



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота

Інформаційна система управління часовим розподілом ресурсів (за зразком Airbnb)

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б.	Група	Роль
1	Мохсені Софіані Микита Яхйяйович	КН-П-212	Розробник (backend)
2	Ковальов Данило Ярославович	КН-П-212	Розробник (frontend)
3	Шарпіло Дмитро Андрійович	КН-П-212	Розробник (backend)
4	Попов Петро Петрович	КН-П-212	Розробник (frontend)

Автор звіту: Ковальов Данило Ярославович

Ментор	Доцент кафедри ІТ та ФП, к.т.н., доцент Краковецький О.Ю.
--------	--

АНОТАЦІЯ

Робота була присвячена розробці функціонального веб-додатку з онлайн-бронюванням житла. Ця система спирається на вже існуючу систему [airbnb.com](https://www.airbnb.com), з покращеним інтерфейсом та функціональними можливостями.

Проект із серверної сторони була реалізована на ASP.NET Core 8.0 з використанням архітектури N-Tier Architecture, API-Centric Architecture, Domain-Driven Design, що дозволяє підтримувати модульність та незалежність функціональних частин проекту. Для роботи з даними були використані Microsoft SQL Server у сервісі Google Cloud SQL, EF Core та API, які реалізовані форматом RESTful API. Сервер для постійної роботи було опубліковано через сервіс Microsoft Azure, що дає змогу отримувати, відправляти дані, а також їх аналізувати. Також було виділено увагу безпеки проекту: авторизація та автентифікація зроблено на базі JWT Token, а доступ до адміністративної функціональності було обмежено через ASP.NET Identity та ролі. Для безпечного зберігання пароля користувача шифруються у базі даних за допомогою BCrypt.

З боку клієнтської частини проект розроблений з використанням бібліотеки React. Архітектура клієнтської частини побудована за feature-based принципом, при якому кожен функціональний модуль, такий як: авторизація, фільтрація, картки житла були ізольовані у власну директорію з компонентами, логікою стану, стилями та API-запитами. Для маршрутизації всередині програми використовується React Router DOM, що дозволяє реалізувати навігацію без перезавантаження сторінки. Для роботи зі станами у проект була інтегрована бібліотека Redux, дана бібліотека дозволяє позбавитися передачі станів між компонентами через “props”, а зберігати їх глобально і працювати з ними в будь-яких компонентах. Для роботи з отриманими даними, кешем та такими станами як (isLoading, isPending, isError, onSuccess) була інтегрована бібліотека React Query, яка також

дозволила абстрагуватися від локальних станів у компонентах. Для надсилання запитів на серверну частину проекту було додано бібліотеку Axios. Axios забезпечив більш гнучку та зручну роботу з HTTP запитамі порівняно з fetch. З використанням Axios було реалізовано функції взаємодії з Rest Api.

Результат проекту є масштабований веб-додаток із високим рівнем безпеки, сучасною архітектурою та гарною ефективністю, з можливістю подальшого розвитку

ЗМІСТ

ВСТУП_____	5
Технічне завдання до проекту_____	6
РОЗДІЛ 1. Загальна характеристика системи_____	11
РОЗДІЛ 2. Реалізація клієнтська частини_____	13
РОЗДІЛ 3. ОЦІНКА_____	20
ВИСНОВКИ_____	24
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ДЛЯ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ_____	26

ВСТУП

На сьогоднішній день безліч речей, які потрібні для повсякденного життя, можна зробити все через пристрій з доступом в Інтернет, речі такі як: замовити їжу, знайти одяг, купити квиток в кіно і так далі. Інтернет дуже швидко розвивається та впроваджується в наші життя. І бронювання послуг є одним із цих завдань. На даний момент систем бронювання домінують кілька компаній та розробників програмного забезпечення, і є багато випадків, коли вони можуть бути корисними. Систему бронювання можна знайти скрізь: у кінотеатрах, лікарнях, салонах краси, ресторанах і так далі. І такі системи можуть бути складними, оскільки є багато різних варіацій структур та користувачів, що призводить до ускладнення розробки ресурсу.

Якщо система бронювання буде дуже гнучкою, то є високий шанс того, що вона може бути складною для користувача та адміністрації. Результатом може бути: непотрібні натискання з боку адміну, і при цьому включаються функції, в яких немає потреб. Системи для бронювання повинні бути простими та логічними у використанні одночасно для користувача та ринку.

У документі розглядаються проблеми, з якими ми зіткнулися під час розробки системи, а також описується повний цикл розробки цього проекту.

Звіт складається з трьох розділів, у яких описуються загальні характеристики системи, деталі створення клієнтської частини, а також результати її розгортання та тестування. Висновки відображають результати, досягнуті у цій частині проекту, загальний висновок - сукупність звітів усіх учасників проекту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

СИСТЕМА УПРАВЛІННЯ РЕЗЕРВУВАННЯМ РЕСУРСІВ (за прикладом сервісу Airbnb.com)

Загальний опис.

Програмне забезпечення (ПЗ) призначене для створення розподіленої системи керування ресурсами, яке передбачає можливість проактивного управління, зокрема, резервування. Функції системи мають забезпечити повний життєвий цикл управління ресурсами: створення, актуалізація, використання та видалення (вживання). Головна особливість системи полягає у тому, що пріоритет використання обмежених ресурсів знаходиться у залежності від терміну його резервування.

Призначення:

ПЗ орієнтується на створення контейнеру управління обмеженими ресурсами на основі аналізу їх резервування. Оброблення конкурентних запитів на обмежені ресурси має враховувати показники пріоритетності, одним з яких є часовий показник моменту резервування. ПЗ створюється за прикладом сервісу бронювання обмеженої кількості місць, у той же час має складатись за принципом універсальності, мати змогу змінити цільове призначення.

Вимоги до функціоналу. Обов'язкова частина

Головна сторінка

Стартовий інтерфейс ПЗ (головна сторінка або активність) має за мету популяризацію системи. На ньому має бути забезпечено відображення помешкань, з можливістю вибирати категорії. Також головна сторінка має містити засоби авторизації та реєстрації або посилання на них.

Реєстрація користувачів:

Користувач ПЗ має змогу зареєструватись за допомогою авторизаційних даних (логіну/паролю).

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - електронна пошта. Засіб оперативного контактування при реєстрації підлягає верифікації через введення коду, пересланого через повідомлення довільного типу. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

Також передбачається наявність у системі користувачів з додатковими правами: адміністраторів. Реєстрація кореневого адміністратора системи має бути реалізоване при інсталяції системи, наступним користувачам змінюють права наявні адміністратори.

системи, наступним користувачам змінюють права наявні адміністратори.

Авторизація та особистий кабінет користувача:

За фактом успішної авторизації користувачу надається доступ до особистого кабінету. Це окремий інформаційний ресурс (сторінка), що являє собою основне робоче місце користувача ПЗ. Повинен забезпечити усі необхідні засоби для:

- Пошук ресурсів за метаданими, групування та фільтрування результатів за різними ознаками, наявними у знайденому контенті. Для об'ємних результатів має бути забезпечена пагінація
- Створення замовлень на вільні ресурси
- Управління власними замовленнями: перегляд, пошук, підтвердження, відмова

Також засобами особистого кабінету має бути передбачено можливість зміни персональних даних.

Зворотній зв'язок

Користувач має змогу встановити власну оцінку для спожитого ресурсу. Також є можливість залишити коментар щоб інші користувачі могли більше дізнатись про ресурс.

Архітектура

ПЗ має бути спроектоване за розподіленою клієнт-серверною архітектурою з чітко відокремленими розподіленими вузлами системи:

Москит - візуальна модель інтерфейсу користувача із зазначенням зв'язків і переходів між різними частинами інтерфейсу.

Backend - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.

Web App - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій.

Усі модулі ПЗ (окрім візуальної моделі Москит) повинні обов'язково включати до свого складу засоби

- розгортання на новому пристрої (інсталяції) із встановленням початкових таблиць у БД та кореневого користувача-адміністратор

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контролю версій (вибір конкретної системи узгоджується групою проекту).

Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- HomeFU
 - home-fu-frontend-public
 - home-fu-backend-public

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Backend - .NET технології із хмарним розміщенням (Azure), Google Cloud SQL

Web – JS, React, TS, HTML, Sass

Mockup - визначається ментором з дизайну

Команда проекту

Для виконання поставлених завдань передбачається робота команди розробників окремо за кожним з модулів проекту. Мінімальна команда проекту - два учасники, у такому складі один учасник виконує модуль, другий - програмну частину Web та розміткову частину Web.

Рекомендована команда проекту - чотири учасники, у такій команді два учасника відповідають за модуль Backend, інші два - програмну частину Web та розміткову частину Web.

Дозволяється включення до команди більшої кількості учасників.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проекту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носіїві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU <http://www.gnu.org/prep/standards/>, у тому числі з додержанням вимог до коментування та документування коду.

Автореферат містить описову частину, у якій розкриваються основні концепти проекту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Архітектура платформи складається з таких частин:

«Frontend».

«Backend»,

«FRONTEND»

З основних аспектів веб-архітектури має бути єдиний стиль та інтерфейсна основа. Функціональна частина має бути реалізована з урахуванням єдиного стилю.

Проект повинен бути інтуїтивно зрозумілий і він повинен відповідати сучасним стандартам для користувацького використання.

Наш проект має безліч сторінок для реалізації великого контенту, а також можливості додавання нових фіч, які дозволять поліпшити проект. Односторінковий проект, на даний момент є не актуальним і не має жодного сенсу, тому що додавання яких-небудь фіч, будуть обиратися в серверну частину, навантаженням на одній сторінці а також не можливості фізично додати будь-яку додаткову функціональність.

1. Проект повинен використовувати однакові елементи взаємодії з користувачем.
2. Розташування кожного елемента взаємодії повинні бути однакові
3. Розміри елементів повинні дотримуватися загального шаблону
4. Всі елементи на різних сторінках повинні дотримуватися загального стандарту, загального стилю.
5. Дизайн елементів має бути врахований та реалізований на початковій стадії розробки

Розробник Frontend повинен скласти грамотний план дій та реалізації, для виконання всіх умов і успішного реалізацію проекту. Від грамотного підходу розробника залежить реалізація самого проекту та його правильних підхід. У разі відхилення від норм або заздалегідь обговореного регламенту, розробка може бути більш тимчасово витратною або проявлятися конфлікти

«BACKEND»

Протягом життєвого циклу системи у «backend» зберігаються і обробляються дані, створенні та редагуванні у всій системі. Компоненти цього компонента включають проектну базу даних та API (Application Programming Interface), які дозволяють отримувати доступ до основних функцій системи через клієнську частину. Незамінна частина «backend» створена на мові програмування C# і Microsoft SQL Server через сервіс Google Cloud SQL у якості бази даних.

Онлайн-платформа має пропрієтарне ліцензію програмного забезпечення. Через це тільки правовласник має доступ до вихідного коду програми. Програмна частина платформи зберігається у приватному хмарному репозиторії веб-сервісу GitHub (через дотримання правил комерційної ліцензії) за посиланням: [GitHub](#). Доступ до готової користувальницької частини онлайн-платформи знаходиться за адресою: [посилання](#).

РОЗДІЛ 2

РЕАЛІЗАЦІЯ FRONTEND

Для виконання реалізації проекту, потрібні деякі аспекти а саме:

1. Вимоги щодо самої реалізації проекту
2. Інструменти, що використовувалися в реалізації проекту
3. Визначення візуалізації проекту
4. Реалізація коду
5. Використання бібліотек виконання роботи.
6. Реалізація мапи

"Вимоги щодо реалізації проекту"

Реалізація сайту, вона може постійно змінюватись. Тому нам потрібно було визначити початкові моменти з реалізації а саме:

Дизайн індістичний і точний, жодних серйозних змін не повинно відбуватися. У разі серйозних змін, була потрібна можлива переробка логіки самого проекту, що вкрай не допустимо.

Сайт повинен бути адаптований під різні гаджети, такі як телефон, планшет, ноутбук, персональний комп'ютер. Є таке визначення як "Mobile First" роблячи "адаптивну" для мобільних пристроїв в першу чергу. Мобільних користувачів набагато більше, ніж користувачів з ноутбуків або персонального комп'ютера.

Точне розуміння, які будуть використані мови програмування для мінімізації конфлікту в коді, а також грамотної, конструювання комунікації з колегами з розробки. Авжеж заздалегідь обговоривши всі позитивні та негативні знання кожного розробника для ефективного, збалансованої роботи.

Комунікація. Дуже велику роль відіграє всередині команди а саме особиста думка щодо реалізації проекту, якісь ініціативи аспекти а також можлива

допомога в уточненні реалізації того чи іншого моменту.

Розподіл точних завдань і розуміння , що кожен учасник виконує , реалізації цього компонента , використовувався Trello.

Trello сильний помічник у грамотному рості завдань, їх пріоритет, розуміння з якою посиленістю потрібно вирішити те чи інше питання, а також з яким підходом потрібно реалізувати

"Інструменти, що використовувалися в реалізації проекту"

Як інструменти одноголосно було обрано використовувати: React, TypeScript, HTML і Sass, JavaScript.

React - це JS Бібліотека (JavaScript), що дозволяє швидко і легко реалізувати більшість завдань, які поставлені командою. Надає можливість використання інструментів, для розробки інтерфейсів користувача.

HTML, класична система верстки, допомагає реалізувати веб-сторінку.

HTML є однією з основних мов розмітки.

У React для опису розмітки використовувався JSX - Розширений синтаксис JavaScript, який дозволяє писати HTML розмітку, прямо всередині JavaScript.

TypeScript - це розширена версія JavaScript з підтримкою статичної типізації та інших можливостей.

Sass - Допомагає реалізувати візуалізацію проекту. Має безліч команд по правильному розташуванню кнопок, стовпців, форумного вікна або будь-якого іншого аспекту на нашому проекті.

"Визначення візуалізації проекту"

Візуалізація проекту насамперед має привертати погляд людини. Цей проект повинен "радувати око". Своєю грамотною кольоровою палітрою, розташування іконок, кнопок, фотографій, грамотна реалізація "hamburger menu". У разі гарної візуалізації користувач може не звернути увагу на мінімальні недоробки, на які могли звернути увагу. Авжеж визначення "красиво" дуже суб'єктивне і по-різному оцінюється. Однак ми підібрали з командою візуал, що максимально приємно виглядає.

"Реалізація коду"

1. Відображення модальних вікон на різних сторінках сайту.

Модальне вікно було реалізовано для відображення додаткової інформації або відображення різних форм або фільтрів. На прикладі було реалізовано модальне вікно прикладом нижче :

```
export const GuestBlock = () => {  
  const counterValue = useSelector((state: RootState) => state.counters.counter)  
  const dispatch = useDispatch()  
}
```

Для відображення модальних вікон використовувалася бібліотека Redux, яка зберігає стан відкриття або закриття.

```

</div>
<div className={style.itemBlock}>
  <div className={style.itemDescription}>
    <h5 className={style.title}>Домашні тварини</h5>
    <span className={style.travelWithAnimal} onClick={() => dispatch(openAnimalsModal())}>
      Подорожуєте із твариною-помічником?
    </span>
  </div>
</div>

```

ome-fu > src > redux > TravelFilter > GuestSlices > TS animalsSlice.ts > [⌘] openAnimalsModal

```

1  import { createSlice } from "@reduxjs/toolkit"
2
3  interface AnimalsModalState {
4    isOpen: boolean;
5  }
6
7  const initialState: AnimalsModalState = {
8    isOpen: false,
9  }
10
11 export const animalsSlice = createSlice({
12   name: "animalsModal",
13   initialState,
14   reducers: {
15     openAnimalsModal: (state) => {
16       state.isOpen = true;
17     },
18     closeAnimalsModal: (state) => {
19       state.isOpen = false;
20     },
21     toggleAnimalsModal: (state) => {
22       state.isOpen = !state.isOpen;
23     },
24   },
25 })
26
27 export const {
28   openAnimalsModal,
29   closeAnimalsModal,
30   toggleAnimalsModal
31 } = animalsSlice.actions;
32
33 export default animalsSlice.reducer;

```

“Пошук”

Потрібно було реалізувати пошук, пошукова система є невід'ємною частиною. У ході розробки потрібно реалізувати цей аспект якісно та стилізовано. При натисканні на полі потрібно вести назву готелю або будь-які слова, що входять в назву готелю. Після натискання Enter або праворуч на значок пошуку , відбувається пошук нашого готелю. Протягом двох секунд наш готель знаходиться і ми спокійно можемо на нього зайти і переглянути всі аспекти, що нас цікавлять.

```
export const Search = ({ onSearch }: Props) => {  
  const [searchQuery, setSearchQuery] = useState("");  
  const [isLoading, setIsLoading] = useState(false);  
  
  const dispatch = useDispatch();  
  
  const selectedDateArival = useSelector(  
    (state: RootState) => state.arrival.selectedDate  
  );  
  const selectedDateDeparture = useSelector(  
    (state: RootState) => state.departure.selectedDate  
  );  
};
```

Пошук...



“Баги”

У ході тестування веб-програми були виявлені деякі "баги" різного характеру, які були потрібні в змінах, для коректної роботи програми. Ці баги відбувалися у сфері адаптивної частини веб-сторінки. Також у випадку не уважності були баги в розташуваннях кнопок або інших форм у зв'язку з конфліктом в основному scss файлі.

"Використання бібліотек для виконання роботи."

Використовувалося бібліотеки: Axios, Redux, React Select, React Swiper, React Query, React Form, React Route.

Використання бібліотек потрібно для прискорення розробки, спрощення роботи з інтерфейсом і забезпечення простої архітектури проекту. Бібліотеки були обговорені з командою , для ефективного використання а також не засмічувати зайвими бібліотеками .

"Реалізація Google map."

Потрібно реалізувати google map. Вирішено підійти до цього питання кардинально і використовувати google cloud, де був наданий apі-ключ підключення, він знаходиться в env.local де завдяки йому, реалізація карти, стала можливою. . Вирішили використовувати env.local для мінімізації втрати конфіденційної інформації.

```
const { isLoading } = useJsApiLoader({  
  id: "google-map-script",  
  googleMapsApiKey: import.meta.env.VITE_GOOGLE_MAPS_API_KEY!,  
  language: "uk",  
});
```

```
lat: data.latitude,  
lng: data.longitude,  
});
```

Завдяки latitude (широта), longitude (довгота), можна було визначити точне місцезнаходження. Frontend отримує координати від Backend і, відповідно, маркер позначає точну позицію знаходження даного готелю або житла.

```
mapContainerClassName={styles.map}  
center={{ lat: data.latitude, lng: data.longitude }}  
zoom={12}  
options={{  
  minZoom: 1,  
  maxZoom: 12,  
}}  
onLoad={onLoad}
```

Також були заздалегідь задані максимально і мінімальне значення для зуму, оскільки автоматично без уточнення даних аспектів карта дуже близько зумило і доводилося довго матати для розгляду самої локації, розташування житла.

РОЗДІЛ 3

ОЦІНКА

У ході розробки , ми враховували аспекти реалізації архітектури проекту і різні потреби для самого веб-сайту. Для оцінювання яких потреб , потрібні для нашого споживача , це непогане визначення для розставляння планів на майбутнє.

Авжеж в ході описаних аспектів у моїй Frontend частині, вже можна визначити оцінювання архітектури та оцінювання самої системи, але потрібно дати точні визначення за кількома деталями.

Основи :

Наприклад, командою було вирішено взяти Airbnb, який дозволяє нам грамотно розподілити реалізацію проекту. Також ми розуміли, як виглядає сам проект, його дизайн, візуалізацію має бути схожою і практично ідеально виглядати. Але в ході виконання нашої роботи , за невеликий період часу до захисту , Airbnb був перероблений розробником колосально і на думку команди досить погано. На даний час, це є плюсом так, як наш проект є унікальним і більш стилізованим.

Також командою було вирішено розподіляти завдання по "Trello", де були уточнені аспекти реалізації, класів стилів, функціональність, безліч сторінок на сайті, для більш цікавої візуалізації.

Завдяки нашій архітектурі та готовності , було підготовлено весь аспект реалізації кожного компонента самостійно. Кожен розробник, не був особливо підв'язаний під кожного розробника, крім мінімальних моментів, таких як, підключення хоста, прийняття API.

Авжеж розробники усвідомлюють про можливі моменти з реалізації

проекту і є можливість , зробити легше деталізацію доступу , також на даний момент , користувач може мати обмеження до певних компонентів.

На даний момент усі наші вимоги до проекту, реалізації, виконані. Команда реалізувала кожен аспект , який спочатку хотіла реалізувати по архітектурі самого проекту і стилізації. Навіть був додатковий час для реалізації певних нововведень, які для користувача вважатимуться зручним.

Масштабованість :

Також не малий аспект по реалізації це вміння проекту швидко і якісно розширювати свої можливості, адаптуватися при практично будь-яких змінах з мінімальною втратою часу розробника. Ідеальна і не мало важливий момент для реалізації даного проекту. Можливість без значних змін, махінацій нашого проекту, адаптуватися під серйозні навантаження.

З вагомих аспектів, коректна архітектура проекту має серйозний потенціал, завдяки стабільній архітектурі , реалізації будь-якого аспекту, нових новознаходок , чи збільшення навантаження не ставлять проблем .

Компоненти повинні бути реалізовані ідеально і не залежати один від одного. При додаванні любого компонента , не повинно , щось зламатися або змінюватися , додавати баги або будь-які інші незручності. Все має бути чітко організована, з послідовністю кроків.

З основних проблем , які можуть статися , це виявлені помилки в коді , на момент розробки. В основному це відбувається через неувважність розробника і не грамотного підходу до реалізації.

Основний аспект це грамотна позиція команди, швидко, якісно реалізація проекту, наші потреби для візуалізації проекту, також вміння грамотно підходити до проекту для мінімізації проблем з кодом, багом або ігнорування якихось недоліків.

Стандартизований користувацький інтерфейс :

Однією з основних вимог команди є дотримання загального дизайну, який повинен бути однаковий на кожній сторінці. Тобто має бути чіткий набір, наших макетів, стилів, розташування кнопок. Все це має бути затверджено Team Leader і відповідно до його керівництва виконувати вимоги .

Другим не мало важливим аспектом, є розставлення пошукових систем , фільтром , інших кнопок , так само , як і на інших сторінках сайту. Інші сторінки, користувач очима шукає дані параметри там, де він і бачив востаннє, це буде вкрай не допустимо і не логічно, якщо команда вирішить поміняти на інших сторінках , розташування таких вагомих кнопок.

Третім пунктом адаптивність повинна бути однаковою і строгою. Мається на увазі також тут про раніше згадану "Масштабованість". Так як адаптивна частина проекту повинна бути практично скрізь, вона допомагає користувачу в незалежно від місця розташування ознайомитися з матеріалами сайту. Також адаптивність повинна дотримуватися норм адаптивної частини, не виходити за межі, щоб користувач не міг сенсором перегортати в важкодоступні частини сайту, які не передбачені для мобільної частини. Масштабованість , також є дуже важливим моментом , оскільки у разі реалізації цієї адаптивної частини , може поламатися саме розташування матеріалів на комп'ютері , що доведеться витратити чимало часу на відновлення та обліку mobile first.

Останній не мало важливий аспект у реалізації це дизайн. Як би дивно не звучало. У нашій команді відсутнє дизайнер але присутній сам "дизайн". Це мається на увазі, що завдяки раніше обговореному дизайну, наш проект не хаотичний і має структуру і розуміння, приблизного розташування візуалізації нашого проекту, це дуже допоможе команді в реалізації проекту а також скоординованої, грамотної підготовки

Напрями для подальшого розвитку застосунку:

Одним з можливих проблем у майбутньому є обробка даних, базою даних. Клієнт може відкрити не один вклад нашого проекту, через що можливе навантаження на наш проект. Для цього потрібно більш серйозний сервер.

Також у зв'язку з відсутністю самого дизайнера, ми усвідомлюємо, що визначення "красиво" дуже суб'єктивне. Так як дизайнер це спеціалізований співробітник, у великому відсотковому співвідношенні, даний співробітник зробить дизайн більш, лаконічніше, красивіше і менш навантаженим. На жаль, дизайнер у нашій команді відсутній але ми вважаємо наш проект, реалізований красиво.

З майбутніх помилок також можливо проблема з адміністративною панеллю. Оскільки, вивчення і розуміння принципу роботи даної системи, вимагають часу і можливі проблеми із самим токіном, оскільки, може разлогинить користувача. Можливо, потрібно збільшити проміжок часу до розлогіна користувача.

Ми сміємо припустити, про те, що в нашому майбутньому розвитку проекту, потрібна локалізація іноземними мовами. Оскільки завдання є Airbnb, де на даному сайті будь-яка людина з будь-якої держави, може ознайомитися з цим сайтом. На даний момент, тільки наші громадяни України можуть сприймати інформацію на 100% і відповідно ним користуватися. Звичайно цей недолік, потрібно врегулювати з доповненням іноземних мов. Особливо міжнародного, яким є англійською.

ВИСНОВКИ

Наш проект реалізований досить добре . Наша команда, включаючи мене, зіткнулася з великими труднощами і змогла їх подолати. Адже труднощі, які ми отримали, роблять нас розумнішими і сильнішими. Завдяки цій роботі, вимоги поставлені нашим керівником, відіграли велику роль у реалізації всього нашого проекту. Всі аспекти, які реалізовані в нашому проекті, описані в пункті "Оцінка". Авжеж , нашій команді завжди є куди рости і ми завжди будемо йти вперед, тому що даний проект навчив багатьом аспектам від розробки в команді до постійного вдосконалення над собою та вивчення нових матеріалів, документацій.

Також ми вважаємо, що в нашому проекті не потрібно колосальних змін або ж критичної нехватки, будь-яких функцій. Всі поставлені завдання нашою командою були реалізовані. Звичайно в чомусь довелося трохи поступитися або скасувати, але наприкінці нашої роботи, всі аспекти були реалізовані і задоволені.

Не малий аспект це команда, це комунікація. Дуже серйозна та важлива частина роботи. Комунікація – це спосіб донести думку розробнику на всі 100%. Щоб тебе зрозуміли та реалізували твої потреби у повному обсязі. Також це командний дух, ми підтримували один одного. Були злети та падіння. Були помилки і швидкі перемоги, але ми розуміли, що ми команда і разом ми зможемо довести, що весь вивчений матеріал, всі наші старання та вивчення програмування, дають свої плоди.

Критично важливо звернути увагу на дизайн. Ми зрозуміли, що в першу чергу потрібно реалізувати повноцінний дизайн та розуміння його реалізації. Так як, ми могли придумати багато різних фіч або нововведень, але воно могло все впертися в дизайн. Тому ми спочатку дотримувалися в теоретичній частині, розуміння як воно виглядатиме.

Наш проект досить структурований і докладно зроблений. Строго і

послідовно. Кожна папка підписана коректно і одразу є розуміння, що знаходиться в цій папці і за що відповідає ця папка. Це спрощує знаходження потрібних файлів для написання коду або його рефакторингу.

Наша команда вважає, що завдання виконані і ми виконували їх добре. Звичайно є куди рости і ми готові продовжувати нашу спільну справу. Ми сподіваємося, що наш проект у майбутньому допоможе людям у знаходженні житла. Ми працюватимемо над ним і далі, не покладаючи рук. Щоб обігнати Airbnb. Наша команда, вдячна за такий цінний досвід, отриманий у ході реалізації цієї роботи.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ
КЛІЄНТСЬКА ЧАСТИНИ

№ п.п.	Ресурс
1.	React; Розробник: Facebook; Документація: посилання .
2.	Visual Studio Code; Розробник: Microsoft; Документація: посилання .
3.	JavaScript; Розробник: Brendan Eich of Netscape; Документація: посилання .
4.	TypeScript; Розробник: Microsoft; Документація: посилання .

5.	HTML; Розробник: WHATWG; Документація: посилання .
7.	SASS; Розробник: Hampton Catlin, Natalie Weizenbaum; Документація: посилання .
8.	Axios; Розробник: Matt Zabriskie та спільнота; Документація: посилання .
9.	Redux; Розробник: Dan Abramov, Andrew Clark Документація: посилання .
10.	React Query; Розробник: Tanner Linsley Документація: посилання .
11.	Google Cloud; Розробник: Thomas Kurian Документація: посилання .