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

випускна кваліфікаційна робота бакалавра

«СИСТЕМА УПРАВЛІННЯ РЕЗЕРВУВАННЯМ РЕСУРСІВ (ЗА ПРИКЛАДОМ СЕРВІСУ AIRBNB) (FRONTEND)»

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

Виконавці проекту

№	П.І.Б.	Група
1	Карпов Д.Р.	КН-П-201
2	Лупашко П.П.	КН-П-201
3	Козлов Я.Я.	КН-Д-202
4	Воробьев А.І.	КН-Д-202
5	Гольденберг В.О	КН-А-201

Автор звіту

_____ Карпов Данило Романович

Одеса – 2024

АНОТАЦІЯ

Проект був започаткований для створення нової системи онлайн-бронювання, яка базується на існуючій платформі airbnb.com. Функціональні можливості та інтерфейс користувача були модернізовані. Розробка нової системи розпочалася з урахуванням вимог керівників департаментів та основної концепції поточної системи airbnb.com.

Ця робота описує дослідження, розробку та тестування системи бронювання, що має на меті полегшити процес купівлі-продажу послуг між користувачами та компаніями. Система пропонує широкий спектр функцій, зокрема:

- **Реєстрація та авторизація:** Користувачі та компанії можуть легко створити облікові записи та отримати доступ до системи.
- **Аутентифікація:** Надійні механізми аутентифікації гарантують безпеку користувачів та захист їхніх даних.
- **Оренда:** Користувачі можуть шукати та бронювати послуги, пропоновані компаніями, на платформі.
- **Додавання та управління готелями:** Компанії можуть додавати інформацію про свої готелі, керувати ними та приймати онлайн-бронювання.
- **Доступ до інформації:** Користувачі та компанії мають доступ до інформації про свої бронювання, клієнтів та зароблені кошти.

Головними завданнями були покращення зручності використання інтерфейсу системи. У цій роботі об'єктом дослідження є алгоритм, що забезпечує комплексну оцінку результатів за численними критеріями в умовах невизначеності початкових даних. Предмет дослідження включає фактор,

який дозволяє користувачеві легко подорожувати та комбінувати об'єкти і рейси найпростішим можливим способом.

Цілями дослідження є:

- Огляд систем бронювання, що дозволяють користувачам взаємодіяти з іншими користувачами та компаніями без необхідності проходити складні процедури.
- Розробка програмного забезпечення з урахуванням досягнутих раніше результатів, а також проведення його тестування в різних умовах і на різних пристроях.
- Впровадження програмного забезпечення для захисту даних у систему з метою підвищення рівня інформаційної безпеки створеної системи.
- Методи дослідження, що застосовуються для досягнення цілей, базуються на комбінації методів системного аналізу, моделювання та когнітивної методології.

Для розробки клієнтської частини було використано веб-фреймворк Angular разом з HTML, TypeScript і SCSS. Ця дослідницька робота зосереджена на створенні системи бронювання з підтримкою специфічного пошукового фільтру.

ЗМІСТ

ВСТУП

Все більше людей використовують комп'ютери та мобільні пристрої в своєму повсякденному житті завдяки швидкому зростанню Інтернету за останні роки. Бронювання послуг стало однією з таких завдань. На ринку систем бронювання домінують кілька розробників програмного забезпечення та компаній, і їхні послуги корисні в багатьох ситуаціях. Системи бронювання застосовуються повсюди: в ресторанах, лікарнях, перукарнях, школах та інших установах. Вони можуть бути дуже складними через різноманітність пристроїв та користувацьких уподобань, що ускладнює процес розробки.

Дуже гнучкі системи бронювання часто є надто складними для користувачів та адміністрації. Це призводить до зайвих кліків та активації непотрібних функцій. Системи бронювання повинні бути логічними і простими у використанні як для ринку, так і для користувача.

У рамках даного звіту розглядається клієнтська частина, а також описується весь цикл її розробки. Звіт складається з трьох розділів: загальні характеристики системи, деталі створення клієнтської частини, результати розгортання і тестування. Висновки відображають досягнуті результати в цій частині проекту, а загальний висновок є сукупністю звітів всіх 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, що забезпечує можливість проактивного управління, включаючи резервування. Функції системи покривають повний життєвий цикл управління ресурсами: створення, актуалізація, використання та видалення. Головна особливість системи полягає в тому, що пріоритет використання обмежених ресурсів залежить від терміну їх резервування.

Призначення:

Програмне забезпечення орієнтоване на створення контейнера для управління обмеженими ресурсами на основі аналізу їх резервування. Обробка конкурентних запитів на обмежені ресурси повинна враховувати показники пріоритетності, серед яких важливим є часовий показник моменту резервування. ПЗ розробляється за прикладом сервісу бронювання обмеженої кількості місць, але має бути універсальним і здатним змінювати своє цільове призначення.

Вимоги до функціоналу. Обов'язкова частина

Головна сторінка

На головній сторінці системи забезпечено відображення фільтрів, пошуку і каталогу апартаментів, що значно полегшує користувачам доступ до

необхідної інформації та сприяє їхній ефективній навігації по сервісу. Завдяки цим функціональним можливостям користувачі можуть швидко знаходити потрібні об'єкти для бронювання, використовуючи зручні фільтри і швидкий пошук.

Крім того, на головній сторінці знаходяться інструменти для авторизації та реєстрації нових користувачів або посилання на них, що робить процес взаємодії з системою ще більш зручним і простим. Це дозволяє новим користувачам швидко і без зайвих зусиль зареєструватися та отримати доступ до усіх функцій сервісу.

Реєстрація користувачів:

Користувач ПЗ має змогу зареєструватись як за допомогою авторизаційних даних (логіну/паролю), так і за допомогою облікових записів популярних соціальних мереж.

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон або електронна пошта. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

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 створення замовлень.

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

Процес бронювання:

Користувач повинен мати можливість перейти на сторінку будь-якого із запропонованих на головній сторінці готелів, отримати на ній ціну за ніч, опис, фото та всю іншу інформацію про готель. Користувач має бути забезпечений інтерфейсом для вибору дат і кількості гостей для бронювання на сторінці апартаментів. Після заповнення цих даних, має бути сторінка бронювання з необхідною інформацією для здійснення бронювання, а саме інформація про банківську картку, повідомлення компанії готелю, фото і номер телефону

Вимоги до функціоналу. Опціональна частина

Розміщення та управління

Компанія має можливість додати свій готель або апартаменти через спеціальну сторінку, де вона заповнює всю необхідну інформацію про об'єкт. Ця процедура не лише спрощує процес реєстрації нових об'єктів, але й забезпечує повну інтеграцію з системою управління, що дозволяє компанії ефективно керувати усіма аспектами своєї гостьової діяльності.

Крім цього, компанія має доступ до спеціальної сторінки хостингу, де вона може здійснювати моніторинг і управління своїми готелями і клієнтами. Ця функція не тільки сприяє оптимальному використанню ресурсів і підвищує

ефективність обслуговування гостей, але і забезпечує компанії зручний інтерфейс для взаємодії з її власною мережею готелів.

Архітектура

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

Mockup - 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

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок: ● root

- ○ Backend
- ○ Web
- ○ Mockup (вихідні файли або посилання на зовнішній проект) Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні

технології створення:

Web - Angular, TypeScript, SCSS, HTML

Mockup - визначається ментором з дизайну

Команда проєкту

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 Backend, другий - Frontend.

Рекомендована команда проекту - три учасники, у такій команді кожен з учасників відповідає за одну з частин: Mockup, Backend, Frontend.

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 носієві 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».

«Design»,

«Backend»,

«FRONTEND»

Імперативним аспектом веб-архітектури є те, що користувальницький досвід і інтерфейс залишаються послідовними протягом усієї платформи. Кожна функціональність повинна бути розроблена з урахуванням цього фундаментального принципу.

Платформа відображає односторінковий додаток, що відповідає максимальної зручності для користувача. Односторінковий додаток - це, по суті, те, що потрібно для того, щоб уникнути оновлення браузера при перезавантаженні, або переході на іншу сторінку.

Один із головних моментів - це те, що клієнтський інтерфейс повинен відповідати сучасності та зручності користування. Цей критерій вважається частиною вимоги узгодженості для програми веб-порталу:

1. Кожна частина платформи повинна використовувати однакові елементи взаємодії користувача.
2. Розташування кожного елемента взаємодії користувача має бути однаковим.
3. Розміри елементів інтерфейсу повинні бути однаковими для кожного випадку використання.

4. Усі елементи повинні відображатися у різних станах в одному стилі.

5. У будь-який час, коли це можливо, для реалізації всіх елементів користувача повинна бути використана заздалегідь визначена колірна схема.

Для виконання цих критеріїв необхідно, щоб розробник фронтенду зобов'язався і ретельно розумів принципи проектування. Однак документування обраних концепцій має вирішальне значення для допомоги адаптаційному процесу.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ FRONTEND

Відповідно до моделі «фази розробки», викладеної нижче, розробка системи була непростою. З часом ми покращили безпеку та функціональність, тому що це був перший масштабний онлайн-додаток, який ми розробили. Кожен великий етап розробки цього додатка був протестований.

1. Вимоги до збірки
2. Вибір інструменту розробки
3. Інтеграція сторінок користувача
4. Інтеграція функціональності входу

Вимоги до збору

Обов'язково потрібно пам'ятати, що деякі аспекти веб-сторінки можуть змінюватися протягом всієї розробки інтерфейсу. Ідея полягала в тому, щоб зібрати всі вимоги до початку розробки інтерфейсу. Незважаючи на те, що

наш головний відділ надав більшість вимог, ми обговорювали, чи є вони розумними, здійсненими і чи вписуються у часові рамки цієї розробки.

Перелік вимог розробки:

- Сторінки користувача повинні бути оптимізовані для використання комп'ютерами, планшетами і смартфонами;
- Користувацькі сторінки повинні бути у першу чергу розроблені у вигляді, створенному дизайнером.
- Сторінки користувача повинні бути пов'язані з базою даних, створеною розробником “backend”, щоб використовувати всю необхідну інформацію.

Вибір інструментів розробки

Створення веб-сайту з правильними інструментами має значний вплив на кількість часу та зусиль, необхідних для його створення. Це пов'язано зі складністю сайту, і необхідним рівнем налагодження. Визначення інструментів розробників має важливе значення для майбутнього розвитку. Використання широко відомих технологій може допомогти розробникам швидше вивчити нову систему.

Вибір зовнішніх інструментів розробки, що беруть участь з урахуванням цих факторів:

- Розмір проекту
- Вид проекту
- Впізнаваність інструментів

Фреймворк Material Angular використовувався для задання стилів користувальницького інтерфейсу. Material — це інтерфейсний фреймворк з відкритим вихідним кодом, вихідний код якого доступний за ліцензією MIT.

Він включає кнопки в стилі CSS, таблиці, форми та інші елементи. Material є одним з найпопулярніших фронтенд-фреймворком з великою документацією, що робить його придатним для використання у розробці системи.

Надаючи адаптивні функції та попередньо визначені стилі, Material спрощує та прискорює розробку інтерфейсу. Завжди можна змінити CSS класи або додати свої власні до загального пакету.

Я обрав SCSS замість звичайного CSS через його препроцесорні можливості, які включають змінні для легкого управління стилями, вкладені селектори для кращої структури коду, міксіни для швидкого і повторного використання стилів, а також підтримку імпорту файлів для організації проекту. SCSS сприяє покращенню підтримки і сумісності стилів у багатьох браузерах, що робить його вибором для ефективного розвитку складних веб-проєктів.

Реалізація сторінок користувача

Спочатку я створив головну веб-сторінку, яка є найважливішим фактором, оскільки це перше, що бачать користувачі, коли відвідують нашу платформу. Інтерфейс користувача та процес бронювання були розроблені, щоб бути максимально безперебійними та простими.

Варіанти наступні:

- Якщо користувачі хочуть шукати певні місця, вони можуть зробити це в полі пошуку.
- Якщо клієнти все ще не впевнені в тому, куди вони хочуть піти, ми рекомендуємо багато популярних готелів та місць.

Архітектурна основа складається з односторінкового додатка, який інкапсулює у собі дані у конкретних компонентах і посилається на основні односторінкові програми. У нашому проекті використовувався стандартний модуль маршрутизатора Angular, який дозволив нам створювати окремі маршрутизаційні зв'язки для кожного з компонентів. Це може бути використано для переходу між різними компонентами.

Після вирішення проблем маршрутизації та проблем побудови веб-сторінки, була реалізована база даних. Це дозволить завантажувати та зберігати інформацію, надану користувачами. Для цього використовувалися мережеві запити до API. Виклик API відбувається, коли клієнтський застосунок робить мережевий запит до нього. API бере запитувані дані та віддає їх клієнту.

Нижче наведено зразок зображення, яке показує клас StaysService успадковує функціональність BaseService та використовує HttpClient для здійснення HTTP-запитів. Він має методи для отримання детальної інформації про проживання за ID, отримання списку проживань користувача та отримання проживань з фільтрацією за вказаними критеріями за допомогою HttpParams.

```
export class StaysService extends BaseService {
  constructor(private http: HttpClient) {
    super();
  }

  public getStay(id: number): Observable<StayItemDetailed> {
    return this.http.get<StayItemDetailed>(environment.apiUrl + `/api/stays/${id}`);
  }

  public getMyStays(): Observable<StayItem[]> {
    return this.http.get<StayItem[]>(environment.apiUrl + '/api/stays/my', this.getOptions());
  }

  public getStays(filter: StayFilter): Observable<StayItem[]> {
    let params = new HttpParams();
    params = params.append('MinGuests', filter.minGuests.toString());
    params = params.append('MinBathrooms', filter.minBathrooms.toString());
    params = params.append('MinBedrooms', filter.minBedrooms.toString());
    params = params.append('MinBeds', filter.minBeds.toString());
    params = params.append('PlaceType', filter.roomType.toString());
    params = filter.location !== '' ? params.append('Location', filter.location) : params;

    return this.http.get<StayBrief[]>(environment.apiUrl + '/api/stays', { params } ).pipe(
      map((staysBrief: StayBrief[]) =>
        staysBrief.map((stay: StayBrief) => ({
          ...stay,
          duration: this.formatDuration(stay.startDate, stay.endDate),
        }))
      )
    );
  }
```

Як засіб відображення різних елементів на кожній із веб-сторінок, ми адаптували шаблон дизайну сторінок. Це було реалізовано за допомогою Material і численних користувацьких класів стилей CSS. З наведеного нижче прикладу можна побачити календар який реалізован за допомогую Material, можна зробити висновок скільки часу зекономив тільки цей елемент

```
<div class="calendar calendar-box">
  <mat-form-field class="calendar-box">
    <mat-label class="ligth">Check in - Check out</mat-label>
    <mat-date-range-input class="calendar-box" [formGroup]="range" [rangePicker]="picker">
      <input matStartDate formControlName="start" placeholder="Start date">
      <input matEndDate formControlName="end" placeholder="End date">
    </mat-date-range-input>

    <mat-datepicker-toggle class="calendar-box" matIconSuffix [for]="picker"></mat-datepicker-toggle>
    <mat-date-range-picker class="calendar-box" #picker></mat-date-range-picker>

    <mat-error *ngIf="range.controls.start.hasError('matStartDateInvalid')">Invalid start date</mat-error>
    <mat-error *ngIf="range.controls.end.hasError('matEndDateInvalid')">Invalid end date</mat-error>
  </mat-form-field>
</div>
```

Додавання функцій входу

Паролі перевіряються двічі для того, щоб запобігти помилкам введення тексту і забезпечити точність введених даних. Ми передаємо інформацію, яку користувач вводить у форму, до бази даних у її поточному стані, як тільки користувач подає форму. Кожне поле форми підлягає ретельній валідації, де система перевіряє, чи виконані всі вимоги до введення для кожного конкретного поля. Особлива увага приділяється електронній адресі, яка повинна бути унікальною для кожного користувача, що забезпечує уникнення дублювання записів і підвищує рівень безпеки даних.

Цей процес перевірки включає кілька етапів: спочатку система переконується, що всі обов'язкові поля заповнені, потім перевіряє правильність введених даних, таких як формат електронної адреси, і лише після цього приймає введені дані для обробки. Валідація полягає у перевірці дотримання

встановлених правил і стандартів для кожного типу даних, щоб гарантувати їх коректність і цілісність перед додаванням до бази даних. Таким чином, наш підхід не лише запобігає можливим помилкам і некоректним записам, але й забезпечує високу якість і надійність даних у системі.

РОЗДІЛ 3

ОЦІНКА

Потреби веб-сайту повинні бути детально розглянуті після ретельного проектування архітектури програми та розробки її основних компонентів. Оцінка потреб користувачів може слугувати основою для формулювання майбутніх цілей і напрямків розвитку. Важливо розуміти, що кожен етап розробки має враховувати вимоги користувачів, адже саме вони визначають кінцевий успіх проєкту.

У розділах «ЗАГАЛЬНІ ХАРАКТЕРИСТИКИ СИСТЕМИ» і «РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ» ми узагальнили початкові критерії розробки веб-додатка. В цих розділах ми детально розглядаємо вимоги як в контексті архітектури додатка, так і в контексті його реалізації. Архітектура додатка є основою, яка визначає структуру та взаємодію між компонентами, а реалізація інтерфейсу охоплює аспект користувацького досвіду, забезпечуючи зручність і функціональність.

Однак важливо пам'ятати, що навіть найретельніше планування не обов'язково гарантує успіх. Кожен етап розробки повинен включати постійну перевірку і коригування, щоб відповідати змінним вимогам користувачів і новим

технологічним викликам. Залучення користувачів до процесу оцінки і тестування може надати цінну інформацію для покращення функціональності і зручності використання веб-додатка, що в кінцевому рахунку сприятиме досягненню його успіху на ринку.

Основи

Спочатку платформа airbnb.com була взята за приклад для створення нашої платформи. Зокрема, дизайн і функціональність повинні були бути схожими на airbnb.com, з кількома покращеннями макета і виправленнями. Враховуючи це в контексті архітектури додатків, є багато рішень, які потрібно розглянути. Створення численних класів стилів, що реалізують бажану функціональність, може виконати цю вимогу. Функціональність односторінкового додатка є ідеальним рішенням для нашого випадку. Коли програма розроблена з обраним підходом, легше регулювати доступ, оскільки користувачам можна обмежити доступ до певних компонентів. Таким чином, сайт може бути адаптований до інформаційних вимог конкретного користувача та потреб бізнесу. Архітектура на основі компонентів дозволяє розробляти кожен компонент окремо, що значно полегшує процес розробки програмного забезпечення. Вимога може вважатися виконаною в поточному контексті, оскільки вона була реалізована під час впровадження. В результаті обраної архітектури один додаток можна розробити швидше, оскільки розміром проекту легше керувати.

Масштабованість

Вимога до розширення веб-додатків тісно пов'язана з початковими критеріями, оскільки інформація з "Основи" легко дозволяє масштабувати існуючий додаток. Від того, як побудований фундамент, залежить легкість масштабування. Додавання нових компонентів до додатка може ускладнити його роботу і викликати проблеми. У багатьох випадках ці проблеми виникають через залежності між класами або через невиявлені помилки під час розробки. Ці помилки можуть призвести до краху всієї системи у виробничому середовищі. Добре продумана архітектура спрощує масштабування і запобігає виникненню проблем. Компоненти веб-додатків є самостійними, тому додавання нових не повинно впливати на існуючі компоненти, за умови правильного управління взаємодією між ними. Сильна залежність між компонентами може викликати труднощі. Окремі компоненти веб-додатка також легко підтримувати завдяки створеній архітектурі. Уражену частину додатка можна виправити, якщо помилки виявлені у виробничому середовищі.

Коли вимога розглядається в контексті реалізації, її можна вважати виконаною. Додавання нових компонентів до веб-додатка досить просто завдяки функціональним можливостям, наданим основним фреймворком. Основний обсяг роботи, необхідної для забезпечення функціонування додатка як компонентного веб-додатка, виконується безпосередньо у самому компоненті. Кожен компонент вимагає певного налаштування, а точка входу додатка повинна бути визначена заздалегідь. Управління всіма основними конфігураціями може створити проблеми в майбутньому, якщо кількість

компонентів значно зросте.

Узгоджений інтерфейс користувача та досвід

Вимога узгодженості інтерфейсу користувача і користувацького досвіду у веб-додатку була розбита на кілька розділів. Розглянуті компоненти стосуються таких проблем, як єдність окремих елементів інтерфейсу користувача та більш широкого дизайну інтерфейсу користувача.

Перший компонент критеріїв показує, що повинен бути обмежений набір загальних стилів макета, що всі види по всьому веб-додатку повинні дотримуватися зазначених правил. Загальний дизайн доступних на даний момент додатків дотримується однакових правил.

Елементи інтерфейсу користувача, що використовуються для взаємодії з користувачем, повинні бути уніфіковані для кожного випадку використання відповідно до другої частини вимоги. Це означає, що на кожній сторінці всі текстові поля і кнопка, що використовуються для додавання нового елемента, повинні виглядати однаково. Розроблений дизайн просто допомагає розробникам правильно впроваджувати елементи інтерфейсу користувача, але вимагає відданості дизайнера в команді.

У третьому розділі критеріїв зазначено, що позиціонування елементів інтерфейсу користувача має бути узгодженим у всіх перспективах. Наприклад, якщо елемент рядка пошуку присутній поверх таблиці даних, він повинен бути присутнім у всіх містах. Коли розташування елементів довільно змінюються в різних містах, користувачеві стає складніше ефективно керувати

додатком. Знову ж таки, система дизайну може допомогти розробникам правильно реалізувати користувальницький інтерфейс, але їх створення вимагає зобов'язань дизайнера і знання.

Четвертий розділ вимоги стосується питання масштабування елементів інтерфейсу користувача. Знову ж таки, розроблена система дизайну допомагає розробникам, пропонуючи класи корисних стилів, які можуть бути використані для опису розміру елемента. Цю умову можна вважати частково виконаною, з урахуванням поточних компонентів, а також самого веб-додатка. Ця проблема повинна бути вирішена в майбутньому, як і розширення нинішньої системи проектування з новими загальними компонентами для кращого задоволення цілей онлайн-програми.

Можливі майбутні вдосконалення сайту

Незважаючи на те, що кілька проблем були виявлені і згодом вирішені під час роботи над цим продуктом, є ще кілька, які повинні бути вирішені в майбутньому.

Їх проблеми з іншими альтернативними компонентами будуть обмежені самим компонентом. Оскільки наявні компоненти підтримують кешування та повторне отримання даних, інші можливі компоненти повинні реалізувати можливість забезпечити кращий та більш чуйний користувальницький досвід. Нові компоненти також повинні бути розділені на менші частини, щоб зробити їх більш зрозумілими та спростити подальшу розробку.

Однією з проблем, яка може виникнути в майбутньому, є розмір компонентів, які складають сайт. Якщо розміри компонентів стають занадто великими, перехід від однієї частини програми до іншої може зайняти багато часу. Якщо

один клієнт має велику кількість відкритих вікон і занадто багато з них завантажуються або підтримуються в пам'яті одночасно, це може викликати проблеми. Для вирішення цих проблем необхідно розробити механізм використання простору веб-застосунку.

До технічних проблем, які залишаються у веб-додатку, як зазначалося раніше, ще належить їх виправити. Кількість спільних компонентів в системі проектування в даний час досить мінімальна. Цього слід дотримуватися, особливо з огляду на необхідність послідовного інтерфейсу користувача та досвіду по всій онлайн-програмі.

ВИСНОВКИ

Сайт, побудований і реалізований під час написання цього документу, все ще знаходиться на ранній стадії, з багатьма компонентами, які можна поліпшити і додати. Робота, виконана для веб-додатку, була повністю пояснена у документі, і, як зазначається в розділі «ОЦІНКА», було зроблено багато роботи, але в майбутньому доведеться зробити ще більше. Сподіваємося, що недоліки онлайн-програми та відсутні елементи будуть вирішені в майбутньому.

З огляду на початкову точку процесу проектування архітектури і висновки, наведені в розділі «ОЦІНКА», можна з упевненістю стверджувати, що починати процес проектування після значного обсягу реалізації не бажано. Найскладніші моменти розробки, що виникли, були пов'язані з тим, що був значний обсяг відсутніх робіт для проектної системи. Наприклад, відсутність системи проектування і процес включення її в існуюче застосування потребували найбільших зусиль.

Можна навіть стверджувати, що це вимагало більших зусиль, ніж решта

реалізації. В результаті ігнорування дизайну можна мати серйозні наслідки в майбутньому. Це питання слід вирішувати і краще враховувати в майбутніх подіях.

Дуже важливо розробити систему дизайну як частину загальної передньої архітектури, оскільки вона приносить користь як кінцевому користувачеві, так і розробнику. Це пов'язано з тим, що система проектування допомагає розробнику створити більш узгоджений користувальницький досвід з меншою роботою, використовуючи спільні компоненти системи проектування та інші утиліти. Оскільки елементи інтерфейсу користувача
відомі по всій системі, узгодженість інтерфейсу користувача дозволяє кінцевому користувачеві більш ефективно керувати системою. Регулювання кількості деталізації веб-програми, використовуючи фреймворк Material як основу для додавання стилів до сторінок веб-додатків, дозволило повністю використовувати існуючі компоненти, які раніше були реалізовані до початку архітектурного проектування.

На даний момент немає вагомих причин критикувати архітектуру сайту. Однією з переваг використання фреймворку Material в інтерфейсі, як зазначалося раніше, є те, що додатки технологічно незалежні. Це дозволяє легко оновлювати та змінювати технології, що використовуються в процесі розробки додатків. В результаті можлива найкраща технологія для кожної даної програми.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ КЛІЄНТСЬКА ЧАСТИНИ

	Ресурс
1.	Angular; Розробник: Google; Документація: https://angular.dev/
2.	Visual Studio Code; Розробник: Microsoft; Документація: https://code.visualstudio.com/docs
3.	TypeScript; Розробник: Microsoft; Документація: https://www.typescriptlang.org/docs/
4.	Material angular; Розробник: Google; Документація: https://material.angular.io/

5.	HTML; Розробник: WHATWG; Документація: https://developer.mozilla.org/en-US/docs/Web/HTML

6.	SASS; Розробник: SASS team; Документація: https://sass-lang.com/documentation/
7.	FETCH API; Документація: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API