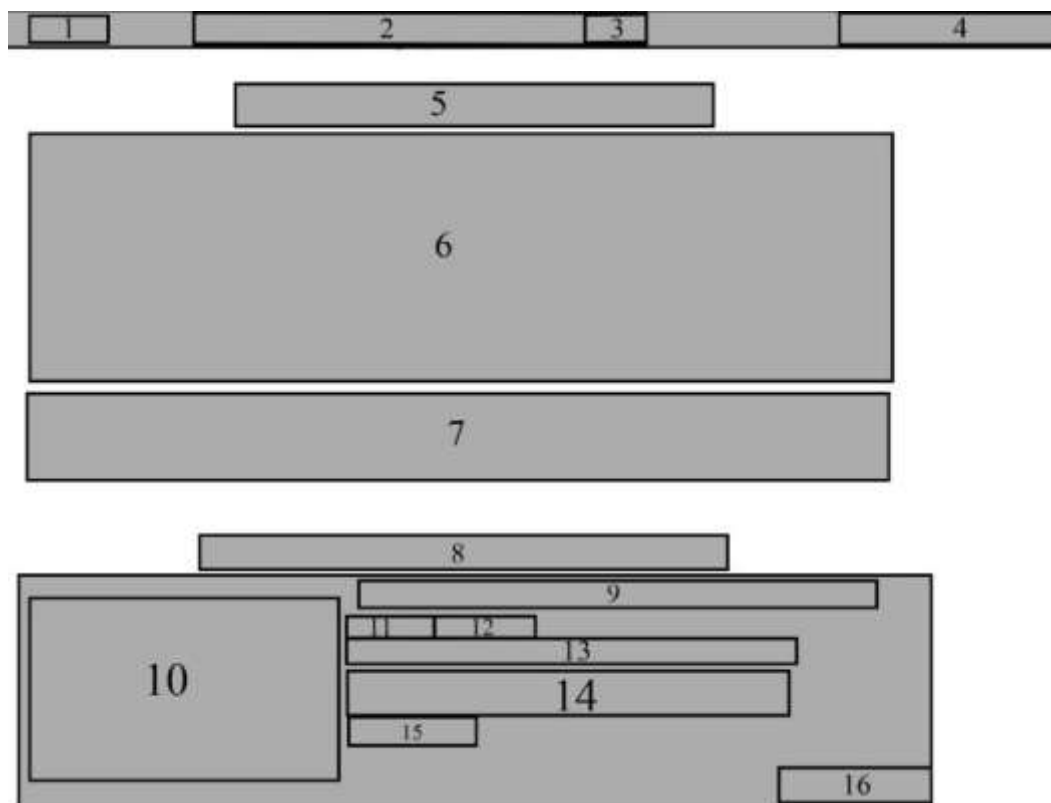


Рисунок 2.15. Макет сторінки житла

Кількість місць для проживання: Максимальна кількість осіб, які можуть розміститися.

Перелік доступних послуг, що входять в оренду кімнати: Додаткові зручності.

Опис кімнати: Детальний опис особливостей кімнати.

Ціна оренди: Вартість бронювання.

Кнопка «Орендувати»: Для ініціації процесу бронювання.

На цій сторінці користувач бачить назву, фото та детальний опис обраного місця проживання. Нижче розташований перелік доступних для бронювання кімнат з детальною інформацією про кожну з них. Ця інформація включає тип кімнати (наприклад, економ, люкс), площу, кількість місць для проживання,

короткий опис, ціну оренди та фотографії кімнат.

Поруч із кожною кімнатою розміщена кнопка «Орендувати», яка доступна лише авторизованим користувачам. Якщо користувач не увійшов в акаунт, замість кнопки відображається повідомлення: «Щоб почати орендувати, увійдіть в акаунт». Після натискання кнопки «Орендувати» відкривається календар, де користувач може обрати дату бронювання.

Макет сторінки адміністратора

Далі розглянемо макет сторінки адміністратора. Для входу на цю сторінку необхідний спеціальний код доступу. Використовуючи код «213134421», користувач отримує доступ до панелі адміністратора. Макет сторінки зображено на рис. 2.16. На ньому можна побачити основні елементи управління, такі як кнопки додавання та видалення кімнат, а також інші важливі функції адміністрування.

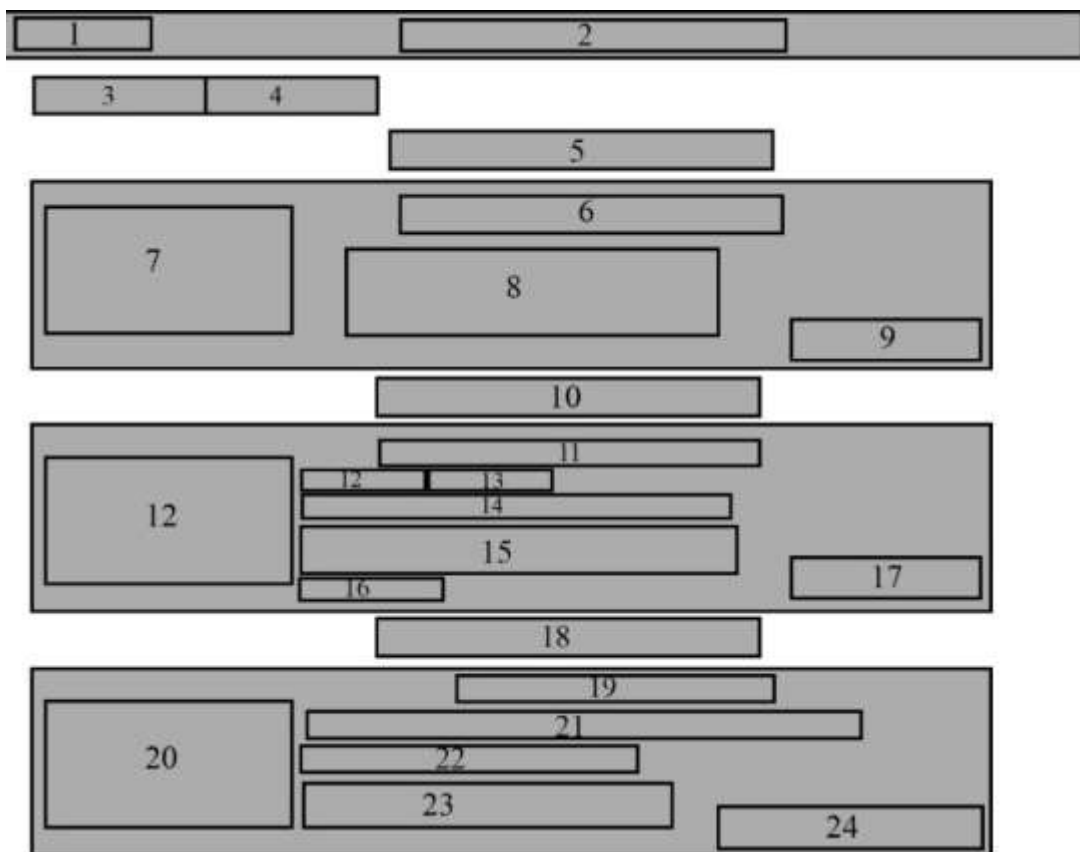


Рисунок 2.16. Макет сторінки адміністратора

Елементи сторінки адміністратора

Сторінка адміністратора розроблена для ефективного управління системою бронювання житла та містить такі елементи:

Логотип: Для навігації та ідентифікації бренду.

Панель адміністратора: Загальний заголовок розділу.

Кнопка додавання житла: Дозволяє додавати нові об'єкти.

Кнопка додавання кімнати: Дозволяє додавати нові кімнати до існуючих об'єктів.

Список місць проживання: Огляд усіх доступних об'єктів.

Назва місця проживання: Ідентифікатор конкретного об'єкта.

Фото місця проживання: Візуальне представлення об'єкта.

Опис місця проживання: Детальна інформація про об'єкт.

Кнопка видалення місця проживання: Для видалення об'єкта із системи.

Список доступних кімнат: Перелік усіх кімнат.

Тип кімнати: Категорія кімнати (наприклад, "люкс", "стандарт").

Площа кімнати: Розмір кімнати.

Кількість місць у кімнаті: Максимальна кількість осіб.

Перелік доступних послуг: Додаткові зручності, включені в оренду.

Опис кімнати: Детальна інформація про кімнату.

Ціна оренди: Вартість бронювання кімнати.

Кнопка видалення кімнати: Для видалення кімнати із системи.

Список орендованих кімнат: Огляд усіх активних бронювань.

Тип орендованої кімнати: Категорія заброньованої кімнати.

Фото орендованої кімнати: Зображення заброньованої кімнати.

Дані користувача, що орендував кімнату: Ім'я та електронна пошта клієнта.

Дата бронювання та ціна: Деталі бронювання.

Опис орендованої кімнати: Додаткова інформація про заброньовану кімнату.

Кнопка скасування оренди: Дозволяє адміністратору скасувати бронювання.

На початку роботи адміністратор бачить список місць проживання,

розміщених на головній сторінці застосунку, разом з їх описом і кнопкою видалення, розташованою поруч із кожним об'єктом.

Прогорнувши сторінку нижче, адміністратор може переглянути список усіх доступних кімнат з повною інформацією - площею, описом, ціною, переліком послуг тощо. Поруч із кожною кімнатою розміщена кнопка для її видалення.

Ще нижче відображається список кімнат, які були орендовані користувачами. Тут міститься інформація про кімнату (фото, опис) та дані клієнта (ім'я, електронна пошта), що дозволяє зв'язатися з ним для вирішення питань щодо оренди. Кожна орендована кімната має кнопку для скасування оренди.

Для запобігання випадковому видаленню будь-якого елемента, перед його видаленням система виводить підтвердження у вигляді спливаючого вікна з кнопкою «Ок».

Основними функціями адміністратора є додавання житла та кімнат. Відповідні кнопки розташовані у верхній частині панелі адміністратора.

Процес додавання кімнати

Далі розглянемо процес додавання кімнати на прикладі, наведеному на рис. 2.17.

Для додавання кімнати спочатку обираємо житло, до якого буде додаватися кімната. Потім вводимо назву (тип) кімнати, вказуємо кількість місць (кількість людей, які можуть проживати у цій кімнаті), а також площу кімнати. Після цього додаємо опис, вказуємо ціну оренди та додаємо посилання на фото кімнати. Натискаємо кнопку «Додати», після чого для підтвердження дії вводимо код адміністратора.

Проектування користувацького інтерфейсу (UI) - це ключовий етап у створенні програмного забезпечення, адже саме він визначає спосіб взаємодії користувача із системою. Для нашого веб-додатку бронювання житла ми зосередилися на таких аспектах, як інтуїтивність використання, мінімалістичний дизайн, адаптивність до різних браузерів, привабливий вигляд та повна

функціональність.

ДОДАВАННЯ КІМНАТИ

Виберіть житло
ГОТЕЛЬ 'ФРАНЦІЯ'

Додайте назву
Назва

Додайте кількість місць
Кількість місць

Додайте площу
Площу

Додайте опис
Опис

Додайте ціну
Ціна

Додайте посилання на картинку
Посилання

ДОДАТИ

Рисунок 2.17. Додавання кімнати

Наша головна мета - створити зручний, ефективний та привабливий інтерфейс, що забезпечить комфортне користування застосунком. Ретельне проєктування допомагає нам розробити інструмент, який максимально задовольняє потреби користувачів та підвищує їхнє задоволення від використання.

Під час розробки інтерфейсу важливо враховувати потреби та очікування цільової аудиторії. Це включає:

Розміщення основних функціональних елементів на зручних і помітних місцях.

Використання зрозумілих іконок та символів.

Забезпечення інтуїтивної взаємодії з додатком.

Крім того, проєктування передбачає застосування найкращих практик веб-дизайну:

Чіткі, послідовні шрифти для кращої читабельності.

Підбір кольорової палітри, яка не лише приваблює, а й сприяє візуальному

комфорту та підтримує єдиний стиль.

Правильне розміщення елементів на сторінці, що визначає логіку та зручність навігації.

Загалом, продумане проєктування інтерфейсу користувача дозволяє створити програмне забезпечення, яке не лише ефективно виконує свої функції, а й дарує приємний досвід користувачам. Інтуїтивно зрозумілий та простий в управлінні інтерфейс сприяє комфортному використанню та позитивному сприйняттю програми.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Аналіз технологій та методів для розробки програмного забезпечення

Давайте розглянемо технології та методи, які ми застосували під час розробки програмного забезпечення для бронювання житла, а також порівняємо їх із можливими альтернативами.

1. Створення структури

Для формування структури та контенту веб-сторінок ми обрали технологію HTML (Hypertext Markup Language). У контексті нашої системи бронювання житла HTML забезпечує створення користувацького інтерфейсу, включно з формами для бронювання та відображенням результатів пошуку.

Альтернатива HTML:

Альтернативою HTML може бути Markdown - легкий текстовий формат, призначений для створення веб-контенту. Зазвичай Markdown використовують для написання документації, блогів, коментарів у репозиторіях Git тощо. Він вирізняється простотою синтаксису, високою читабельністю та можливістю легко конвертуватися в HTML для подальшого відображення у браузері.

Порівняння HTML та Markdown наведено в табл. 3.1.

Таблиця 3.1 – Порівняння HTML і Markdown

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

2. Препроцесор

Для розробки стилів нашого програмного забезпечення для бронювання житла ми обрали SCSS (Sass CSS). Це розширення CSS, яке дозволяє використовувати змінні, вкладені стилі та інші функції, що значно спрощує

процес. Завдяки SCSS, стильове оформлення стає більш організованим, узгодженим і легким для підтримки.

Альтернатива SCSS:

Альтернативою SCSS є Less. Це також CSS-препроцесор, який додає до CSS змінні, міксіни, вкладеність та інші корисні можливості для покращення структурованості та простоти керування стилями.

Порівняння SCSS і Less наведено в табл. 3.2.

Таблиця 3.2. Порівняння SCSS і Less

	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

Для забезпечення динамічної поведінки веб-сторінок, обробки подій, взаємодії з користувачем та зміни вмісту в реальному часі ми обрали JavaScript (JS). У нашому програмному забезпеченні для бронювання житла JavaScript використовується для валідації форм, асинхронного завантаження даних та покращення загального користувацького досвіду.

Альтернатива JavaScript:

Альтернативою JavaScript є TypeScript - сучасна мова програмування, що розширює можливості JavaScript, додаючи строгий контроль типів та інші корисні функції. TypeScript допомагає виявляти помилки ще на етапі розробки, що значно підвищує надійність і стабільність проєктів, особливо великих і складних.

Порівняння JavaScript і TypeScript наведено в табл. 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

Для керування даними в нашому програмному забезпеченні для бронювання житла ми обрали Firebase. Це комплексна платформа від Google для розробки мобільних та вебдодатків, що пропонує широкий спектр сервісів, зокрема базу даних у режимі реального часу, аутентифікацію користувачів, зберігання файлів та інші можливості. У нашому проєкті Firebase використовується для зберігання та управління даними про користувачів, житло, бронювання та інші важливі аспекти застосунку.

Альтернатива Firebase:

Альтернативою Firebase є AWS Amplify - сервіс від Amazon Web Services, що допомагає розробникам швидко запускати та масштабувати веб- та мобільні додатки. Amplify надає інструменти для автоматизації розробки, включаючи розгортання коду, управління інфраструктурою та інтеграцію з іншими сервісами AWS.

Порівняння Firebase та AWS Amplify наведено в табл. 3.4.

Аналіз технологій та методів розробки програмного забезпечення для систем бронювання житла підтверджує, що використання HTML, SCSS, JavaScript та Firebase є ефективним та доцільним рішенням.

HTML та SCSS забезпечують гнучку та зручну побудову структури й візуального оформлення веб-сторінок, даючи змогу детально керувати дизайном.

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

JavaScript додає необхідну динаміку та інтерактивність, що підвищує зручність та продуктивність користувацького досвіду.

Firebase, завдяки широкому набору сервісів, дозволяє зосередитися на реалізації функціональності додатка, звільняючи від необхідності налаштовувати та адмініструвати серверну інфраструктуру.

Такий підхід сприяє швидкій розробці, впровадженню та подальшій підтримці сучасного програмного забезпечення, що відповідає очікуванням користувачів.

3.2 Розробка алгоритмів реєстрації, авторизації та додавання кімнат

Давайте розглянемо процес розробки алгоритмів реєстрації та авторизації користувачів для нашого програмного застосунку. Алгоритм реєстрації передбачає створення нового облікового запису, тоді як авторизація відповідає за перевірку прав доступу користувача до системи.

На початковому етапі розробки алгоритмів важливо визначити послідовність дій, що відбуваються під час взаємодії користувача з додатком. Процес реєстрації включає такі кроки:

- Введення користувачем необхідних даних (наприклад, електронної пошти, пароля).
- Їх перевірку на коректність і валідність.
- Додавання нового облікового запису до бази даних.

Особливу увагу слід приділяти вимогам безпеки, щоб захистити облікові записи від можливих загроз, таких як підбір пароля або несанкціоновані методи авторизації.

Далі розглянемо блок-схеми алгоритмів реєстрації та авторизації. Спочатку представлена блок-схема алгоритму реєстрації (рис. 3.1).

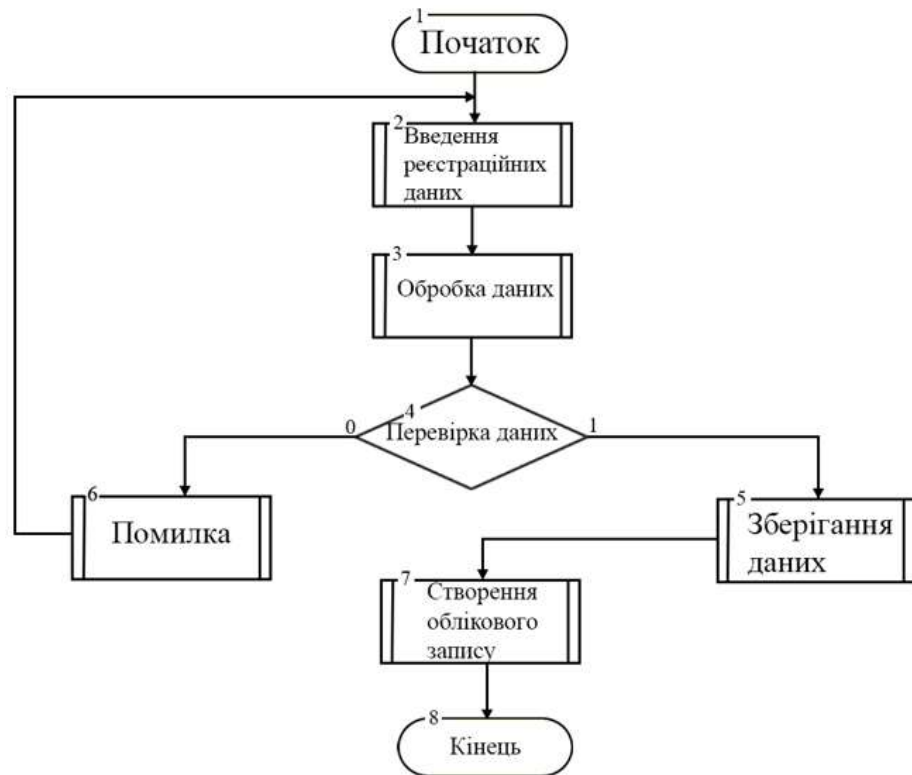


Рисунок 3.1. блок-схема алгоритму реєстрації

Для початку реєстрації, користувач повинен відкрити вікно входу/реєстрації, яке розташоване у правій верхній частині інтерфейсу.

Далі, користувач має заповнити реєстраційні дані та натиснути кнопку «Зареєструватись» (рис. 3.2).

- Потім система обробляє введені користувачем дані.

Наступним кроком програма перевіряє коректність введених даних, а також чи відсутній такий користувач у базі даних.

Якщо результат перевірки позитивний, програма зберігає введені користувачем дані до бази даних з користувачами (рис. 3.3).

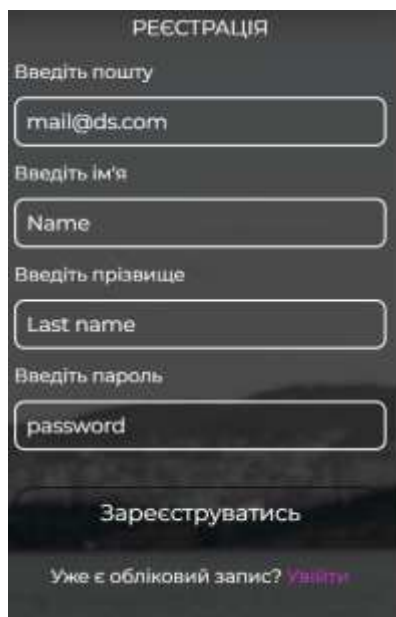


Рисунок 3.2. Вікно реєстрації

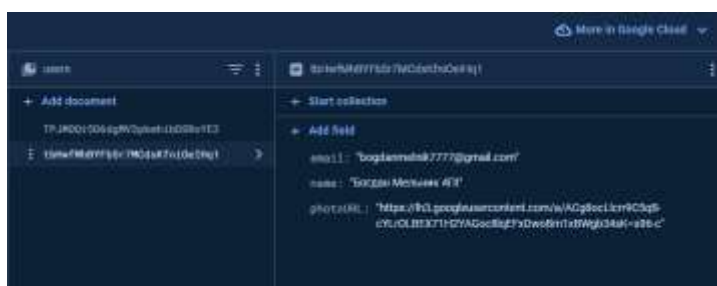


Рисунок 3.3. База даних з користувачами

Якщо ж результат перевірки негативний, користувач має перевірити введені дані та ввести їх коректно.

Після успішної перевірки та збереження даних у базі даних, програма створює новий обліковий запис користувача.

Завершення процесу реєстрації.

Далі розглянемо блок-схему алгоритму авторизації, що зображена на рис. 3.4.

Щоб увійти до облікового запису, користувач повинен відкрити вікно входу/реєстрації, розташоване у верхній правій частині інтерфейсу, і натиснути кнопку «Увійти» в нижній частині вікна реєстрації.

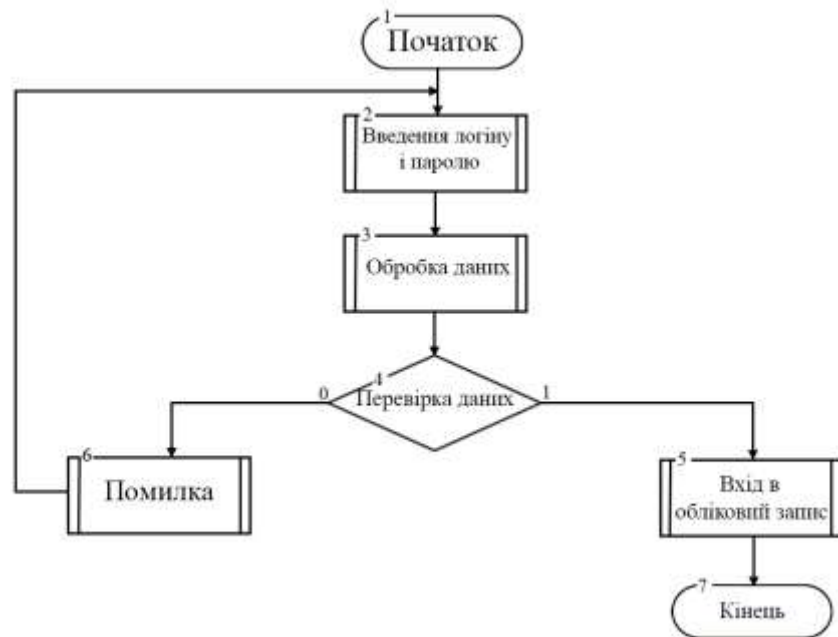


Рисунок 3.4. Блок-схема алгоритму авторизації

Далі користувач має заповнити свої дані для входу та натиснути кнопку «Увійти» (рис. 3.5).

Рисунок 3.5. Вікно входу до системи

Після введення даних користувачем, система обробляє їх.

Далі програма перевіряє коректність введеної інформації.

Якщо дані введені правильно, система виконує вхід до облікового запису користувача.

У разі виявлення помилок, користувачу пропонується перевірити та виправити введену інформацію, після чого повторити спробу входу.

Після успішної перевірки та входу завершується процес авторизації.

Реалізація алгоритмів реєстрації та авторизації користувачів - це ключовий етап у розробці програмного забезпечення. Реєстрація створює новий обліковий запис, а авторизація перевіряє права доступу до системи.

Процес реєстрації починається з введення користувачем необхідних даних, які передаються на обробку. Програма перевіряє правильність введених даних і з'ясовує, чи існує вже користувач з такими параметрами в базі даних. Якщо інформація коректна і обліковий запис ще не створено, новий користувач реєструється та зберігається в базі.

У разі успішної авторизації користувач отримує доступ до функціоналу застосунку.

Окремим етапом є розробка алгоритму додавання кімнат. Цей процес передбачає створення та реєстрацію нових приміщень у системі за допомогою додатка. Алгоритм описує послідовність дій, необхідних для успішного додавання кімнати та її відображення в інтерфейсі користувача.

Блок-схема алгоритму додавання кімнати представлена на рис. 3.6.

- ☐ Початок процесу.
- ☐ На першому етапі адміністратор обирає житло, до якого потрібно додати нову кімнату.
- ☐ Далі вводиться тип кімнати (наприклад, "економ", "люкс").
- ☐ Вказується кількість місць, передбачених для проживання в кімнаті.
- ☐ Задається площа кімнати.
- ☐ Додається текстовий опис кімнати.
- ☐ Вводиться вартість оренди кімнати.
- ☐ Завантажуються фотографії кімнати.
- ☐ Програма здійснює перевірку правильності заповнення всіх полів. Якщо дані введені некоректно або якесь поле залишене порожнім, адміністратору потрібно внести виправлення.

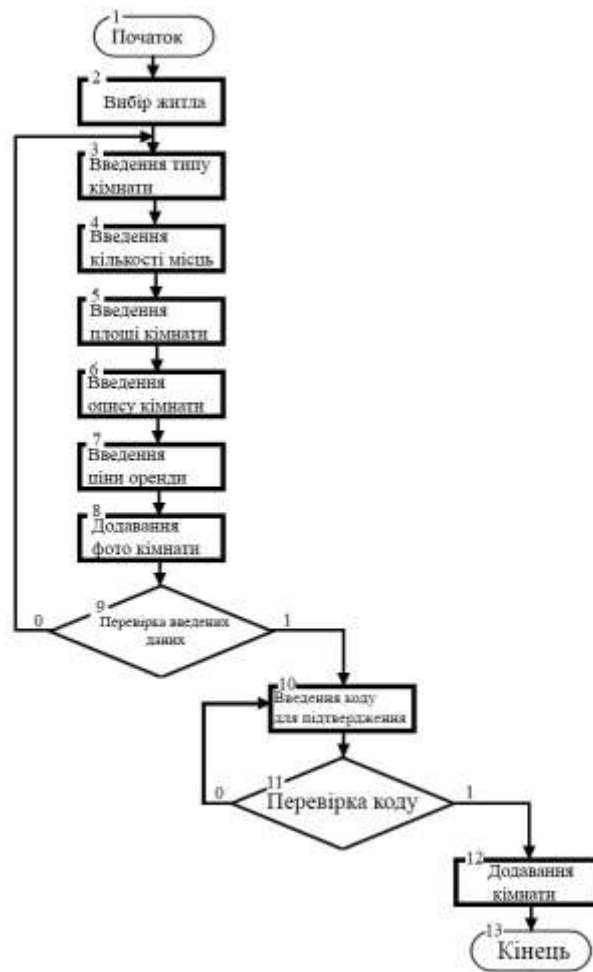


Рисунок 3.6. Блок схема алгоритму додавання кімнати

□ Після успішної перевірки, для підтвердження додавання кімнати необхідно ввести код адміністратора у спеціальне поле, яке з'явиться на екрані (рис. 3.7).

Рисунок 3.7. Поле вводу коду доступу

Після цього відбувається перевірка введеного адміністративного коду. Якщо код введено некоректно, необхідно повторити спробу введення.

У разі успішної перевірки коду система додає нову кімнату. На цьому процес додавання приміщення завершується.

Блок-схема алгоритму наочно ілюструє весь процес додавання кімнати: від вибору житла та введення характеристик приміщення до завантаження фотографій та підтвердження через адміністративний код.

Контроль правильності введених даних та верифікація коду доступу є важливою складовою цього процесу. Це гарантує безпеку операції та відповідність дій адміністратора встановленим вимогам.

Завершення процедури додавання підтверджує успішне створення нового об'єкта в системі, що, у свою чергу, дозволяє користувачам бронювати доступні приміщення через програмний додаток.

3.3 Тестування програмного додатку

Планування тестування - це ключовий етап для забезпечення якості програмного забезпечення. Воно охоплює визначення цілей, методів, ресурсів та графіку виконання тестів, що дозволяє застосувати системний підхід до виявлення та усунення помилок у програмному продукті.

Основна мета тестування - виявити помилки, недоліки та невідповідності ще до впровадження додатка в реальне середовище. Тестування також дає змогу переконатися, що додаток відповідає як функціональним, так і нефункціональним вимогам, визначеним на етапі проєктування.

Основні види тестування веб-додатків

Функціональне тестування:

- UI-тестування (тестування користувацького інтерфейсу): Перевірка взаємодії користувача з кнопками, формами, меню тощо.
- Тестування форм: Перевірка валідності введених даних і коректності їх обробки.
- Тестування навігації: Перевірка правильності переходів між сторінками додатка.
- Інтеграційне тестування:
- Взаємодія між компонентами: Перевірка коректної роботи зв'язків між модулями.

- Тестування бази даних: Перевірка запитів, цілісності даних, їх збереження та відновлення.

Тестування продуктивності:

- Навантажувальне тестування: Визначення здатності додатка працювати з великою кількістю запитів.

- Стрес-тестування: Перевірка поведінки системи в умовах перевантаження.

- Тестування масштабованості: Оцінка можливості розширення продуктивності при зростанні навантаження.

Тестування безпеки:

- Сканування вразливостей: Автоматичне виявлення відомих загроз.

- Тестування автентифікації та авторизації: Перевірка механізмів входу та управління доступом.

- Тестування захисту даних: Оцінка методів шифрування і захисту під час збереження та передачі даних.

Крос-браузерне та крос-платформне тестування:

- Сумісність з браузерами: Перевірка роботи в Chrome, Firefox, Safari, Edge тощо.

- Тестування на різних ОС: Windows, macOS, Linux, Android, iOS.

- Тестування зручності використання (Usability Testing): Тестування з реальними користувачами: оцінка зручності інтерфейсу, логіки дій, зрозумілості навігації.

- Тестування адаптивності (Responsive Testing): Тестування на різних екранах: перевірка відображення на телефонах, планшетах, моніторах з різною роздільною здатністю.

- Тестування на відповідність стандартам: Перевірка відповідності вебстандартам: оцінка додатка згідно з вимогами сучасних стандартів розробки.

Методи тестування:

Тестування «чорної скриньки» (Black Box Testing):

Суть: Тестування без доступу до коду; оцінка поведінки системи на основі

введених і очікуваних даних.

Приклади: Тестування форм, навігації, інтерфейсу.

Переваги: Не потребує знання програмування; дозволяє тестувати з точки зору кінцевого користувача.

Недоліки: Не охоплює внутрішню логіку; складно повністю покрити всі сценарії.

Тестування «білої скриньки» (White Box Testing):

Суть: Тестування з доступом до коду; перевірка внутрішньої логіки, умов, циклів тощо.

Приклади: Юніт-тести, перевірка логіки обробки даних.

Переваги: Висока точність; виявлення глибинних помилок; оптимізація коду.

Недоліки: Вимагає глибоких технічних знань; не враховує користувацький досвід.

Тестування «сірої скриньки» (Gray Box Testing):

Суть: Комбінований підхід з частковим доступом до внутрішньої структури.

Приклади: Тестування баз даних, API з обмеженим доступом до коду.

Переваги: Поєднує переваги двох попередніх методів; підходить для комплексних систем.

Недоліки: Потребує як технічних знань, так і розуміння сценаріїв використання; складність у реалізації.

Комплексне тестування дає змогу виявити та виправити помилки ще на етапі розробки, забезпечуючи: відповідність функціональним вимогам; стабільність та безпеку системи; комфортну взаємодію для користувачів; конкурентоспроможність продукту на ринку.

Особливу увагу ми приділяємо тестуванню користувацьких сценаріїв, що включає:

- Реєстрацію та авторизацію користувачів.
- Огляд і вибір житла.

- Процес бронювання.

Коректність відображення контенту на головній сторінці (табл. 3.5).

Ці тести дозволяють забезпечити зручність, інтуїтивність і надійність взаємодії кінцевого користувача з веб-додатком.

Таблиця 3.5 – Тест 1

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

На головній сторінці користувач бачить кнопки авторизації та реєстрації у верхній частині. Він також може перейти до перегляду кімнат місця проживання, яке його зацікавило.

Далі протестуємо реєстрацію та авторизацію. Створимо новий обліковий запис у системі (табл. 3.6).

Таблиця 3.6 – 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 етапом тестування є перевірка процедури авторизації користувача, створеного на попередньому етапі. Цей тест має на меті підтвердити, що система дозволяє користувачам успішно входити до облікового запису за наявності правильних облікових даних.

Перевірка включає:

- Коректне відображення форми входу.
- Успішну авторизацію при введенні правильних даних.
- Надання користувачу відповідних прав доступу після входу.

Це дозволяє підтвердити, що процес авторизації є стабільним, безпечним і відповідає функціональним вимогам. Результати тестування авторизації наведено в табл. 3.7.

Наступним критичним кроком є тестування функціоналу оренди, що є основною складовою веб-додатку. Цей етап має на меті переконатися, що користувачі можуть зручно та безпечно переглядати доступне житло, перевіряти його наявність і здійснювати бронювання.

Під час тестування перевіряється:

- Правильність відображення інформації про житло та окремі кімнати.
- Коректна робота календаря доступності.
- Наявність відображення зайнятих та вільних дат.
- Оновлення статусу кімнати після бронювання обраної дати.

Ці перевірки гарантують, що система оренди функціонує згідно з очікуваннями користувачів. Результати тестування цього етапу подано в табл. 3.8.

Результати тестування функціоналу оренди підтвердили його стабільну та коректну роботу. Користувачі без труднощів можуть переглядати інформацію про житло, перевіряти доступність кімнат на вибрані дати та здійснювати бронювання. Усі перевірки засвідчили, що система працює надійно, забезпечуючи комфорт і зручність для кінцевих користувачів.

Таблиця 3.7 – 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

Таблиця 3.8 – 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 користувацької частини веб-додатку є ключовим етапом для забезпечення якості та конкурентоспроможності продукту. Ми охопили основні функціональні можливості: реєстрацію, авторизацію та процедуру оренди. Під час перевірок не було виявлено помилок у роботі користувацького інтерфейсу, що свідчить про якісну реалізацію та стабільне функціонування системи. Тести проводилися на основі реальних сценаріїв використання, що дозволило відтворити умови, наближені до практичної експлуатації.

Тестування адміністративної частини веб-додатку має вирішальне значення для забезпечення її надійності, безпеки та ефективності. Цей модуль містить інструменти для керування контентом, налаштуванням системи, аналітикою та іншими важливими компонентами, тому його тестування потребує особливої уваги.

Адміністративна панель відіграє центральну роль у підтримці працездатності системи та задоволенні потреб адміністраторів. Вона надає можливості для: модерації користувачів; оновлення інформації; моніторингу активності; налаштування параметрів, що визначають функціонування веб-додатку.

Метою тестування адміністративного модуля є своєчасне виявлення й усунення потенційних проблем, які можуть виникнути під час реальної експлуатації. Це сприяє підвищенню загальної стабільності, безпеки та зручності

використання системи як для адміністраторів, так і для звичайних користувачів.

Етапи тестування адміністративної частини

Першим кроком є перевірка коректності процедури входу до панелі адміністратора (див. Табл. 3.9). Далі перевіряється правильність відображення всіх елементів інтерфейсу на сторінці адміністратора, зокрема: графічних компонентів; функціональних кнопок; взаємодії з елементами керування.

Особливу увагу ми приділяємо консистентності дизайну, розташуванню елементів, відповідності стандартам веб-розробки та принципам зручності користування. Результати тестування сторінки адміністратора відображено в табл. 3.10.

Таблиця 3.9 – 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

Таблиця 3.10 – 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

Наступний етап тестування - це додавання нової кімнати, щоб переконатися в коректності її виконання. Цей процес включає перевірку відображення нової кімнати як на веб-сторінці для користувачів, так і у відповідному розділі панелі адміністратора.

Результати тестування додавання кімнати наведено в табл. 3.11.

Таблиця 3.11 – 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 введення адміністратора,	Вся інформація кодується для відображається коректно
Ввести код адміністратора	Кімнату додано	Кімнату успішно додано

Після завершення тестування функції додавання нової кімнати, ми перейдемо до перевірки коректності роботи функції видалення кімнати. Ми хочемо переконатися в її правильному функціонуванні. Це включатиме перевірку відображення змін на сторінках адміністратора та користувача, а також перевірку оновлення списку доступних кімнат у реальному часі. Результати тестування функції видалення кімнати наведено в табл. 3.12.

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. Ми переконаємося, що система правильно обробляє запити на видалення та оновлює інтерфейс у реальному часі. Результати цього тесту відображено в табл. 3.13.

Таблиця 3.12 – 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

Таблиця 3.13 – 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 оренди кімнати	Дата в календарі звільнилась	Оренду успішно видалено

Результати тестування адміністративної частини веб-додатку підтверджують її стабільну, безпечну та зручну роботу. Усі перевірені функції повністю відповідають поставленим вимогам, що свідчить про високу якість реалізації цього компонента. Під час тестування помилок виявлено не було, що підтверджує готовність адміністративного модуля до використання в реальному середовищі та його відповідність експлуатаційним очікуванням.

ВИСНОВКИ

У межах цієї кваліфікаційної роботи бакалавра розробили веб-додаток для бронювання житла.

На початковому етапі ми провели аналіз ринку оренди житла, підкресливши зростаючу популярність онлайн-бронювань. Ми визначили ключові функціональні цілі та вимоги до майбутнього програмного забезпечення, а також порівняли його з існуючими рішеннями, такими як Airbnb та Booking.com, щоб виявити їхні сильні та слабкі сторони.

Ми також обґрунтували вибір Visual Studio Code як середовища розробки, проаналізувавши його можливості порівняно з IntelliJ IDEA та Atom.

Архітектура веб-додатку базується на трирівневій моделі: фронтенд, бекенд і сервер бази даних. Це забезпечує ефективність розробки та легке масштабування. Особливу увагу ми приділили створенню зручного й інтуїтивного інтерфейсу з адаптивним дизайном, що коректно відображається на різних типах пристроїв.

Процес розробки включав вибір та обґрунтування технологій, а також створення алгоритмів для реалізації ключових функцій: реєстрації, авторизації користувачів, додавання та бронювання кімнат. Використання HTML, SCSS, JavaScript і платформи Firebase дозволило створити функціональний, безпечний та надійний веб-застосунок, орієнтований на потреби кінцевих користувачів.

Тестування охоплювало перевірку коректності роботи інтерфейсу, функціональності авторизації та реєстрації, процесів оренди та адміністративної панелі. Це дозволило нам виявити та усунути потенційні недоліки ще на етапі розробки, що позитивно вплинуло на якість кінцевого продукту.

У результаті комплексного підходу до проєктування, реалізації та тестування, ми створили сучасний, безпечний і зручний у використанні веб-додаток для бронювання житла. Цей додаток відповідає актуальним вимогам ринку та має потенціал для подальшого розвитку й комерційного впровадження.

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://forsethgvild.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/EBUP_T200673.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.
1. [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 рсц: <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. Катільніков, О. П. Косовець [Електронний ресурс] – Режим доступу до ресурсу: <https://ir.lib.vntu.edu.ua/bitstream/handle>
20. JavaScript Підручник. Основи вебпрограмування [Електронний ресурс] – Режим доступу до ресурсу: <https://w3schoolsua.github.io/js/index.html>