Міністерство освіти і науки України

Університет митної справи та фінансів

Факультет інноваційних технологій

Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка веб сервісу для замовлення та надання послуг перукаря»

Виконала: студентка групи ІПЗ19-2

Спеціальність 121 Інженерія програмного забезпечення

Решетник Катерина Олександрівна

(прізвище та ініціали)

Керівник доц. Мала Ю.А.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент _____

(місце роботи)

_____

(посада)

_____

(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

АНОТАЦІЯ

Решетник К.О. Розробка веб сервісу для замовлення та надання послуг перукаря. Кваліфікаційа робота успішно виконав на здобуття бакалаврського ступеня за спеціальністю 121 "Інженерія програмного забезпечення" в Університеті митної справи та фінансів у 2023 році. Основним завданням роботи було розроблення вебсервісу для замовлення та надання послуг перукаря.

У результаті досліджень та розробок *Решетник К.О.* створила мобільний додаток, написаний мовою програмування Java, який надає зручний інтерфейс для користувачів, щоб замовляти різні послуги перукаря. Крім того, вебдодаток, розроблений з використанням мови програмування JavaScript, був створений для онлайн-платформи, де перукарі можуть пропонувати свої послуги та отримувати бронювання від клієнтів.

Проєкт допомагає перукарям залучати нових клієнтів, а також забезпечує зручність та швидкість процесу замовлення та надання послуг перукаря. Така інтернет-платформа для перукарів створює сприятливе середовище для зручного і надійного замовлення послуг, що підвищує якість обслуговування та задоволення клієнтів.

Ключові слова: розробка мобільного додатку, розробка вебсайт, онлайн-бронювання послуг, інтернет-платформа для перукарів.

ABSTRACTS

Reshetnyk K.O. Development of a web service for ordering and providing hairdresser services. The qualification work was successfully completed for a bachelor's degree in the specialty 121 "Software Engineering" at the University of Customs and Finance in 2023. The main task of the work was to develop a web service for ordering and providing hairdresser services.

As a result of research and development, *Reshetnyk K.O.* created a mobile application written in the Java programming language, which provides a user-friendly interface for users to order various hairdressing services. In addition, a web application developed using the JavaScript programming language was created for an online platform where hairdressers can offer their services and receive bookings from clients.

The project helps hairdressers attract new clients, as well as provides convenience and speed in the process of ordering and providing hairdressing services. Such an online platform for hairdressers creates a favorable environment for convenient and reliable ordering of services, which increases the quality of service and customer satisfaction.

Keywords: mobile application development, website development, online booking of services, online platform for hairdressers.

ЗМІСТ

ВСТУП

У сучасному світі спостерігається загальна тенденція до цифрової трансформації різних галузей, включаючи сферу краси. Розробка вебзастосунку для замовлення та надання послуг перукаря є одним з аспектів цієї трансформації, що може сприяти конкурентоспроможності та розвитку бізнесу. Цифрові технології вносять значні зміни в спосіб, яким ми шукаємо та замовляємо послуги. Клієнти стають все більш звиклими до зручності та швидкості онлайн-замовлень. Розробка вебзастосунку для замовлення та надання послуг перукаря надає можливість майстрам пристосуватись до цих змін та залучати нових клієнтів.

Мета кваліфікаційної роботи полягає в розробці такого додатка, який допомагатиме майстру у плануванні графіку роботи та відстеженні записів клієнтів. Створення власного додатку надає можливість перукарням персоналізувати його під свої потреби та бренд, вони можуть відобразити свої послуги, розклад роботи та іншу важливу інформацію,а також може бути інтегрований з системою управління перукарні, що спрощує процеси приймання замовлень, управління клієнтською базою та фінансами.

Актуальність роботи полягає в популяризації роботи майстрів-перукарів з моніторингу онлайн-обліків клієнтів та покращення якості наданих послуг.

Головні цілі які включені у розробку цього проєку:

- Ефективне планування графіку: додаток допоможе майстрам у плануванні їх робочого графіку. Вони зможуть встановити доступні дні та години, коли вони приймають клієнтів, щоб уникнути конфліктів у графіку та впевнено планувати свій час.
- Зручний запис клієнтів: клієнти матимуть можливість записатися до майстра через вебсайт. Додаток забезпечить зручний і легкий процес

запису, де клієнти зможуть обрати зручний для них час та послуги, які вони бажають отримати.

- Онлайн-облік записів: додаток буде відстежувати всі записи клієнтів для майстра. Майстер зможе переглянути свій розклад та бачити, коли він має найбільш заповнений графік або вільний час. Така реалізація допоможе забезпечити оптимальне управління часом та ресурсами.
- Покращення якості наданих послуг: запис клієнтів через додаток та вебсайт дозволить перукарю планувати свій робочий день більш ефективно, що дасть змогу майстру бути більш упевненим у своїх можливостях, надавати послуги вчасно та високої якості, що позитивно вплине на задоволення клієнтів.

Об'єктом предметного дослідження є цифрова трансформація сфери краси. Практична цінність полягатеме у забезпеченні конкурентоспроможності та розвитку бізнесу перукарень. Для досягнення поставленої мети будуть використані наступні методи дослідження:

1. Аналіз сучасних тенденцій: в рамках дослідження буде проведено глибокий аналіз цифрової трансформації різних галузей, зокрема сфери краси. Будуть вивчені тенденції у використанні цифрових рішень та інтернет-технологій, які дозволяють полегшити процеси планування та роботи перукарні. При цьому особлива увага буде приділена розумінню потреб та очікувань клієнтів.
2. Опитування та спостереження: в рамках дослідження будуть проведені опитування серед майстрів перукарів та їх клієнтів. Метою опитування буде виявлення поточних проблем, з якими зіштовхуються майстри та їх клієнти при плануванні робочого графіку та запису на послуги. Крім того, спостереження буде проведено для збору даних про реальну практику роботи перукарні та виявлення потреби у цифровому рішенні.

3. Розробка та тестування додатка: на основі отриманих даних та аналізу буде розроблений прототип додатка для перукарень. Цей додаток буде включати функціонал для планування графіку роботи, запису на послуги, збереження даних про клієнтів та їх вподобання. Додаток буде протестований на практиці з метою оцінки його ефективності та вдосконалення.
4. Аналіз результатів та рекомендації: після проведення тестування додатка будуть проаналізовані отримані результати. Будуть виділені переваги та недоліки розробленого додатка, а також надані рекомендації щодо його подальшого вдосконалення та впровадження в реальну практику.

Дослідницький проєкт має великий потенціал для розвитку сфери краси та покращення роботи перукарень. Розробка цифрового додатка, який допоможе оптимізувати процеси планування та взаємодії з клієнтами, може позитивно вплинути на конкурентоспроможність перукарень та забезпечити збільшення їхнього клієнтського потоку. Крім того, такий додаток може забезпечити зручність та комфорт клієнтам, спрощуючи процес запису на послуги та забезпечуючи доступ до необхідної інформації про перукарню.

Загалом, розробка даного додатка є важливим кроком у напрямку цифрової трансформації сфери краси. Застосовуючи використання сучасних технологій та інтернет-рішень допоможе підвищити ефективність та зручність роботи перукарні, забезпечуючи задоволення потреб клієнтів та сприяючи розвитку цієї сфери. Результати цього дослідження можуть бути використані як підґрунтя для подальшої роботи над цифровими інноваціями в галузі перукарського мистецтва.

РОЗДІЛ 1

ОПИС ЗАВДАННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ З ОБҐРУНТУВАННЯМ АКТУАЛЬНОСТІ ТЕМИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Опис завдання

Мета розробки даного проекту полягає у створенні мобільного додатку, який допомагатиме майстрам-перукарям у плануванні графіку роботи та відстеженні запису клієнтів.

Головним функціоналом додатку буде можливість майстру створювати свій робочий графік та відмічати у ньому доступні та зайняті часові інтервали. Майстер зможе встановлювати робочі години, перерви та інші параметри, що відповідають його робочому графіку.

Запис клієнтів буде відбуватися через веб-сайт, де клієнти матимуть можливість обрати послуку, дату та час візиту. Після запису клієнт отримає підтвердження про успішне бронювання.

Сам майстер зможе відстежувати записи клієнтів у встановленому мобільному додатку. Він буде мати доступ до свого календаря, де відображатимуться всі зроблені записи. Майстер отримуватиме повідомлення про нові записи та може вносити зміни до свого робочого графіку через додаток.

Під час розробки мобільного додатку використовувалася платформа Android Studio та мова програмування Java. Android Studio надає зручне середовище для розробки мобільних додатків на платформі Android, а Java є однією з основних мов програмування для розробки Android-додатків.

При розробці веб-сервісу для вебсайту використовувався текстовий редактор Visual Studio Code та мова програмування JavaScript.[10]

Visual Studio Code є популярним інструментом для розробки веб-додатків, а JavaScript є широко використовуваною мовою програмування для фронтенд розробки. JavaScript дозволяє динамічно взаємодіяти зі сторінкою та реалізувати необхідний функціонал веб-сайту.

Загальний результат розробки проекту буде передбачати мобільний додаток для майстрів-перукарів з планування графіку роботи та відстеження запису клієнтів, а також веб-сайт, де клієнти зможуть записатися до майстрів.

1.2 Актуальність обраної теми

Тема «Розробка веб сервісу для замовлення та надання послуг перукаря» залишається й досі актуальною і має потенціал для успішного впровадження в сучасному світі. Оскільки це допомагає автоматизувати та полегшити процес роботи для майстрів та забезпечити зручний спосіб запису клієнтів.

Основні причини чому ця тема є актуальною:

Зручність та доступність.

Веб-сервіси для замовлення та надання перукарських послуг пропонують зручність та доступність для клієнтів. Клієнти можуть швидко і легко знайти вільних перукарів, переглянути профіль майстра, відгуки клієнтів і ціни, а також забронювати зручний для них час.

Ефективне управління бізнесом.

Для самого перукаря або салону краси, веб-сервіс надає зручний інструментарій для управління розкладом роботи, обліку клієнтів, прийому замовлень та ведення фінансової обліковості. Таке дозволяє оптимізувати робочий процес, покращити ефективність та знизити ризик помилок.

Розширення клієнтської бази.

Веб-сервіси дають перукарям можливість привернути нових клієнтів шляхом представлення своїх послуг та портфоліо в онлайн-середовищі.

Більшість людей шукають перукарні та салони краси через Інтернет, тому наявність онлайн-сервісу може допомогти в залученні нових клієнтів та збільшенні прибутку.

Покращення взаємодії з клієнтами.

Веб-сервіси дозволяють забезпечити кращу взаємодію з клієнтами шляхом зручних інструментів для комунікації, таких як онлайн-чат, система відгуків та оцінок, сповіщення про підтвердження запису та нагадування про зустрічі.

Тенденція до цифрової трансформації.

У сучасному світі спостерігається загальна тенденція до цифрової трансформації різних галузей, включаючи сферу краси. Розробка веб-застосунку для замовлення та надання послуг перукаря є одним з аспектів цієї трансформації, що може сприяти конкурентоспроможності та розвитку бізнесу.

Враховуючи зручність, доступність та практичність такого веб-сервісу як інструменту для замовлення та надання послуг, можна стверджувати, що ця тема є актуальною і привабливою для розвитку та використанні в сучасному суспільстві.

1.3 Аналіз предметної області

Ринковий потенціал

Веб-сервіс для замовлення та надання послуг перукаря є інноваційним рішенням, яке має значний ринковий потенціал у сфері краси та перукарських послуг.

Ринковий аналіз

Аналіз ринку перукарських послуг показує стабільне зростання попиту на такі послуги. За останні роки споживачі все більше цінують зручність, доступність та швидкість отримання послуг через онлайн-платформи, це створює сприятливі умови для розвитку веб-сервісів, що спрямовані на замовлення та надання послуг перукаря.

Конкурентна перевага

Веб-сервіс для замовлення та надання послуг перукаря має кілька конкурентних переваг:

1. Зручність і доступність: користувачі можуть замовляти послуги перукаря в будь-який зручний для них час та місце, використовуючи лише пристрій з Інтернетом.
2. Розширений вибір: система надає користувачам доступ до різних перукарів з різними навичками та спеціалізаціями, дозволяючи їм знайти найбільш підходящого фахівця.
3. Зручна комунікація: користувачі можуть легко спілкуватись з перукарями через систему повідомлень, що спрощує організацію зустрічей та обговорення деталей.
4. Рейтинг та відгуки: наявність рейтингу та відгуків дозволяє користувачам приймати усвідомлені рішення при виборі перукаря і забезпечує додаткову довіру до сервісу.

Отже, веб-сервіс для замовлення та надання послуг перукаря має значний ринковий потенціал у сфері краси та перукарських послуг. Зручність, доступність та розширений вибір, які надає такий сервіс, сприяють задоволенню потреб споживачів і розвитку сучасного ринку перукарських послуг.

Функціональні вимоги

1. Реєстрація та авторизація:
   - Користувачі повинні мати можливість створювати облікові записи для доступу до сервісу.
   - Система повинна надати можливість входу в систему для зареєстрованих користувачів з використанням електронної пошти або соціальних мереж.
2. Перегляд доступних послуг:
   - Користувачам має бути надана можливість перегляду переліку доступних послуг, які надають перукарі.
   - Послуги повинні бути описані, включаючи назву, опис, ціну та тривалість надання.
3. Замовлення послуг:
   - Користувачі повинні мати можливість замовляти певну послугу.
   - Користувачі повинні мати можливість обрати дату та час зустрічі для послуги.
4. Підтвердження замовлення:
   - Перукар повинен мати можливість підтвердити або відхилити замовлення, отримане від користувача.
   - Користувач повинен бути повідомлений про статус свого замовлення, включаючи підтвердження або відхилення.
5. Управління профілями перукарів:
   - Перукарі повинні мати можливість створювати свої профілі, включаючи інформацію про свої навички, послуги, ціни, графік роботи тощо.
   - Перукарі мають змогу редагувати свої профілі та оновлювати інформацію про них.
6. Календар та сповіщення:

- Користувачам має бути надана можливість переглядати свій календар замовлень та зустрічей з перукарем.

7. Статистика та аналітика:
   - Адміністраторам системи має бути надана можливість переглядати статистичні дані про замовлення, відгуки, активність користувачів та інші метрики (це міра, що дозволяє отримати числове значення деяких властивостей програмного забезпечення та його специфікацій.).
   - Система повинна надати засоби для аналізу та генерації звітів зі зібраних даних.

Мобільна стратегія

Мобільна стратегія є також однією із важливих складових веб-сервісу. У зв'язку зі зростанням використання мобільних пристроїв, мобільний додаток дозволяє забезпечити зручний доступ до сервісу та покращити користувацький досвід.

Такі функції, як замовлення послуг, оповіщення та рейтинги, є важливими для успішної реалізації веб-сервісу для перукарів.

Ретельна розробка та реалізація мобільної стратегії сприятимуть популярності та успіху проекту.

Безпека

Огляд загальних аспектів безпеки:

Аутентифікація і авторизація: забезпечення безпеки облікових записів користувачів та перукарів шляхом надійної аутентифікації та обмеження доступу до функцій системи.

Захист даних: захист конфіденційності, цілісності та доступності даних, що зберігаються в системі, шляхом використання шифрування, механізмів контролю цілісності, резервного копіювання та інших заходів.

Функціональні аспекти безпеки:

Безпека авторизації: використання сильних паролів, двофакторної аутентифікації та інших методів для запобігання несанкціонованому доступу до облікових записів.

Керування правами доступу: забезпечення правильного призначення рівнів доступу користувачам і перукарам, обмеження доступу до конфіденційної інформації.

Захист від вразливостей: регулярне оновлення програмного забезпечення, використання надійних бібліотек та фреймворків, виявлення та виправлення вразливостей.

Політики безпеки та згода з вимогами:

Визначення політик безпеки: розробка та документування політик безпеки, які включають правила для користувачів, перукарів і адміністраторів системи.

Основні аспекти безпеки, які повинні бути враховані при розробці додатку для замовлення та надання послуг перукаря.

Управління ресурсами

Ефективне управління ресурсами дозволяє оптимізувати використання доступних ресурсів, забезпечити належне функціонування сервісу і забезпечити задоволення потреб користувачів.

Конкретний функціонал може бути адаптований під потреби проекту та вимоги користувачів. Детальне планування та управління ресурсами допомагають забезпечити ефективне функціонування сервісу і задоволення клієнтів.

Користувацький досвід

1. Огляд функціональності

Реєстрація та авторизація користувачів.

Веб-сервіс надає можливість користувачам створювати облікові записи та авторизуватися в системі. Дозволяє користувачам здійснювати замовлення послуг та використовувати інші функції сервісу.

2. Перегляд доступних послуг

Користувачі мають можливість переглядати список доступних послуг, які пропонуються перукарями. Інформація про кожну послугу повинна бути чітко представлена, включаючи опис, тривалість, вартість та фотографії.

3. Замовлення послуг

Користувачі можуть обирати певну послугу зі списку та вказувати дату та час зустрічі. Додаткові вимоги або коментарі можуть бути також додані. Після надання усієї необхідної інформації користувачі підтверджують своє замовлення.

4. Підтвердження замовлення

Перукарі мають можливість підтвердити або відхилити замовлення, яке надійшло від користувача. У разі підтвердження замовлення, інформація про це повідомляється користувачеві.

5. Інтерфейс користувача

Зручність навігації.

Веб-сервіс повинен мати зрозумілий та інтуїтивно зрозумілий інтерфейс для легкої навігації користувачів. Головне меню повинно бути видимим і доступним на всіх сторінках, з можливістю швидко переходити до необхідних розділів.

Привабливий дизайн.

Дизайн веб-сервісу має бути привабливим та естетичним, з використанням приємних кольорових схем та графічних елементів, що сприятеме позитивному враженню користувачів та створює гармонійну атмосферу.

Інтуїтивний процес замовлення.

Процес замовлення повинен бути логічним та легким у використанні. Користувачі повинні легко знайти необхідні послуги, обрати бажану дату та час. Інтерфейс повинен відображати усю необхідну інформацію та кроки, щоб уникнути плутанини.

6. Комунікація та повідомлення

Сповіщення про стан замовлення.

Користувачі повинні отримувати сповіщення про статус свого замовлення, такі як підтвердження, скасування або зміни, яке забезпечить користувачам бути в курсі процесу та планувати свій час.

Таким чином, веб-сервіс має бути зручним, інтуїтивно зрозумілим та привабливим для користувачів. Розглянуті аспекти, такі як функціональність, навігація, дизайн та комунікація, впливають на загальний користувацький досвід.

1.4 Технічне завдання проекту

1 Огляд проекту.

1.1 Назва проекту: додаток для планування графіку та відстеження запису клієнтів майстрам.

1.2 Опис проекту: розробка мобільного додатку для майстрів (перекурів), який допомагатиме їм планувати графік роботи та відстежувати запис клієнтів, які реєструються через веб-сайт.

2 Функціональні вимоги.

2.1 Реєстрація майстра:

- Майстер-перукар створює обліковий запис, вказуючи особисті дані (ім'я та прізвище, електронну пошту, пароль).
- Після реєстрації, майстер отримує підтвердження про успішну реєстрацію.

2.2 Аутентифікація майстра:

- Майстер може увійти в додаток, використовуючи свій облікові дані (електронна пошта та пароль).

2.3 Планування графіку роботи:

- Майстер може обирати графік роботи на певні дні тижня.
- Задання робочих годин для кожного дня.
- Можливість встановлювати перерви між клієнтами.

2.4 Відстеження запису клієнтів:

- Майстер отримує сповіщення про новий запис клієнта.
- Можливість переглядати список запланованих клієнтів на кожен день.

3 Технічні вимоги.

3.1 Мобільний додаток:

- Розробка на платформі Android з використанням Android Studio та мови програмування Java.
- Аутентифікація майстра використовуючи платформу для розробки застосунків Firebase з використанням облікових даних (електронна пошта, пароль).
- Використання локальної бази даних для зберігання облікових записів майстрів та графіку роботи.
- Повідомлення про нові записи клієнтів за допомогою push-сповіщень.

3.2 Вебсервіс.

- Розробка на основі JavaScript з використанням VS Code редактора.
- Створення вебсайту для реєстрації та запису клієнтів.
- Збереження даних про клієнтів, записи та послуги в базі даних.
- Надання RESTful API для взаємодії з мобільним додатком.

4 Інтерфейс користувача.

4.1 Мобільний додаток:

- Початковий екран з полями для введення електронної пошти та пароля для аутентифікації.
- Екран планування графіку роботи з можливістю редагування днів та годин роботи.
- Екран списку запланованих клієнтів з детальною інформацією про кожного клієнта.

4.2 Вебсайт.

- Головне вікно з позицією календаря, для вибору доступних днів і часу для запису.
- Вікно реєстрації клієнтів з заповненням полів вводу (ім'я та прізвище, номер телефону,засначення обраної послуги, час).
- Вікно про успішне бронювання.

5 Завдання для розробки.

- Розробка мобільного застосунку для Android з використанням Android Studio та мови програмування Java.
- Розробка веб-сервісу, який дозволяє клієнтам обирати та бронювати необхідні їм послуги.
- Інтеграція між мобільним додатком та веб-сервісом через RESTful API.
- Тестування та налагодження додатку.
- Обґрунтування процесів у звіті, про розробку та функціональні можливості застосунку.

1.6 Висновок до розділу 1

Розробка веб-застосунку для замовлення та надання послуг перукаря є актуальною у контексті цифрової трансформації галузі краси. Проведений аналіз предметної області показав, що хоча на ринку вже є схожі продукти, розробка власного веб-застосунку дозволяє перукарням персоналізувати продукт та впроваджувати додаткові функції. Технічне завдання проекту визначає основні функції та вимоги до веб-застосунку. В цілому, розробка такого веб-застосунку може сприяти покращенню конкурентоспроможності та розвитку бізнесу перукарень.

РОЗДІЛ 2

ОГЛЯД ІСНУЮЧИХ РОЗВ'ЯЗКІВ ПОСТАВЛЕНОЇ ЗАДАЧІ, ЇХ АНАЛІЗ ТА ВИБІР МЕТОДІВ ВИРІШЕННЯ

2.1 Існуючі розв’язки поставленої задачі

У сучасному світі спостерігається зростаюча потреба у цифрових рішеннях для замовлення та надання послуг в тому числі є і перукарні. На сьогоднішній день існує кілька розв'язків, які стосуються замовлення та надання послуг перукаря. Деякі з них вже успішно функціонують на ринку та набули популярності серед перукарень та їхніх клієнтів.

Одним з таких розв'язків є вебзастосунок Booksy, який дозволяє клієнтам шукати перукарів, переглядати їхні портфоліо, переговорювати щодо доступності та замовляти послуги.

Booksy був запущений в 2013 році в Сполучених Штатах Америки. Згодом він почав розширюватись та набувати популярності в різних країнах світу, включаючи Європу, Азію та Австралію.

Також працює в Україні. Він вже став популярним серед перукарень та їхніх клієнтів, надаючи зручний інструмент для замовлення та керування послугами перукаря. Клієнти можуть шукати перукарні у своєму регіоні, переглядати інформацію про них, переговорювати про доступність та записуватися на зручний для них час.

Застосунок Booksy є одним з відомих розв'язків в сфері краси та перукарського бізнесу, який допомагає покращити процес замовлення та надання послуг, спрощує комунікацію між перукарнями та їхніми клієнтами та забезпечує зручність і ефективність усім сторонам.

Другим за популярністю є вебзастосунок, StyleSeat, надає аналогічні можливості замовлення та пошуку перукаря. Клієнти можуть вибирати перукарів на основі відгуків та рейтингу, переговорювати про деталі та підтверджувати свої записи. StyleSeat також дозволяє перукарям управляти своїм розкладом та розрахунки з клієнтами.

Третім - Treatwell, це онлайн-платформа, яка спеціалізується на бронюванні послуг у сфері краси. Клієнти можуть шукати доступних перукарів, переглядати фотографії та відгуки, а також бронювати часи в зручний для них спосіб.

Ці розв'язки допомагають спростити процес замовлення та надання послуг перукаря, забезпечуючи зручний і доступний інтерфейс для клієнтів та перукарів. Вони також забезпечують підвищення видимості та конкурентоспроможності перукарень, а також поліпшення клієнтського досвіду.

Це були спільні характеристики між цими застосунками. Але щодо різниці між ними.[13] Ось кілька основних відмінностей:

1. Географічна доступність: Booksy, StyleSeat і Treatwell активно працюють у різних країнах і регіонах, але їх покриття може варіюватись. Деякі з них можуть бути більш популярними та доступними у певних регіонах, тому важливо переконатися, що платформа, яку вибирає користувач, буде підтримувати його місцезнаходження.
2. Функціонал: кожен з цих застосунків пропонує різноманітні функції. Booksy, наприклад, зосереджується на забезпеченні зручного розкладу та управлінні клієнтами для перукарів. StyleSeat забезпечує можливості пошуку перукарів на основі рейтингів та відгуків клієнтів. Treatwell спеціалізується на бронюванні послуг у галузі краси загалом, включаючи перукарів. Важливо визначати, які функції є в пріоритеті, щоб знайти платформу, яка буде краще відповідає потребам користувачів.

3. Рейтинги та відгуки: кожен з цих застосунків має систему рейтингів та відгуків, де клієнти можуть висловлювати свої думки про перукарів і послуги, які вони надають. Проте, способи збору та відображення рейтингів та відгуків можуть відрізнятись, і це може впливати на довіру клієнтів до певної платформи.
4. Інтеграція з платіжними системами: кожна платформа може мати власну систему оплати та обробки платежів. Деякі з них надають можливість безпосередньо здійснювати оплату через застосунок, що забезпечує зручність та швидкість процесу. Важливо ознайомитися з політикою оплати кожної платформи та переконатися, чи буде вона відповідати потребам користувачів.

Всі ці різноманітні застосунки мають спільну мету - полегшити процес замовлення та надання послуг перукаря. Вибір конкретної платформи залежить від потреб людини, географічного місцезнаходження та функцій, які необхідні.

Середовища розробки популярних сервісів для замовлення та надання послуг перукаря

Нажаль інформація про конкретні середовища розробки, які були використані при створенні застосунків Booksy, StyleSeat і Treatwell не є загально доступною. Але можна зробити лише загальні припущення про можливі середовища, що можуть бути використані при розробці цих веб-застосунків.

1. Фронтенд-розробка: для розробки користувацького інтерфейсу, який взаємодіє з клієнтами, можуть використовуватись HTML, CSS і JavaScript[3], а також фреймворки та бібліотеки, такі як React, Angular або Vue.js.
2. Бекенд-розробка: для реалізації серверної логіки та обробки даних можуть використовуватись різні технології, такі як Node.js, Python, Ruby

або PHP. Розробники можуть використовувати фреймворки, такі як Express для Node.js або Ruby on Rails, для прискорення розробки бекенду.

3. Бази даних: для зберігання даних, таких як інформація про перукарів, клієнтів, розклади та бронювання, можуть використовуватись різні бази даних, наприклад MySQL, PostgreSQL або MongoDB.
4. Хмарні платформи: для розгортання та масштабування застосунків можуть використовуватись хмарні платформи, такі як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform.

Припущення що базуються на загальноприйнятих практиках розробки веб-застосунків, але конкретні технології та середовища розробки можуть відрізнятися в залежності від вибору розробників та вимог проекту.

2.2 Аналіз і характеристика об’єкта проектування

Опис ідеї проекту: цей проект передбачає розробку мобільного додатку, який допомагатиме майстрам перукарям планувати свій графік роботи та відстежувати записи клієнтів. Клієнти матимуть можливість записатися до майстра через веб-сайт, а сам майстер буде мати доступ до цих записів через мобільний додаток.

Основні функціональні можливості:

1. Планування графіку роботи: майстри зможуть встановлювати свої робочі години, зазначати періоди, коли вони доступні для запису клієнтів.
2. Запис клієнтів: клієнти зможуть записуватися до майстра через веб-сайт, вказуючи бажану дату та час.

3. Відстеження записів: майстри зможуть переглядати список своїх записів та докладну інформацію про кожного клієнта, включаючи дату, час та послугу.
4. Нагадування та сповіщення: додаток може надсилати нагадування майстрам про надходження записів або зміни у графіку роботи, а також сповіщати клієнтів про підтвердження або зміни у їх записах.
5. Керування клієнтською базою: майстри зможуть зберігати інформацію про своїх клієнтів, включаючи контактні дані, історію записів та виконаних послуг.

Використані технології:

1. Мобільний додаток: розробка мобільного додатку здійснювалась за допомогою Android Studio та мови програмування Java.
2. Веб-сервіс: розробка веб-сервісу використовувала VS Code та мову програмування JavaScript.
3. База даних: для зберігання даних про майстрів, клієнтів, записи та іншу інформацію використовувалась Firebase.

Переваги проекту:

1. Зручність для майстрів: додаток допомагає майстрам перукарям ефективно планувати свій графік роботи та відстежувати записи клієнтів, спрощуючи їх робочий процес.
2. Зручність для клієнтів: клієнти можуть легко записатися до майстра через веб-сайт, обираючи зручну для них дату та час.
3. Нагадування та сповіщення: додаток надсилає нагадування майстрам та клієнтам, сповіщаючи їх про зміни у записах або графіку роботи.
4. Зручне керування клієнтською базою: майстри можуть зберігати інформацію про клієнтів та відстежувати історію записів та послуг.

2.3 Опис алгоритму і програмного забезпечення

Розробка веб-сайту.

1. Використання VS Code для розробки веб-сервісу з використанням мови JavaScript та HTML/CSS для створення користувацького інтерфейсу.
2. Встановлення та налаштування Firebase як бази даних та хостингу для збереження даних клієнтів та майстрів.[14]
3. Розробка функціональності реєстрації та авторизації для майстрів, що дозволяє їм створювати облікові записи та входити в систему.
4. Розробка сторінки для запису клієнтів, де клієнти можуть вибрати майстра, дату та час, коли вони бажають записатися.
5. Збереження даних про записи клієнтів у базі даних Firebase.

Розробка мобільного додатку:

1. Використання Android Studio для розробки мобільного додатку з використанням мови Java.
2. Розробка користувацького інтерфейсу додатку, де майстри можуть відстежувати записи клієнтів та планувати свій графік роботи.
3. Інтеграція з Firebase для отримання даних про записи клієнтів, які були зроблені через веб-сайт.[17]
4. Відображення списку записів клієнтів, включаючи дату, час та інформацію про клієнта.
5. Надання можливості майстрам встановлювати статус запису (прийнятий, відхилений, підтверджений) та нагадувань про наближення дати запису.

Забезпечення спільної роботи вебсайту та мобільного додатку.

1. Синхронізація даних між веб-сайтом та мобільним додатком, щоб майстри мали доступ до оновленої інформації про записи клієнтів на обох платформах.

2. Запити до Firebase для отримання та оновлення даних про записи клієнтів.[16]

Загалом, програмне забезпечення складається з веб-сайту, розробленого з використанням VS Code та мови JavaScript, та мобільного додатку, розробленого з використанням Android Studio та мови Java. Firebase використовується як база даних для збереження та обміну даними між веб-сайтом та мобільним додатком.

2.4 Вибір і обґрунтування структури проектування системи

При проектуванні системи для планування графіку та відстеження запису клієнтів майстрів-перукарів було використано таку структуру проектування системи:

1. Клієнт-серверна архітектура: рекомендується використовувати клієнт-серверну архітектуру,[7] де веб-сайт та мобільний додаток виступають як клієнти, а веб-сервер з базою даних Firebase виступає як сервер, що дозволить забезпечити розділення обов'язків між клієнтською та серверною частинами, а також забезпечити спільний доступ до даних.
2. Фронтенд та бекенд: розділіть функціонал між фронтендом (веб-сайт, мобільний додаток) та бекендом (серверна частина). Фронтенд буде відповідати за інтерфейс користувача, взаємодію з користувачем та відображення даних, а бекенд буде відповідати за бізнес-логіку, збереження та обробку даних.
3. Модульна структура: розгляньте можливість розбиття системи на модулі або компоненти для полегшення розробки та підтримки. Наприклад, модуль аутентифікації, модуль записів клієнтів, модуль календаря тощо. Це дозволить зосередитися на розробці та тестуванні окремих компонентів системи.

4. Інтеграція з Firebase: firebase може виступати як центральна база даних та сервер для збереження і обміну даними між веб-сайтом та мобільним додатком. Використання Firebase дозволить спростити розробку серверної інфраструктури, а також забезпечить масштабованість та безпеку даних.
5. API: розробіть API для взаємодії між фронтендом та бекендом - це дозволяє забезпечити стандартизований спосіб обміну даними між компонентами системи, а також легшу інтеграцію з іншими системами або сторонніми сервісами.
6. Безпека: зверніть увагу на захист даних і безпеку системи. Застосуйте методи аутентифікації та авторизації для доступу до функціоналу системи. Розгляньте застосування шифрування для збереження чутливих даних, таких як паролі користувачів.
7. Масштабованість: врахуйте можливість масштабування системи у майбутньому. При проектуванні системи слід враховувати зростання кількості користувачів та обсягу даних. Застосуйте оптимізацію запитів до бази даних та розподілення навантаження, якщо це необхідно.
8. Тестування: забезпечте належне тестування системи та її компонентів. Використовуйте автоматизовані тести для перевірки функціональності та якості коду. Розробіть юніт-тести для перевірки окремих компонентів та інтеграційні тести для перевірки взаємодії між компонентами.

Структура проектування системи та її компонентів забезпечить гнучкість, масштабованість та безпеку проекту, а також спростить розробку та підтримку системи.

2.5 Інструкція роботи користувача з системою

Для майстрів (користувачів мобільного додатку)

1. Завантаження та встановлення додатку.

Завантажте та встановіть мобільний додаток з магазину додатків на своєму пристрої.

2. Реєстрація/вхід у систему.

При першому запуску додатку, створіть свій обліковий запис, вказавши необхідну інформацію (ім’я та прізвище, пошта, пароль) або увійдіть в існуючий обліковий запис вказавши (пошту та пароль), якщо ви вже зареєстровані.

3. Перегляд записів клієнтів.

Після успішного входу до системи, ви побачите календар, при натисканні на дату буде з’являтися вікно з записом клієнтів на цей день, які були зроблені через вебсайт. Ви можете переглянути інформацію про кожен запис, таку як (час, послугу та ім'я клієнта).

4. Планування графіку та встановлення статусу запису.

За допомогою календаря в додатку, ви можете планувати свій графік роботи, встановлюючи доступні часи та дати.

Коли клієнт здійснює запис через вебсайт, ви отримуєте сповіщення про новий запис і можете підтвердити або відхилити.

Клієнт (користувач вебсайту)

1. Відкриття вебсайту:

Запустіть веббраузер на своєму пристрої і відкрийте вебсайт, який був розроблений для запису клієнтів до майстрів.

2. Бронювання:

Оберіть зручну дату та час для вашого запису.

3. Підтвердження запису:

Після заповнення полів з вказаними вашими даннимі такі, як (ім’я та прізвище, номер телефону, послуга та час), підтвердьте свій запис на веб-сайті.

Зачекайте на підтвердження від майстра через мобільний додаток.

2.6 Висновок до розділу 2

На підставі аналізу існуючих розв'язків та врахування потреб бізнесу перукарень і клієнтів, можливо розробити веб-застосунок, який буде відповідати вимогам сучасного ринку та забезпечувати зручність, ефективність та задоволення користувачів. Деталі реалізації такого застосунку будуть розглянуті в наступному розділі.

РОЗДІЛ 3

ВИРІШЕНННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Розробка мобільного додатку

Розробка застосунку була реалізована в android studio на мові програмування java.

Створюємо проект з назвою CutGuruApp – це буде назва мобільного додатку. Вибраємо шаблон проекту Empty Activity. Це буде перше вікно, назвемо його LoginActivity.[9] Переходимо до файлу з розшиереннням activity_login.xml та відмальовуємо дизайн. (Додаток А)

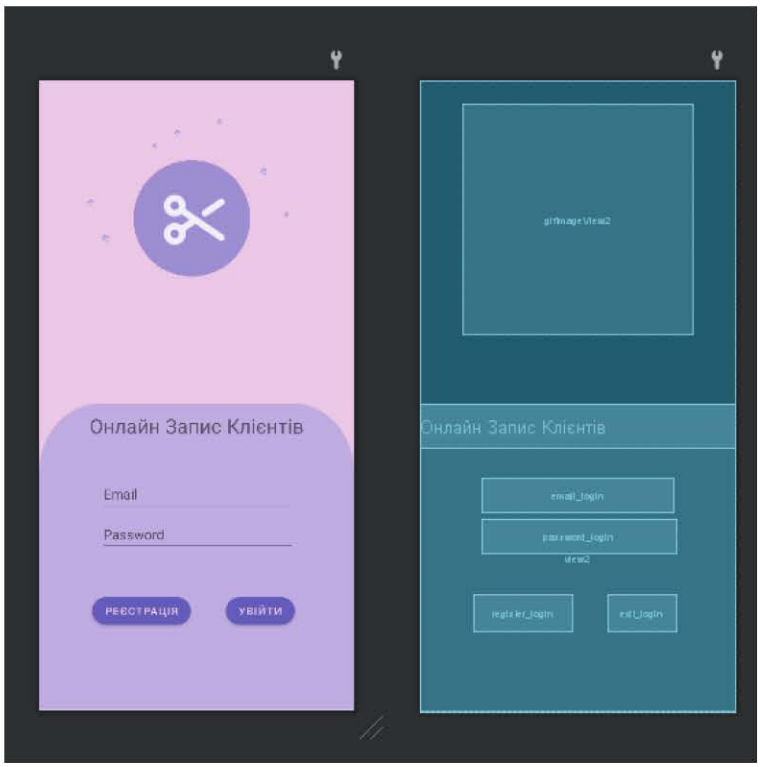

Рисунок 3.1.1 – екран входу activity_login.xml

Прописуємо функціонал входу у додаток в класі LoginActivity.

```
package com.example.cutguruapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;


public class LoginActivity extends AppCompatActivity {
    private EditText email_login;
    private EditText password_login;
    private Button exit_login;
    private Button register_login;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        register_login=findViewById(R.id.register_login);
        email_login = findViewById(R.id.email_login);
        password_login = findViewById(R.id.password_login);
        exit_login = findViewById(R.id.exit_login);

        mAuth = FirebaseAuth.getInstance();

       register_login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(LoginActivity.this,
RegisterActivity.class);
                startActivity(intent);
            }
        });


        exit_login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (email_login.getText().toString().isEmpty() ||
password_login.getText().toString().isEmpty()) {
                    Toast.makeText(LoginActivity.this, "Поля не можуть бути
порожніми.", Toast.LENGTH_LONG).show();
                } else {

mAuth.signInWithEmailAndPassword(email_login.getText().toString(),
password_login.getText().toString())
                            .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
```

```
                                @Override
                                public void onComplete(@NonNull
Task<AuthResult> task) {
                                    if (task.isSuccessful()) {
                                        Intent intent = new
Intent(LoginActivity.this, MainActivity.class);
                                        startActivity(intent);
                                    } else {
                                        Toast.makeText(LoginActivity.this,
"Не правильно введені дані.", Toast.LENGTH_LONG).show();
                                    }
                                }

                            });
                        }
                    }
                });
            }


}
```

Створюємо новий клас RegisterActivity. Переходимо до файлу з розшиеренням activity_register.xml та відмальовуємо дизайн. (Додаток Б)

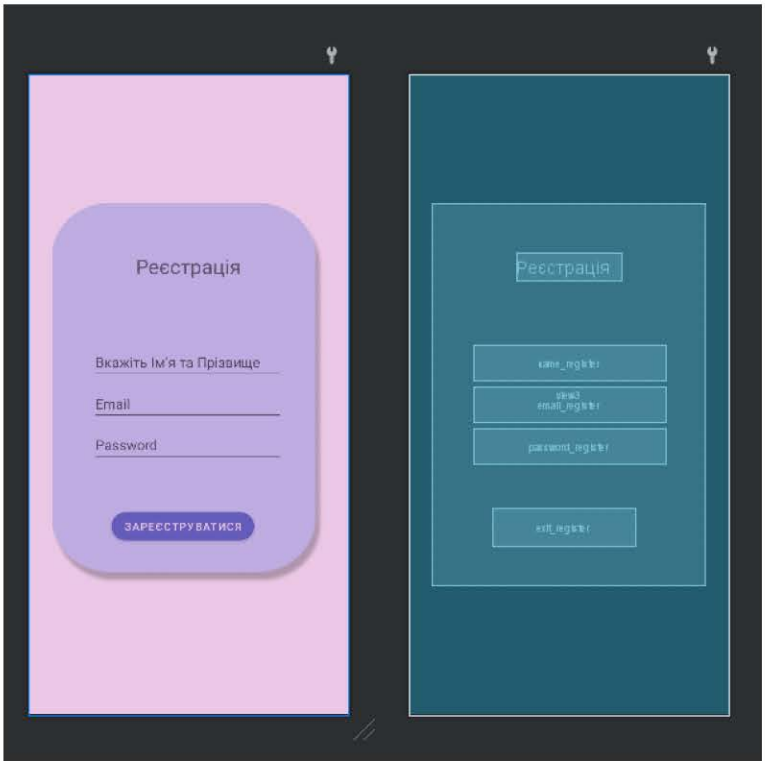

Рисунок 3.1.2 – екран реєстрації activity_register.xml

Прописуємо функціонал для реєстрації майстра.

```
package com.example.cutguruapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class RegisterActivity extends AppCompatActivity {

    private EditText name_register;
    private EditText email_register;
    private EditText password_register;
    private Button exit_register;
    private FirebaseAuth mAuth;
    private DatabaseReference database;
    private FirebaseDatabase user_register;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        mAuth=FirebaseAuth.getInstance();
        user_register=FirebaseDatabase.getInstance();
        database= user_register.getReference();


        name_register= findViewById(R.id.name_register);
        email_register= findViewById(R.id.email_register);
        password_register= findViewById(R.id.password_register);
        exit_register= findViewById(R.id. exit_register);

        exit_register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(name_register.getText().toString().isEmpty()||
email_register.getText().toString().isEmpty()||
password_register.getText().toString().isEmpty()) {
                    Toast.makeText(RegisterActivity.this,"Поля не можуть бути
порожніми.", Toast.LENGTH_LONG).show();
                }else {

mAuth.createUserWithEmailAndPassword(email_register.getText().toString(),pass
word_register.getText().toString())
                                .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                                    @Override
```

```
                                public void onComplete(@NonNull
Task<AuthResult> task) {
                                    if(task.isSuccessful()){

database.child("Users").child(mAuth.getCurrentUser().getUid()).child("name").
setValue(name_register.getText().toString());

database.child("Users").child(mAuth.getCurrentUser().getUid()).child("email")
.setValue(email_register.getText().toString());

database.child("Users").child(mAuth.getCurrentUser().getUid()).child("passwor
d").setValue(password_register.getText().toString());

                                        Intent intent= new
Intent(RegisterActivity.this, MainActivity.class);
                                        startActivity(intent);
                                    }else{

Toast.makeText(RegisterActivity.this,"Не правельно введений пароль або
логін\n"
                                                +"Спробуйте ще
pas",Toast.LENGTH_LONG);
                                    }

                                }
                            });

                    }
                }
            });

    }
    }
```

Після успішного входу/ реєстрації ми переходимо до головної сторінки додатку. Де буде календар в якому майстер буде відстежувати запис клієнтів.

Створюємо новий клас MainActivity та переходимо до файлу activity_main.xml в якому відмальовуємо дизайн сторінки. (Додаток В)

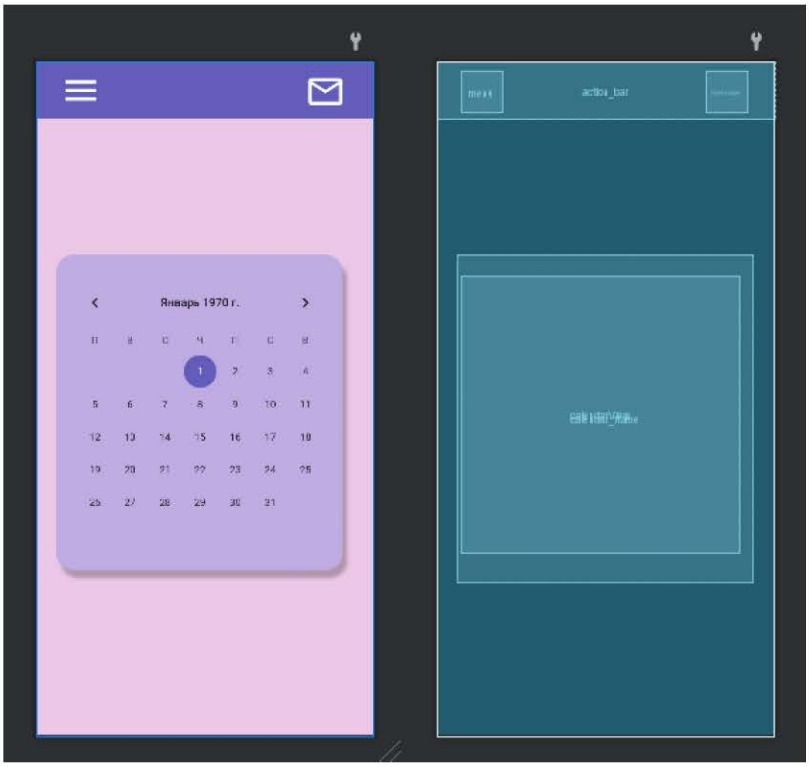

Рисунок 3.1.3 – головний екран activity_main.xml

В MainActivity прописуємо функціонал цієї сторінки.

```
package com.example.cutguruapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import android.app.Dialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.CalendarView;
import android.widget.CalendarView.OnDateChangeListener;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {
private CalendarView calendarView;

private Dialog eventDialog;
private View menu;
private View massage;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        calendarView=findViewById(R.id.calendarView);
```

```
        menu=findViewById(R.id.menu);
        massage=findViewById(R.id.massage);

        menu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
MenuActivity.class);
                startActivity(intent);
            }
        });
        massage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
MassageActivity.class);
                startActivity(intent);
            }
        });
        calendarView.setOnDateChangeListener(new OnDateChangeListener() {
            @Override
            public void onSelectedDayChange(CalendarView view, int year, int
month, int dayOfMonth) {
                showEventDialog(year, month, dayOfMonth);
            }
        });
    }


    private void showEventDialog(int year, int month, int dayOfMonth) {
        // Отримуємо заплановані події для вибраної дати (рік, місяць, день)
        String events = getEvents(year, month, dayOfMonth);

        // Перевіряємо, чи є заплановані події
        if (events.isEmpty()) {
            Toast.makeText(this, "Немає запланованих подій",
Toast.LENGTH_SHORT).show();
            return;
        }

        // Створюємо Dialog
        eventDialog = new Dialog(this);
        eventDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
        eventDialog.setContentView(R.layout.box_event_calendar);
        eventDialog.setCancelable(true);

        // Налаштування розміщення та вигляду вікна
        Window window = eventDialog.getWindow();
        if (window != null) {
            WindowManager.LayoutParams layoutParams = new
WindowManager.LayoutParams();
            layoutParams.copyFrom(window.getAttributes());
            layoutParams.width = WindowManager.LayoutParams.WRAP_CONTENT;
            layoutParams.height = WindowManager.LayoutParams.WRAP_CONTENT;
            layoutParams.gravity = Gravity.BOTTOM;
            window.setAttributes(layoutParams);
        }
        // Отримуємо посилання на time_event (час) та name_event (ім'я та
послуга)
        TextView time_event = eventDialog.findViewById(R.id.time_event);
        TextView name_event = eventDialog.findViewById(R.id.name_event);
```

```
// Розбиваємо рядок з подіями на окремі частини
        String[] eventArray = events.split("\n");

// Встановлюємо заплановані події у вікні Dialog
        CardView card_services_event =
eventDialog.findViewById(R.id.card_services_event);
        card_services_event.removeAllViews();

// Встановлюємо значення першої події у відповідні TextView
        String firstEvent = eventArray[0];
        String[] eventDetails = firstEvent.split(" - ");
        if (eventDetails.length >= 3) {
            String time = eventDetails[0];
            String service = eventDetails[1];
            String name = eventDetails[2];
            time_event.setText(time);
            name_event.setText(service + " - " + name);
        }

        eventDialog.show();
    }
    private String getEvents(int year, int month, int dayOfMonth) {
        // Отримуємо заплановані події з вашого джерела даних (наприклад,
бази даних, API і т.д.)
        // Повертаємо рядок з запланованими подіями для вибраної дати
        // Можете налаштувати цей метод відповідно до ваших потреб
        // Приклад використання псевдокоду:
        // events = Database.getEvents(year, month, dayOfMonth);
        String events = "8:00 AM - Послуга 1 - Ім'я 1\n" +
                "10:00 AM - Послуга 2 - Ім'я 2\n" +
                "2:00 PM - Послуга 3 - Ім'я 3";
        return events;
    }

}
```

Тепер реалізуємо екран «меню» в якому майстре зможе зазначати особисті данні(ім’я та прізвище, номер телефону, посаду, фото та посилання на сайт через який клієнти будуть бронювати місця). Нижче зробимо 3 кнопки («Розклад», «Послуги», «Запис»), які будуть відповідати назвам своїх сторінок. Реалізація екрану меню. (Додаток Г)

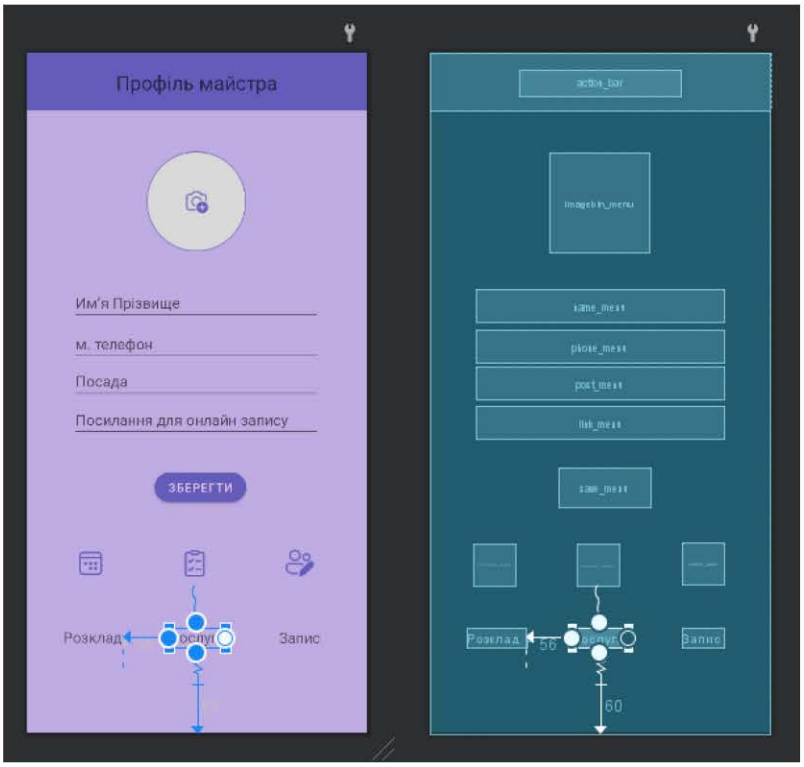

Рисунок 3.1.4 – екран меню activity_menu.xml

Прописуємо функціонал сторінки.

```
package com.example.cutguruapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

public class MenuActivity extends AppCompatActivity {

    private ImageButton
imagebtn_menu,schedule_menu,services_menu,clients_menu;
    private EditText name_menu,phone_menu,post_menu,link_menu;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);

        imagebtn_menu=findViewById(R.id.imagebtn_menu);
        schedule_menu=findViewById(R.id.schedule_menu);
        services_menu=findViewById(R.id.services_menu);
        clients_menu=findViewById(R.id.clients_menu);

        name_menu=findViewById(R.id.name_menu);
        phone_menu=findViewById(R.id.phone_menu);
```

```
        post_menu=findViewById(R.id.post_menu);
        link_menu=findViewById(R.id.link_menu);
        schedule_menu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
ScheduleActivity.class);
                startActivity(intent);
            }
        });
        services_menu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
ServicesActivity.class);
                startActivity(intent);
            }
        });
        clients_menu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MenuActivity.this,
ClientsActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

Переходимо до реалізації цих екранів(«Розклад», «Послуги», «Запис»). (Додаток Ґ)

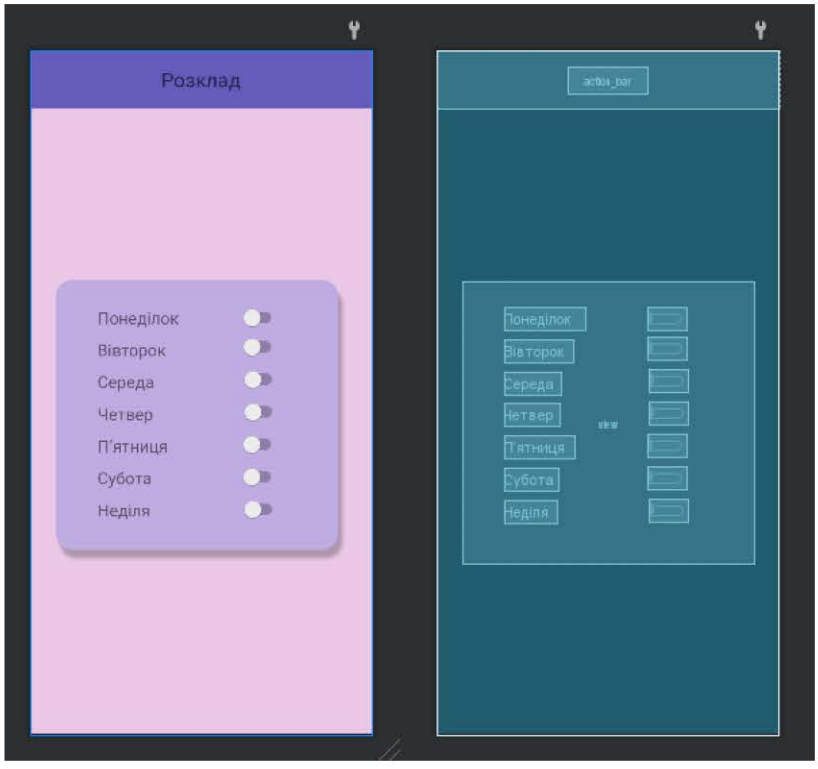

Рисунок 3.1.5 – екран розкладу activity_schedule.xml

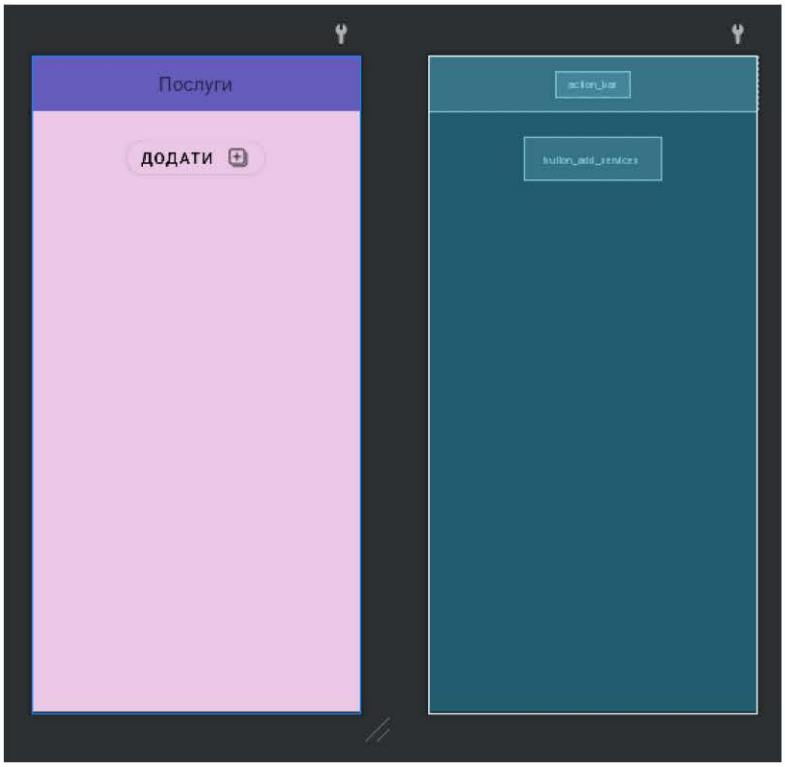

Рисунок 3.1.6 – екран послуги activity_services.xml

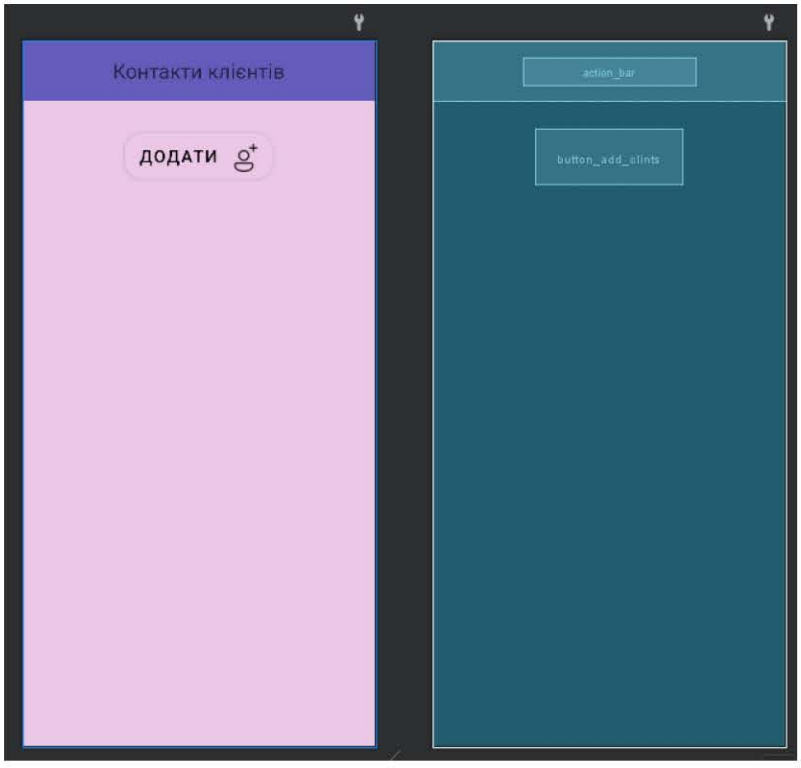

Рисунок 3.1.7 – екран записів activity_clients.xml

Прописуємо функціональні можливості цих сторінок.

```
package com.example.cutguruapp;


import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.view.Gravity;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.TimePicker;



public class ScheduleActivity extends AppCompatActivity {

    private Switch switch1,switch2,switch3,switch4,switch5,switch6,switch7;
    Dialog dialog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_schedule);
        switch1 = findViewById(R.id.switch1);
        switch2=findViewById(R.id.switch2);
        switch3=findViewById(R.id.switch3);
        switch4=findViewById(R.id.switch4);
        switch5=findViewById(R.id.switch5);
        switch6=findViewById(R.id.switch6);
        switch7=findViewById(R.id.switch7);
        dialog=new Dialog(ScheduleActivity.this);

        switch1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
            }
        });
        switch2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
            }
        });
        switch3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
            }
        });
        switch4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
```

```
            }
        });
        switch5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
            }
        });
        switch6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
            }
        });
        switch7.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAddDialogTime();
            }
        });
    }
    private void showAddDialogTime() {
        dialog.setContentView(R.layout.box_timer);
        dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
        dialog.setCancelable(true);

        TextView time_work_form1 = dialog.findViewById(R.id.time_work_form1);
        TextView time_work_form2 = dialog.findViewById(R.id.time_work_form2);
        TextView break work form1 =
dialog.findViewById(R.id.break_work_form1);
        TextView break work form2 =
dialog.findViewById(R.id.break_work_form2);
        Window window = dialog.getWindow();
        if (window != null) {
            WindowManager.LayoutParams layoutParams = new
WindowManager.LayoutParams();
            layoutParams.copyFrom(window.getAttributes());
            layoutParams.width = WindowManager.LayoutParams.WRAP_CONTENT;
            layoutParams.height = WindowManager.LayoutParams.WRAP_CONTENT;
            layoutParams.gravity = Gravity.BOTTOM;
            window.setAttributes(layoutParams);
        }

        time_work_form1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showTimePickerDialog(time_work_form1);
            }
        });

        time_work_form2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showTimePickerDialog(time work form2);
            }
        });

        break work form1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                showTimePickerDialog(break_work_form1);
            }
        });

        break_work_form2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showTimePickerDialog(break_work_form2);
            }
        });

        dialog.show();
    }
    private void showTimePickerDialog(final TextView textView) {
        TimePickerDialog timePickerDialog = new TimePickerDialog(this,new
TimePickerDialog.OnTimeSetListener() {
            public void onTimeSet(TimePicker timePicker, int hourOfDay, int
minute) {
                String time= String.valueOf(hourOfDay) + ":" +
String.valueOf(minute);
                textView.setText(time);
            }
        },10,00,true);
        timePickerDialog.show();
    }

}
```

У даному розділі було проведена розробка мобільного додатку для замовлення та надання послуг перукаря. В процесі розробки було використано Android Studio як основне інтегроване середовище розробки (IDE), Java.

3.2 Розробка вебсайту

Розробка вебсайту була реалізована в Visual Studio Code на мові JavaScript з використанням HTML/CSS. (Додаток Д)

Реалізація головного вікна.

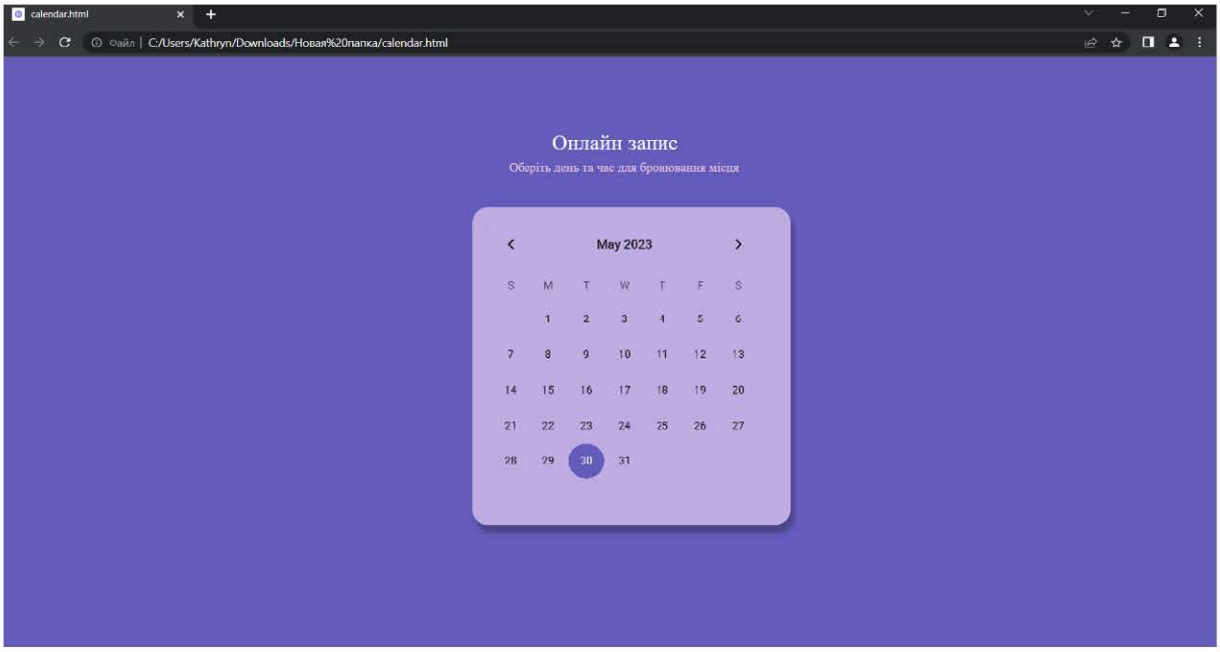

Рисунок 3.2.1- головне віно вебсайту

Реалізація вікна бронювання, де клієнт вказує ім'я та прізвище, номер телефону, обирає доступні послуги та час. (Додаток Д)

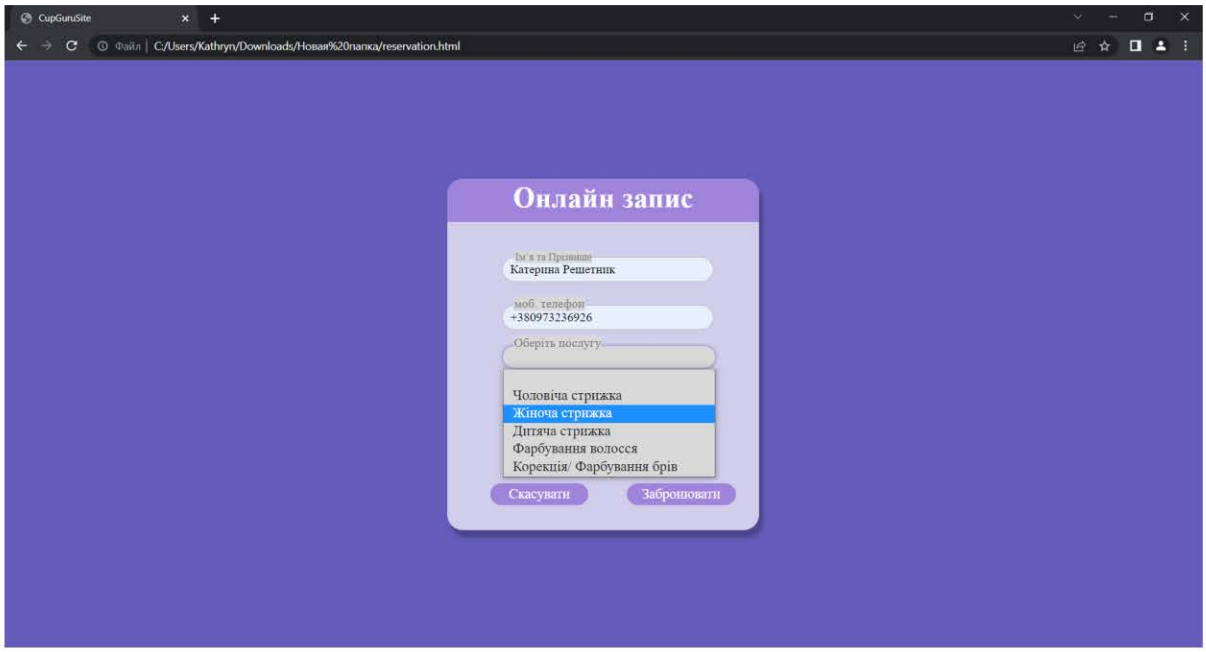

Рисунок 3.2.2- віно бронювання

Після заданих параметрів при натисканні кнопки «забронювати» з'являється повідомлення про успішність проведеного бронювання, що підтверджує клієнтам їхнє замовлення було успішно зареєстровано та зафіксовано в системі. Таке повідомлення надає клієнтам впевненість у тому,

що їхні вимоги будуть виконані та візит до перукарні буде запланований відповідно до їхніх побажань. (Додаток Д)

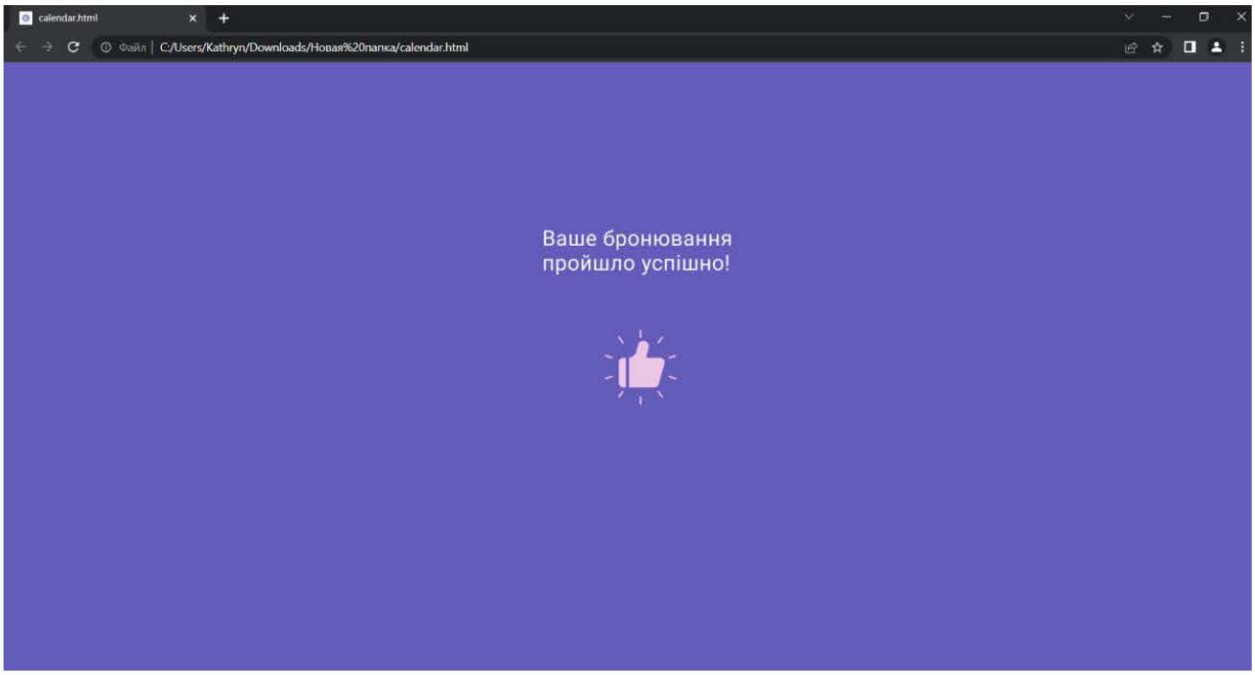

Рисунок 3.2.3- повідомлення про успішної бронювання

В рамках технічного завдання проекту були використано такі інструменти та технології, як Visual Studio Code та JavaScript. Розробка вебсайту передбачала створення інтерактивного і респонсивного інтерфейсу, що працює на різних пристроях та браузерах.

3.2 Висновок до розділу 3

В даному розділі була показана розробка мобільного додатка та вебсайту застосування якого дає можливість залучити нових клієнтів, полегшує процес замовлення та надання послуг і сприяє покращенню загального досвіду користувачів. Використання сучасних технологій та зручних інструментів допомагає перукарням підтримувати конкурентоспроможність та ефективно використовувати ресурси.

ВИСНОВОК

Під час виконання кваліфікаційної роботи було розроблено веб-застосунок для допомоги майстру перукаря у плануванні графіку та відстеженні записів клієнтів. Враховуючи загальну тенденцію до цифрової трансформації різних галузей, розробка такого застосунку стає важливим аспектом покращення конкурентоспроможності та розвитку бізнесу у сфері краси.

Цифрові технології значно змінюють спосіб, яким ми замовляємо послуги, і розробка веб-застосунку для замовлення та надання послуг перукаря відповідає цьому тренду. Завдяки такому застосунку клієнти можуть швидко і зручно знайти перукарню, ознайомитися з послугами, записатися на зручний для них час та отримати якісні послуги. В той же час, перукарні отримують можливість привернути нових клієнтів, персоналізувати веб-застосунок під свої потреби та зручно управляти графіком роботи та клієнтською базою.

Такі онлайн-сервіси для перукарень мають значний потенціал на ринку послуг. Вони дозволяють забезпечити зручну та ефективну взаємодію між клієнтами та майстрами, сприяють зростанню аудиторії та покращенню якості наданих послуг. Застосунок, розроблений в рамках цієї роботи, має великий потенціал на ринку, оскільки надає можливість замовлення послуг перукаря з будь-якого місця, де є доступ до Інтернету. Унікальний дизайн, спеціалізовані функції та розширені можливості, персоналізація та покращений клієнтський досвід, а також розширені аналітичні можливості роблять цей застосунок привабливим для клієнтів та перукарів. Загалом, цей застосунок є потужним інструментом для замовлення та надання послуг перукаря, який може стати ключовим фактором успіху на ринку.

Отже, кваліфікаційна робота підтвердила актуальність розробки веб-застосунку для перукарень. Вона підкреслила потребу у цифрових рішеннях

для полегшення процесу запису клієнтів, управління графіком роботи та збереження даних про клієнтів.

Основні функції розробленого веб-застосунку включають.

Реєстрація користувачів. Клієнти та перукарі можуть створити свої облікові записи для використання застосунку. Це дозволяє зберігати персоналізовану інформацію та забезпечує безпеку даних.

Пошук та вибір перукарень. Клієнти можуть переглядати перукарні, ознайомлюватися з профілями перукарів, переглядати фотографії робіт та відгуки клієнтів. Це допомагає їм знайти перукарню, яка відповідає їхнім потребам.

Запис на послуги. Клієнти можуть переглядати графік роботи перукаря та обирати зручний для себе час. Вони можуть записуватися на певну послугу та отримувати підтвердження про запис.

Управління графіком роботи: перукарі можуть вести свій графік роботи, вказувати доступні часові проміжки та оновлювати їх у режимі реального часу. Це дозволяє їм легко керувати своїм робочим часом і уникати перекриття записів.

Сповіщення: застосунок надсилає клієнтам та перукарям сповіщення та нагадування про надходження записів, зміни графіку або скасування, що допомагає уникнути забуття або непорозумінь.

Управління клієнтською базою: застосунок зберігає інформацію про клієнтів, таку як контактні дані та попередні надані послуги, які дозволятимуть перукарям надавати персоналізовані послуги та підтримувати довгострокові відносини з клієнтами.

Звітність та аналітика: застосунок надає зручні інструменти для генерації звітів та аналізу даних. Дозволить перукарям оцінювати продуктивність, відстежувати тренди та приймати обґрунтовані управлінські рішення.

Таким чином цей застосунок для перукарень може спростити процес управління бізнесом, поліпшити задоволення клієнтів та забезпечити більш ефективну роботу перукарні. Також в подальшому можна буде в досконавлювати цей проєк, додаючи нові функціональні можливості та застосовувати нові технології для покращення роботи додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джозеф Хіґні Основи управління проєктами: Планування проекту. Україна, 2020. 47с.
2. Пол Дж. Філдінг Як керувати проєктами: Межі проєкту. Фабула, 2020. 56с.
3. Маріяна Хавербеке Виразнийt JavaScript: Проєкт: електроне життя. 2015. 121с.
4. Романовський О. О. Шляхи впровадження інновацій, підприємництва та підприємницької освіти в системі національної освіти України: проєкт ООП. Вінниця, 2010. 139с.
5. The Impact Of Agile Methodologies On Code Quality — Smashing Magazine. (2023, May 25). Smashing Magazine. https://www.smashingmagazine.com/2023/05/impact-agile-methodologies-code-quality/
6. Design Risks: How to Assess, Mitigate, and Manage Them. (n.d.). Nielsen Norman Group. https://www.nngroup.com/articles/design-risk-management/
7. Посібник з архітектури програми. URL: https://developer.android.com/topic/architecture
8. Бібліотека прив'язки даних. URL: https://developer.android.com/topic/libraries/data-binding
9. Налаштуйте панель додатків. URL:https://developer.android.com/develop/ui/views/components/appbar/setting-up
10. Робота з JavaScript у коді Visual Studio URL:https://code.visualstudio.com/docs/languages/javascript
11. Основні елементи управління: TimePicker. URL:https://metanit.com/java/android/4.16.php

12. Діалогові вікна: DatePickerDialog и TimePickerDialog. URL:https://metanit.com/java/android/18.1.php
13. Compare Booksy vs Treatwell 2023 | Capterra. (n.d.). Capterra. URL:https://www.capterra.com/salon-software/compare/142741-181827/Booksy-vs-Treatwell
14. Документація: Firebase для Android. URL:https://firebase.google.com/docs/android/setup
15. Firebase: додавання до проекту JavaScript. URL:https://firebase.google.com/docs/web/setup
16. Аутентифікація Firebase. URL: https://firebase.google.com/docs/auth
17. База даних: Firebase Realtime Database. URL:https://firebase.google.com/docs/database