Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка кростплатформеного мобільного додатку для менеджменту міні-готелів»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1812»

_____ /Олена ЗАВЕРТЯЄВА/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Олександр ІВАНОВ/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of cross-platform mobile application management for small hotels»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group ПІ31812: /Olena ZAVERTIAIEVA/

Scientific Supervisor: /Oleksandr IVANOV/

Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: ~~Факульт~~ет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Завертяєвій Олени Ігорівні
1. Тема роботи: «Розробка кростплатформеного мобільного додатку для менеджменту міні-готелів»
Керівник роботи: Іванов Олександр Петрович, доцент
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Вступ
Збір та аналіз вимог до мобільного додатку
Проектування мобільного додатку
Проектування прототипу мобільного додатку
Розробка прототипу мобільного додатку
Розробка мобільного додатку
Тестування мобільного додатку
Загальні висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
Презентація
Відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи
1	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки.	31.01.22 – 02.02.22
2	Попередній вибір методів рішення задач.	03.02.2022 –05.02.2022
3	Визначення вимог до технічних засобів.	06.02.2022 – 16.02.2022
4	Узгодження і затвердження технічного завдання.	17.02.2022 – 18.02.2022
5	Програмування та відлагодження програми.	19.02. 2022 – 15.05.2022
6	Тестування програми	16.05.2022 – 20.05.2022
7	Розробка, узгодження і затвердження програмної документації.	21.05.2022 –12.06.2022
8	Подання кваліфікаційної роботи до кафедри	07.06.22
9	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	22.06.2022

Студент _____ Олена ЗАВЕРТЯЄВА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Олександр ІВАНОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 7 розділів:

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області, а також проводиться опитування зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 7 сторінок;

– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних,формалізація задачі, розробка фізичного проекту, приводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи, вибір мови програмування. Складається з 18 сторінок;

– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методами «чорного» та «білого» ящика. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 10 сторінок;

– висновки. Складається з 1 сторінки;

– список літератури – включає в себе бібліографічний список використаної літератури. Складає 2 сторінки;

– додатки – містить код програми. Кількість таблиць: 3 штуки. Кількість рисунків: 34 штуки.

Ключові слова: готель, менеджмент, ефективна система, бронювання, пошук.

ЗМІСТ

Перелік умовних познак, символів, скорочень і термінів

БД – база данних

Бронювання - це попереднє замовлення місць і номерів у готелі.

SDK (від англ. software Development Kit) — набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи (програмної або програмно-апаратної).

Графічний інтерфейс користувача (ГІК, англ. GUI, Graphical user interface) — тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

ВСТУП

Ця робота спрямована на програмне моделювання високоефективної системи керування міні-готелем, яка суттєво відрізняється від того, що пропонують сучасні аналоги таких систем, оскільки вона має функціонал, котрий включає в себе всі необхідні методи для опрацювання інформації про готель.

Мета роботи полягає у створенні кросплатформенного мобільного додатку, який допоможе зручно та ефективно вести облік роботи готелю: створення БД номерів готелю, редагування існуючої БД, створення/редагування інформації про бронювання, формування звітів про прибуток готелю.

Матеріали та методи. У двадцять першому столітті без засобів автоматизації важко уявити будь-яку сферу діяльності. Автоматизація служить одним із напрямів науково-технічного прогресу, що дозволяє звільнити людину від участі в процесах обробки та застосування інформації, або значно зменшують рівень даної участі. У сучасному світі використання спеціалізованих додатків у різних сферах послуг стає дедалі популярнішим. Практично у кожного підприємства є свої додатки, сервіси або сайти, які дозволяють ефективно вести облік роботи підприємства, взаємодіїти з клієнтами та надавати важливу інформацію. Від наявності вищезазначених сервісів залежить якість роботи підприємства. Тому створюємий додаток повинен гармонійно поєднувати в собі всі функції, які потрібні для продуктивного менеджменту готеля, також цей доддаток має бути добре оптимізован для роботи на різних пристоях.

Практичне призначення розроблюваного ПЗ полягає в тому, щоб створити систему, яка спростить роботу адміністратора готелю.

1. ЗБІР ТА АНАЛІЗ ВИМОГ

Метою встановлення вимог є надання розгорнутого визначення функціональних - а також не функціональних вимог, які учасники проєкту очікують затвердити в системі, що реалізується та розгортається.

Методи збору вимог до програмного забезпечення

При вивченні предметної області та збору вимог про майбутній об’єкт інформатизації використовується методологія системного аналізу. На цій стадії розробники повинні уточнити межі вивчення функціональних вимог до програми, вхідні та вихідні дані для функціонування програми та визначити коло користувачів майбутньої програми.

Метод інтерв'ю – вербально-комунікативний метод опитування, який полягає у проведенні розмови між дослідником (інтерв’юером) і суб'єктом (респондентом) за заздалегідь розробленим планом. При цьому інтерв'юер ставить запитання, що передбачені планом, організовує та спрямовує спілкування з кожною окремою людиною і фіксує отримані відповіді згідно з інструкцією. Запис відповідей респондента може проводитися дослідником особисто або механічно за допомогою записуючих пристроїв на різні носії інформації. На відміну від бесіди, в якій респонденти і дослідник є активними співрозмовниками, при інтерв'юванні питання побудовані в певній послідовності і їх задає виключно дослідник (інтерв'юер), а респондент лише відповідає на них. Інтерв'юер може спостерігати за поведінкою опитуваного, що значно полегшує інтерпретацію отриманих даних.

Прототипування (prototyping) – це найбільш часто використовуваний сучасний метод виявлення вимог. Програмні прототипи конструюються для візуалізації системи або її частини для замовників з метою отримання їх відгуків.

Еволюційний прототип (evolutionary), який зберігається після виявлення вимог і використовується для створення кінцевого програмного продукту. Еволюційний прототип націлений на прискорення поставок товарів. Як правило, він концентрується на ясно викладених вимогах, тому першу версію продукту можна надати замовнику досить швидко (хоча її функціональні можливості, як правило, неповні).

Для збору та аналізу вимог були використані такі методи, як метод інтер’ю та метод еволюційного прототипування. Комбінування методу інтер’ю та еволюційного прототипування дозволяє швидко та зручно визначити вимоги та розроблювати програму крок за кроком, доки всі вимоги не булуть реалізовані та узгодженні з замовником.

Після виконання збору та аналізу вимог, здобута інформація була розглянута та проаналізована. На основі зроблених висновків було розроблено бізнес-процес додатку. Бізнес-процес для роботи додатку «Менеджмент готелю» виглядає наступним чином:

Система оброблює дані згідно запитів користувача, зберігає внесені змінни, формує звіти.

Бізнес-процес є завершеним, коли користувач виконав всі необхідні дії та отримав повідомлення про їх успішне виконнання.

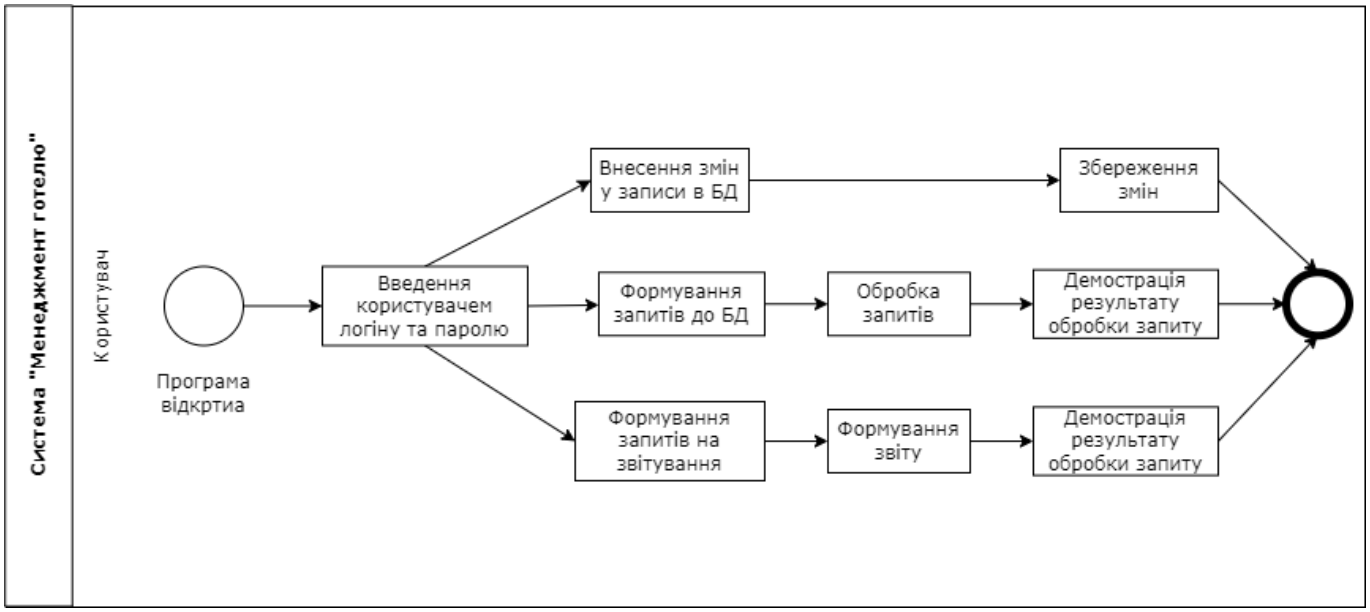

Рисунок 1.1 – Бізнес-процес «Система "Менеджмент готелю"»

1.1 Опис аналогів

1.1.1 ALICE

ALICE має як додатки для окремих відділів (для персоналу, гостя та консьєржа), так і комплексний пакет, який ви можете використовувати для підвищення ефективності роботи на своєму об'єкті.

Завдяки ALICE Staff користувачі можуть розраховувати на такі функції, як внутрішній обмін повідомленнями, щоб підтримувати зв'язок із співробітниками, а також можливість керувати квитками та призначати їх. ALICE Concierge корисний для тих, хто працює на стійці реєстрації, дозволяючи співробітникам інтегрувати вашу систему управління нерухомістю для доступу до важливої інформації про гостей та дозволяючи відслідковувати всі запити в рамках однієї системи, включаючи інциденти, бронювання, транспортні послуги, будильник тощо.

У цьому додатку є гостьовий компонент, де гості можуть озвучувати свої потреби та бажання, надаючи їм оновлення статусу в реальному часі та сповіщення про хід виконання конкретної послуги. Метрики відстеження гостей також доступні за всіма запитами та бронюваннями.

ALICE доступний як для операційних систем iOS, так і для Android.

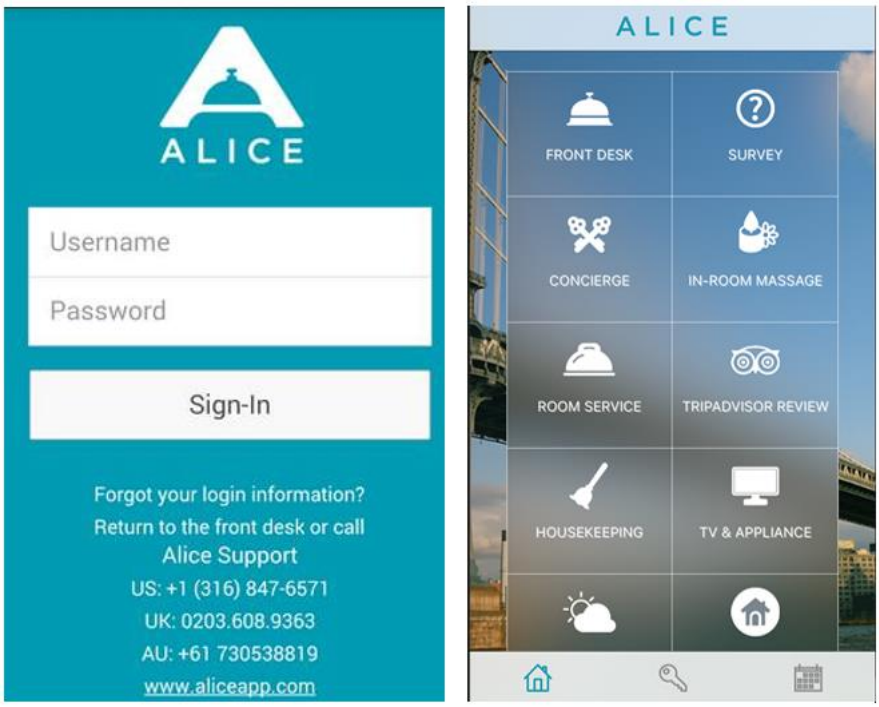

Рисунок 1.2 – Приклад роботи програми «ALICE»

1.1.2 Nano Hotel Booking

Даний додаток для керування готелем - чудовий варіант для менеджерів готелів, які хочуть стежити за своїм готелем. Nano Hotel Booking виконує більшу частину тієї ж роботи, що й традиційні настільні рішення, включаючи календар, який інформує користувачів про заповненість та вакансії номерів, а також про те, скільки кімнат буде звільнено за встановлений період часу.

Також можна виділити споруджувані приміщення. Крім того, користувачі мають можливість визначати сезонні ціни і записувати побутову техніку в кожній кімнаті, таку як холодильник, телевізор або телефон.

Nano Hotel Booking and Property Management сумісний із системами iOS.

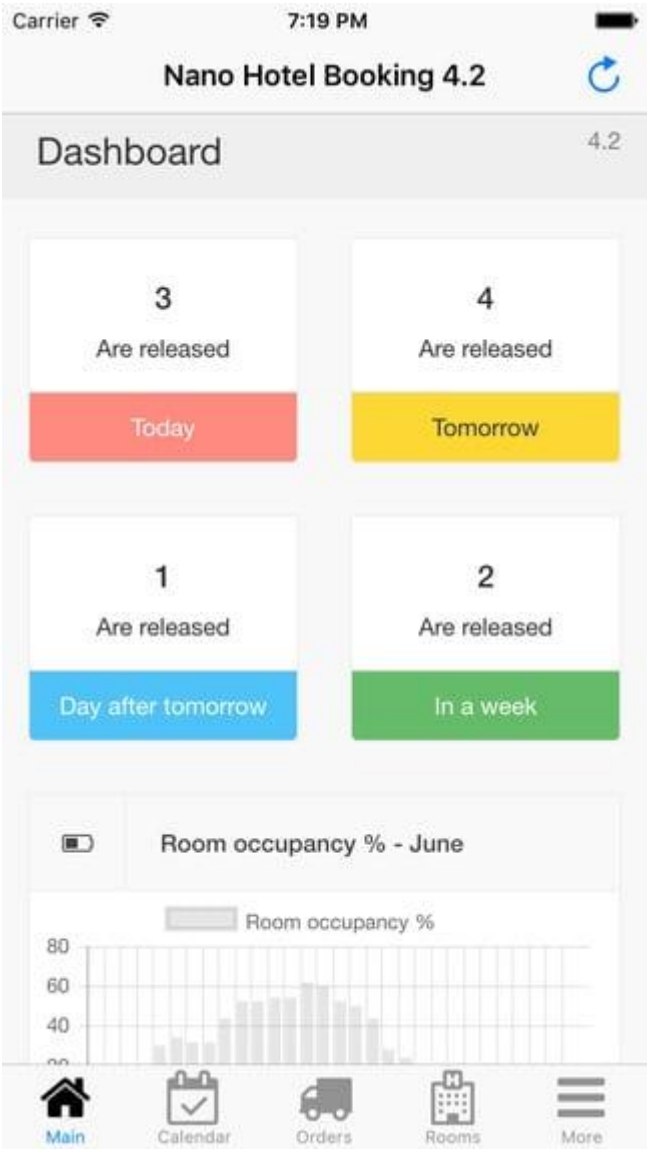

Рисунок 1.3 – Приклад роботи програми «Nano Hotel Booking»

1.1.3 Roombler

Roombler не тільки є механізмом бронювання, але й дозволяє необмеженій кількісті користувачів працювати з додатком, що зазвичай обмежується традиційним управлінням нерухомістю у сфері гостинності. Користувачі можуть перетягувати бронювання для раптової зміни дати бронювання, додавати кілька номерів до одного бронювання та додавати збори та платежі до бронювання.

Користувачі також можуть інтегрувати онлайн-сайти про подорожі, такі як Booking.com, Expedia та Airbnb. І якщо ці функції недостатньо переконливі, Roombler запевняє, що навчання не потрібне, що, як ми сподіваємося, означає, що користувачі не зазнають крутої кривої навчання, яка часто буває з традиційним програмним забезпеченням для керування готелями.

Roombler працює на iOS.

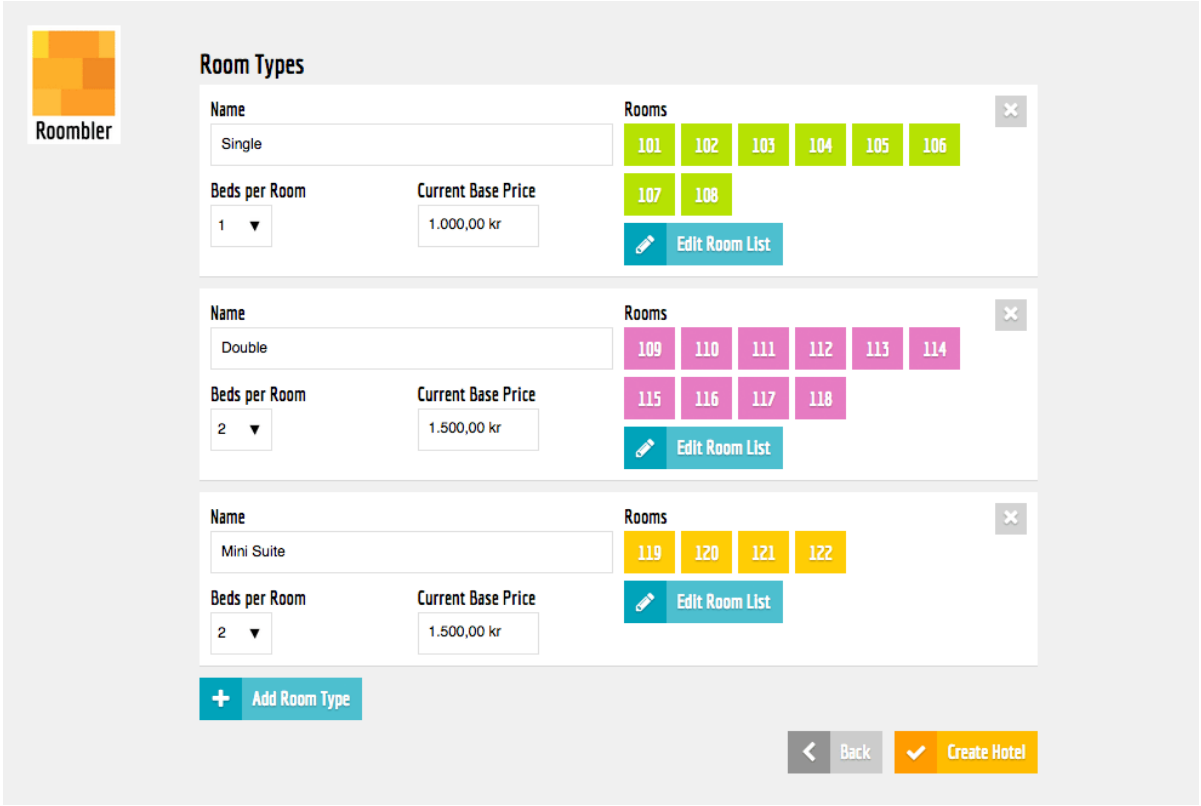

Рисунок 1.4 – Приклад роботи програми «Roombler»

1.1.4 SabeeApp

Для керування готелем або готелем типу «ліжко та сніданок» потрібне програмне забезпечення, яке трохи відрізняється від того, що використовується в гігантських готелях. SabeeApp – це хмарне програмне забезпечення, «спеціально

розроблене для незалежних готелів та курортів, квартир, гостьових будинків та особливих потреб хостелів».

Він пропонує комплексну систему управління, яка включає стійку реєстрації для бронювання та прибирання, механізм бронювання та спосіб приймання безпечних платежів.

SabeeApp автоматично оновлюватиме тарифи та доступність вашого готелю в режимі реального часу. Оскільки це веб-інтерфейс, немає потреби в установці, обслуговуванні або резервному копіюванні.

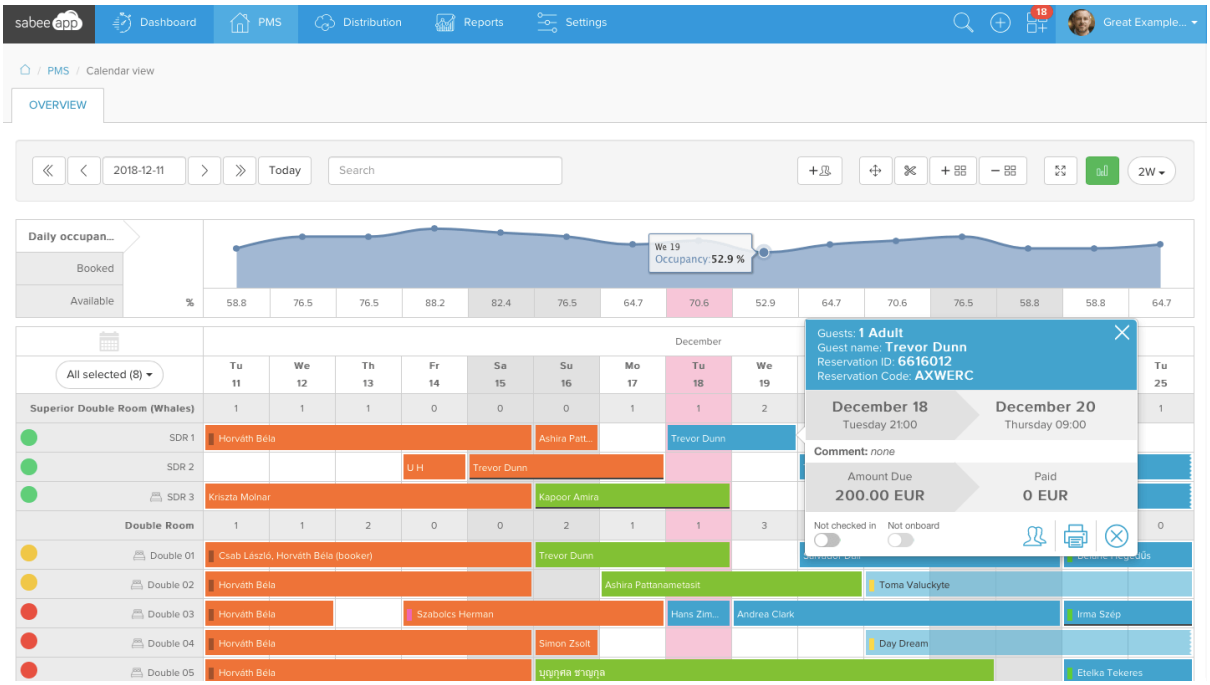

Рисунок 1.5 – Приклад роботи програми «SabeeApp»

1.1.5 Tourismart

Tourismart має можливості будь-якого провідного програмного забезпечення для управління нерухомістю у сфері гостинності, за винятком того, що цей варіант може поміститися у вас на долоні. Можливості включають спілкування в режимі реального часу, коли готелі можуть розмовляти з гостями, щоб полегшити обслуговування номерів. Також доступний профіль гостей, якій може допомогти готелярам відслідковувати переваги своїх гостей і навіть прогнозувати їх майбутні покупки або запити.

Управління репутацією та push-сповіщення доповнюють видатні функції Tourismart, дозволяючи вам надавати більш якісні послуги, запобігати поганим

відгукам та попереджати гостей про потенційні пропозиції та знижки у вашому готелі, наприклад про знижки у спа-салоні або половинну знижку на напої у барі.

Touismart працює як з Android, так і з iOS.

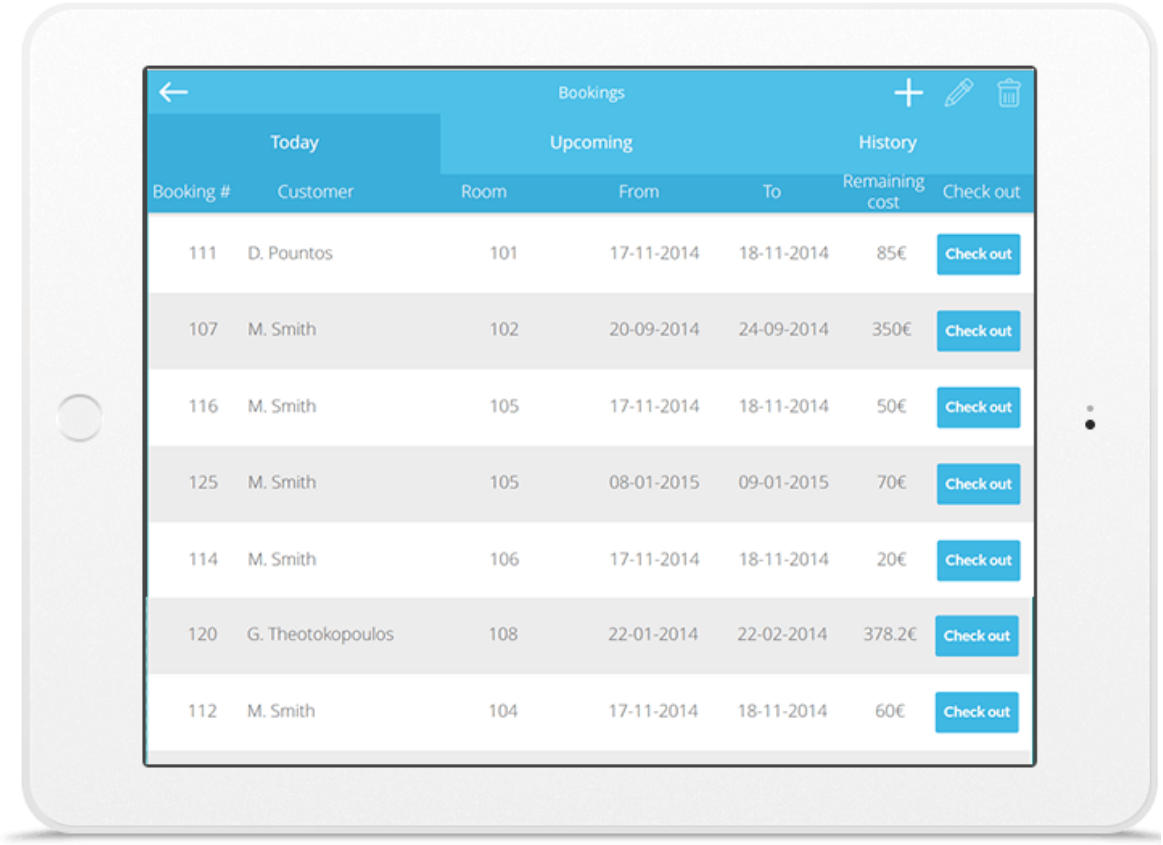

Рисунок 1.6 – Приклад роботи програми «Tourismart»

1.2 Висновки на основі аналогів

Аналоги не відповідають вимогам замовника а саме:

1. Перший аналог не має перевірки на коррекність введення даних.
2. Другий аналог не надає змогу формувати звіти про прибутки підприємства;
3. Недоліком третього аналогу є те, що цей додаток не є кросплатформеним.
4. Аналог №4 працює тільки при наявності інтернет з’єднання.
5. Аналог №5 не надає змогу формувати звіти про прибутки підприємства. В ньому відсутня перевірка на коректність введенних даних.

2. ПРОЕКТУВАННЯ

2.1 Зовнішнє проектування

2.1.1 Функціональне призначення

Програма реалізує високоефективну модель керування готелем, яка поєднує в собі необхідні функції для покращення та спрощення роботи менеджера готелю.

2.1.2 Експлуатаційне призначення

Даний додаток допомагає зручно вести документацію готельного бізнесу, спрощу процес бронювання, пошуку вільних номерів та у зручному вигляді формує фінансові звіти.

2.1.3 Функціональні вимоги

Завданням даного ПЗ є спрощення роботи менеджера готелю.

Дії які повинно реалізовувати ПЗ:

- реєстрація користувача, використовуючи e-mail;
- вхід користувача в систему за допомогою логіну і паролю;
- формування бази даних, яка містить в собі інформацію про номери готелю, бронювання та прибутки готелю;
- внесення змін до існуючої БД;
- пошук вільних номерів готелю згідно заданих дат та заданої кількості людей;
- формування звіту про прибуток згідно заданих умов(конкретний кімната готелю, певний період часу);
- формування звіту про загальний прибуток.

2.1.4 Вхідні дані:

1. userEmail – електронна адреса використовується при реєстрації аккауту та подальшого входу в аккаут. Дані повинні відповідати формату *@*.*, можуть бути використані такі символи A-Z, a-z, 0-9 і “-” (example@ex.com);

2. userPassword - пароль використовується при реєстрації аккауту та подальшого входу в аккаут. Можуть бути використані такі символи A-Z, a-z, 0-9, пароль не може починатися з цифри.
3. dateIn – дата заїзду в бронюванні. Формат даних – дата (dd/mm/yyyy);
4. dateOut - дата виїзду в бронюванні. Формат даних – дата (dd/mm/yyyy);
5. nameOfPerson – ім’я та фаміля клієнта. Допустимі такі символи A-Z, a-z
6. countRooms – кількість кімнат у номері. Формат даних – число, більше 0;
7. numberOfPersons – кількість людей, які можуть бути розміщені у номері. Формат даних – число, більше 0;
8. costOfRoom – вартість номеру на добу. Формат даних – число, більше 0;
9. numberOfFloor – поверх готелю, на якому знаходиться кімната. Формат даних – число, більше 0;
10. balcony – наявність балкону в номері. Формат даних – checkbox(приймає значення true/false);
11. dateOfBooking – масив об’єктів, об’єкт містить інформацію про людину яка бронює номер та дати бронювання.

2.1.5 Вихідні дані

Результат звіту подається на єкран користувачу у вигляді текстової інформації, де вказана вся інформація згідно типу звіту.

Результат обробки запиту до БД щодо бронювання номеру готелю на певні дати подається користувачу на екран у вигляді повідомлення про успішне бронювання. У випадку, коли бронювання здійснити неможливо, користувач отримує відповідне повідомлення.

2.1.6 Опис зовнішнього інформаційного середовища.

Дані userEmail, userPassword, nameOfPerson, countRooms, costOfRoom, numberOfPersons, numberOfFloor вводяться з клавіатури з контролем введення. dateIn та dateOut обираються за допомогою спеціального елементу календар, з

контролем введення. Balcony – елемент checkbox, приймає значення true, якщо номер має балкон, false – без балкону.

Для функціонування програми необхідно наявність інтернет-підключення.

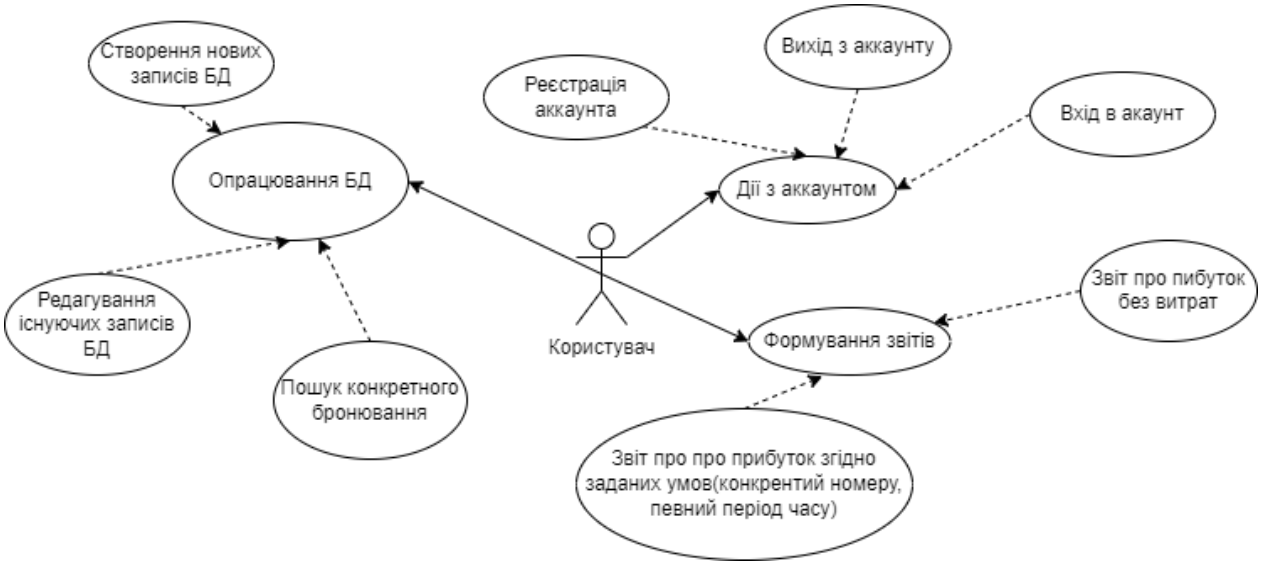

Рисунок 2.1. – Діаграма прецедентів

2.2 Внутрішнє програмування

2.2.1 Статична архітектура системи

Мобільний додаток можна розбити на 2 логічні групи екранів взвємодії:

- початкова сторінка, з якої відбувається перехід до 2 екранів (екран авторизації та екран регістрації);
- головна сторінка програми з якої відбувається перехід до 9 екранів (екран додавання кімнати, екран редагування інформації про кімнати, екран видалення кімнати, екран додавання бронювання, екран видалення конкретного бронювання, екран пошуку бронювання, екран формування звіту про загальний прибуток, екран формування звіту про прибуток за певний період часу, екран формування звіту про прибуток конкретного номеру готелю);

Схема екранів системи та їх взаємодії представлена на рисунку 2.2.

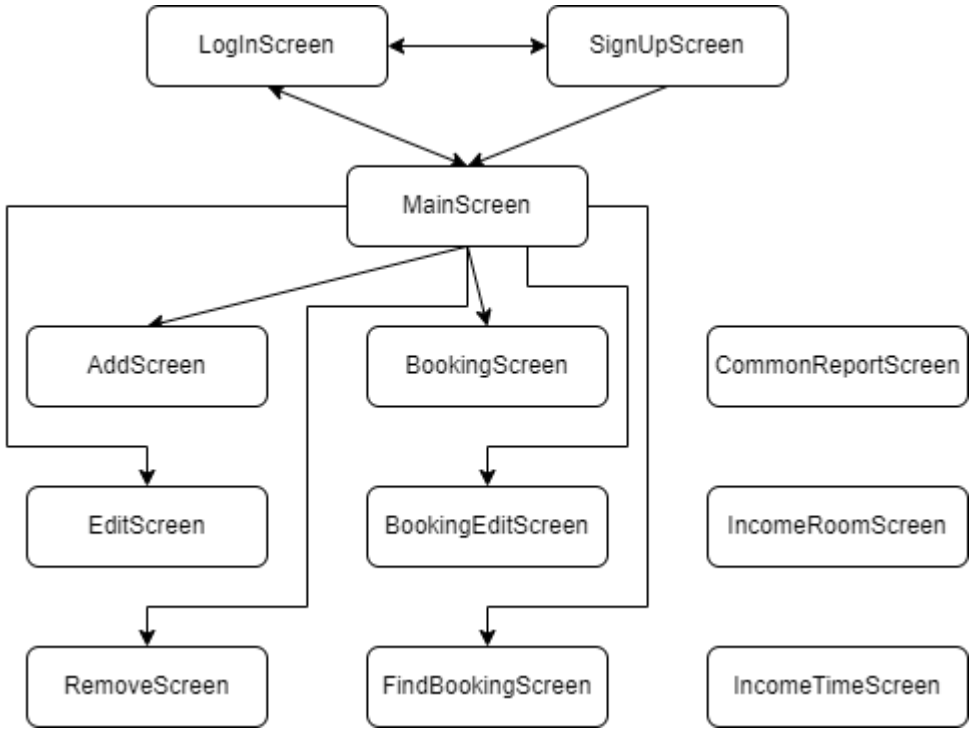

Рисунок 2.2 – Схема взаємодії екранів додатку

Кожен екран мобільного додатку реалізується як окремий компонент.

Компонент «LogInScreen» є початковою сторінкою додатку. На єкрані здійснюється автризація користувача, також є можливість перейти до компоненту «SingUpScreen» для реєстрації користувача.

Компонент «SingUpScreen» – сторінка реєстрації. На єкрані здійснюється реєстрація користувача, також є можливість перейти до компоненту «LogInScreen» для авторизації користувача.

Компонент «MainScreen» є головною сторінкою додатку. З цієї сторінки здійснюється перехід на сторінки, для виконання основних дії системи.

Компонент «AddScreen» – сторінка для додавання нової кімнати до БД.

Компонент «EditScreen» – сторінка для редагування інформації існуючої кімнати у БД.

Компонент «RemoveScreen» – видалення існуючого номеру з БД.

Компонент «BookingScreen» – сторінка для додавання нового бронування до БД.

Компонент «EditBookingScreen» – сторінка для видалення існуючого бронювання кімнати у БД.

Компонент «FindBookingScreen» – пошук інформації про бронювання згідно імені клієнту.

Компонент «CommonReportScreen» – сторінка для формування загального звіту про прибуток.

Компонент «IncomeRoomScreen» – сторінка для формування звіту про прибуток певного номеру.

Компонент «IncomeTimeScreen» – сторінка для формування звіту про прибуток за певний період часу.

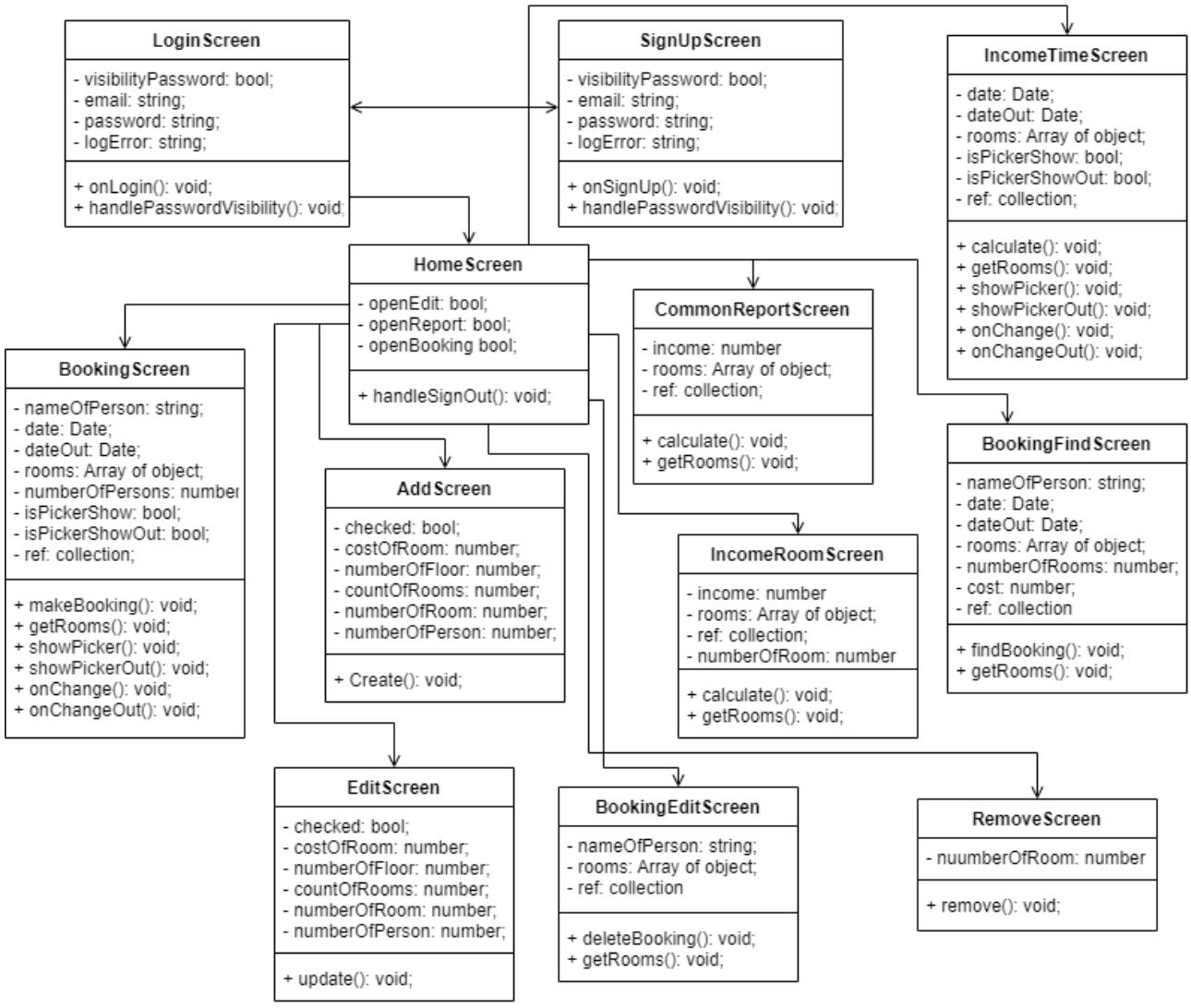

Рисунок 2.3 – Загальна діаграма классів

2.2.2. Структура хмарнного сховища даних

Реєстрація/авторизація користувача реалізується за допомогою сервіса Firebase Authentication. Для реєстрації/авторизації використовується логін і пароль.

База даних готелю створена за допомогою сервусу Firebase Firestore Database. БД являє собою колекцію документів Rooms. Кожен документ має свій унікальний ідентифікатор, який складається з номеру кімнати готелю та назви власника.

Поля документа:

- balcony: поле типу bool зберігає інформацію про наявність балкону в кімнаті;
- costOfRoom: поле типу number, зберігає інформацію про вартість кімнати за день перебування;
- countOfRooms: поле типу number, зберігає інформацію про кількість кімнат в номері;
- dateOfBooking: поле типу array, зберігає інформацію про дати бронювання та клієнта, який бронює;
- numberOfFloor: поле типу number, зберігає інформацію про поверх;
- numbersOfPerson: поле типу number, зберігає інформацію про кількіть людей, які можуть проживати в кімнати;
- numberOfRooms: поле типу number, зберігає інформацію порядковий номер кімнати;
- owner: поле типу string, зберігає інформацію про власника готелю. Власник готелю визначається за значенням логіну(email) з якого ведеться робота з БД.

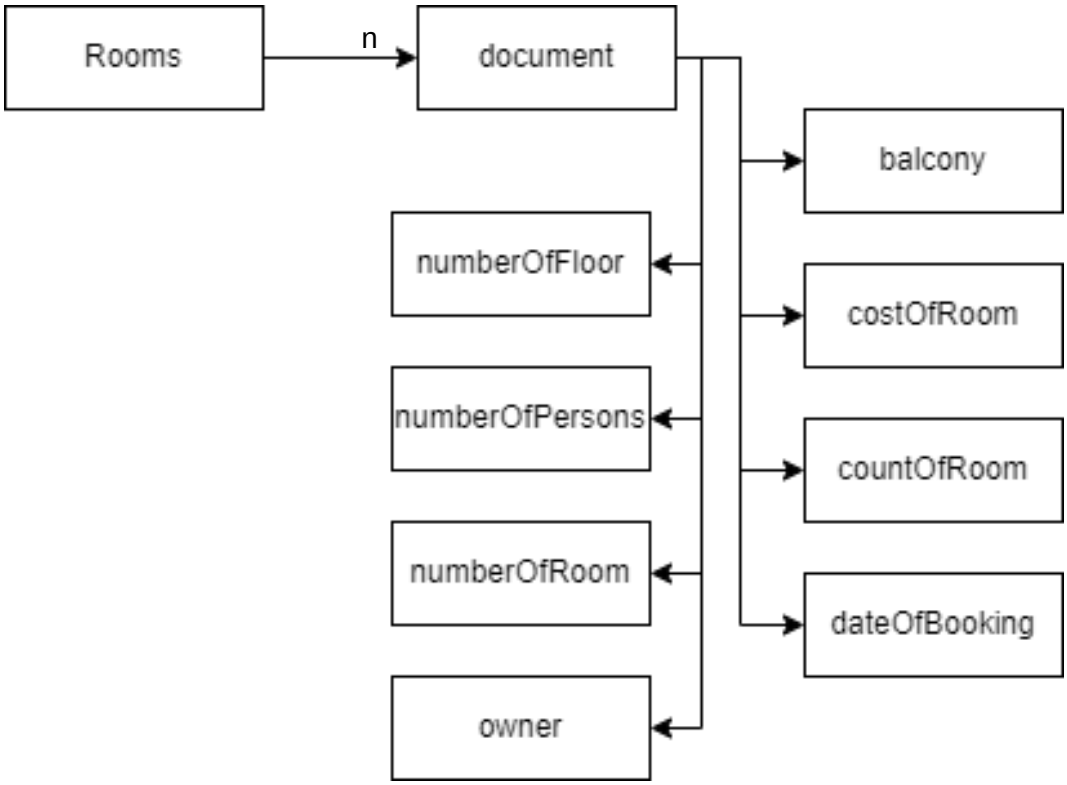

Рисунок 2.4 – Структура БД

Система взаємодіє з БД через запити до БД. Запити здійснюють додавання, редагування, видалення та пошук інформації.

2.2.3. Проектування інтерфейсу користувача

Інтерфейс користувача (UI) — це спосіб взаємодії людини та комп'ютера. UI включає екрани, клавіатуру, мишу та зовнішній вигляд робочого столу. Це також спосіб, за допомогою якого користувач взаємодіє з програмою або веб-сайтом.

Зростаюча залежність багатьох компаній від веб-додатків і мобільних додатків привела до того, що багато компаній надають перевагу інтерфейсу користувача, намагаючись покращити загальний досвід роботи користувачів.

Типи інтерфейсів користувача:

- графічний інтерфейс користувача (GUI);
- інтерфейс командного рядка (CLI);
- кероване меню інтерфейс користувача;
- сенсорний інтерфейс користувача;
- голосовий інтерфейс користувача (VUI);
- інтерфейс користувача на основі форми;

- інтерфейс користувача природною мовою.

UI та UX

Про інтерфейс користувача часто говорять у поєднанні з користувацьким досвідом (UX), який може включати естетичний вигляд пристрою, час відгуку та вміст, який представляється користувачеві в контексті інтерфейсу користувача. Обидва терміни підпадають під концепцію взаємодії людини та комп'ютера (HCI), яка є сферою дослідження, зосередженою на створенні комп'ютерних технологій та взаємодії між людьми та всіма формами IT-дизайну.

Графічні інтерфейси користувача

Елементи графічного інтерфейсу включають такі речі, як вікна, випадаюче меню, кнопки, смуги прокрутки та значки. Зі збільшенням використання мультимедіа як частини графічного інтерфейсу, звук, голос, відео та віртуальна реальність все частіше стають графічним інтерфейсом багатьох програм.

Мобільні інтерфейси

Наростаюча популярність мобільних додатків також вплинула на інтерфейс користувача, що призвело до того, що називається мобільним інтерфейсом користувача. Інтерфейс мобільного зв'язку спеціально спрямований на створення зручних інтерактивних інтерфейсів на невеликих екранах смартфонів і планшетів і вдосконалення спеціальних функцій, таких як сенсорне керування.

Мобільний інтерфейс користувача — це дисплей або екран на мобільному пристрої. Це простір, де користувачі можуть взаємодіяти з тим, що на екрані – від кнопок меню до текстових полів (і все між ними, залежно від того, чи можуть користувачі натискати, прокручувати, гортати, вводити чи просто бачити це).

Більшість цих взаємодій користувачів відбувається на основі дотиків і відбувається на барвистих сенсорних дисплеях, які переповнені взаємодіями високого рівня. Звичайно, основні принципи дизайну мобільного інтерфейсу користувача відрізняються від принципів традиційного інтерфейсу для настільних комп'ютерів. Дії та інформація мають бути великими, чіткими та простими.

Найважливіщі принципи мобільного інтерфейсу користуача:

1. Приоритезація вмісту

Тривалість уваги людини дійсно коротка. Тому надзвичайно важливо привернути увагу користувачів протягом перших кількох секунд взаємодії з вашими продуктами.

Розробляючи продукт, треба зводити елементи інтерфейсу до мінімуму. Простий дизайн – це те, що дозволяє користувачеві зацікавитися продуктами та легко користуватися ними.

Необхідно відображати лише важливий вміст і функції, які потрібні користувачеві. Додатковий вміст має бути доступним через меню. Списки меню, як короткі, так і довгі, повинні мати прогресивне розкриття та просту термінологію.

2. Інтуїтивно зрозуміла навігація

Користувачі повинні інтуїтивно мати можливість переміщатися по вашому додатку за допомогою чітких шляхів і мати можливість виконувати всі основні завдання без будь-яких пояснень. Переглядаючи додаток, користувач завжди повинен знати, де він знаходиться, не замислюючись, як він туди потрапив або що йому робити далі.

3. Цільові розміри сенсорного екрана

Люди взаємодіють із сенсорними екранами за допомогою пальців. Важливою частиною є зробити елементи інтерфейсу достатньо великими, щоб охопити ці дії.

Люди користуються великими пальцями частіше, ніж будь-яким іншим пальцем під час користування смартфоном. Маленькі сенсорні цілі можуть викликати проблеми використання, оскільки вони вимагають більшої точності та схильні до помилок.

Важливий не тільки розмір цілі, а й відстань між цілями. Якщо кнопки дій розташовані занадто близько одна від одної, користувач ризикуєте зробити небажані дії, що призведуть до помилкової роботи програми.

4. Мінімізуйте введення даних

Необхідно зменшити кількість введення необхідних текстів, скорочуючи форми, видаляючи непотрібні поля та використовуючи параметри «Запам’ятати мене» для подальшого використання. Полегшити роботу користувачів, можна замінивши введення з клавіатури автоматизованими функціями(визначення геолокації, автопідбір даних, використання спеціальних елементів).

Варіанти відображення клавіатури встановлюються залежно від необхідних даних, наприклад, для телефонних номерів встановлюється цифрова клавіатура для швидшого введення.

5. Елементи інтерфейсу повинні бути чітко видимими

Важливо мати достатній контраст між вмістом і фоном у ваших дизайнах, щоб вони були розбірливими в будь-якій обстановці, навіть на вулиці при сонячному світлі.

6. Правильно підібраний колір

Правильне використання кольору важливо для зручної роботи користувача, тому колір відіграє провідну роль у дизайні інтерфейсу. Він стимулює всі органи чуття, доставляючи повідомлення миттєво, як жоден інший спосіб спілкування.

При розробці додатку було використано такі основні кольори:

- Колір заднього фону додатку: #95B8D1
- Колір тексту та кнопок: #13315C
- Колір тексту кнопок та полей для вводу даних: #E8F1F2

На основі сказаного вище були розроблені ескізи головних екранів додатку.

Ескізи мобільного додатку мають схематичний характер та служать для зручності та розуміння розробником системи, що проектується.

Структура мобільного додатку складається з наступних екранів:

- екран авторизації користувача;
- екран реєстрації користувача;
- головниий екран;
- екран додавання кімнати до БД;

– екран видалення кімнати з БД;
– екран редагування кімнати в БД;
– екран додавання бронювання до БД;
– екран видалення бронювання з БД;
– екран пошук бронювання в БД;
– екран загального звіту про прибуток;
– екран звіту про прибуток за вказаний період часу;
– екран звіту про прибуток певної кімнати;

Схематичне зображення екрану авторизації користувача подано на рисунку 2.5

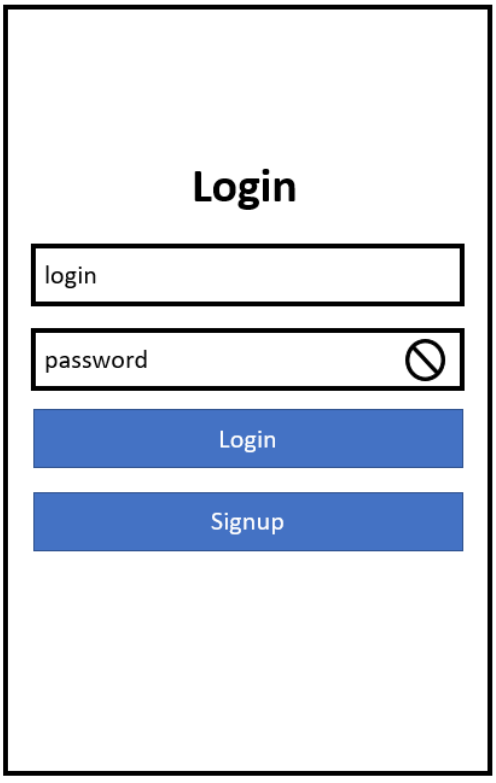

Рисунок 2.5 – Ескіз екрану авторизації користувача

Схематичне зображення екрану реєстрації користувача подано на рисунку 2.6

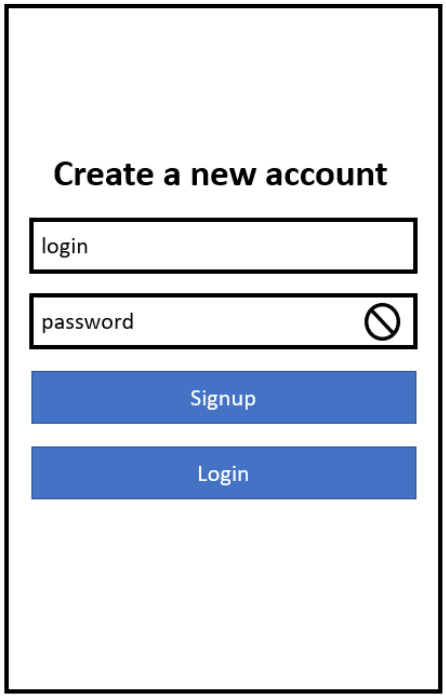

Рисунок 2.6 – Ескіз екрану реєстрації користувача

Схематичне зображення головного екрану подано на рисунку 2.7

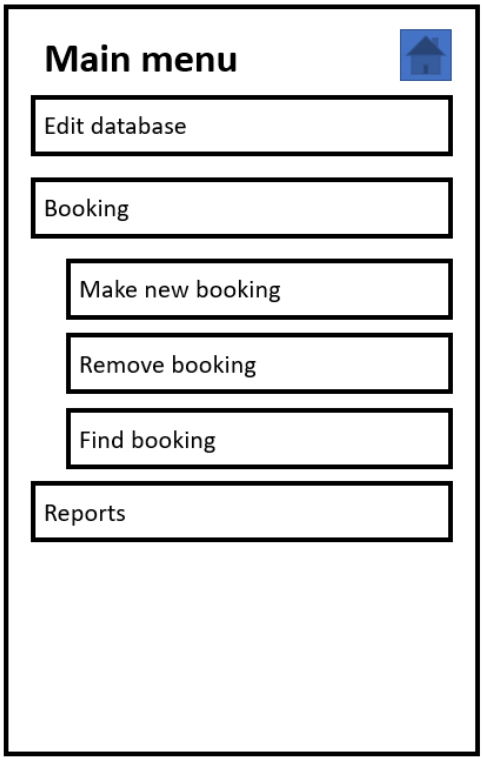

Рисунок 2.7 – Ескіз головного екрану

Схематичне зображення екрану додавання кімнати до БД подано на рисунку 2.8

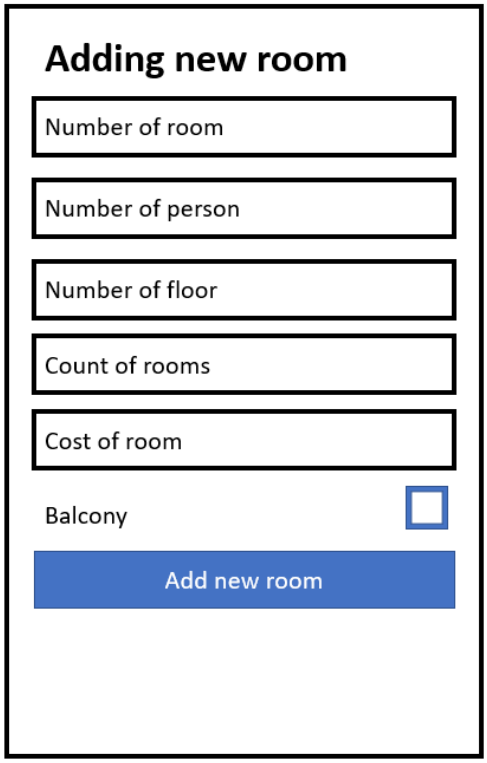

Рисунок 2.8 – Ескіз екрану додавання кімнати до БД

Схематичне зображення екрану редагування кімнати в БД подано на рисунку 2.9

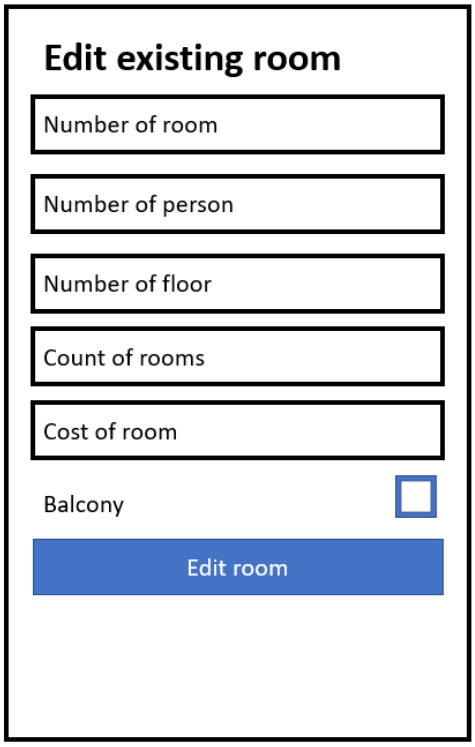

Рисунок 2.9 – Ескіз екрану редагування кімнати в БД

Схематичне зображення екрану видалення кімнати з БД подано на рисунку 2.10

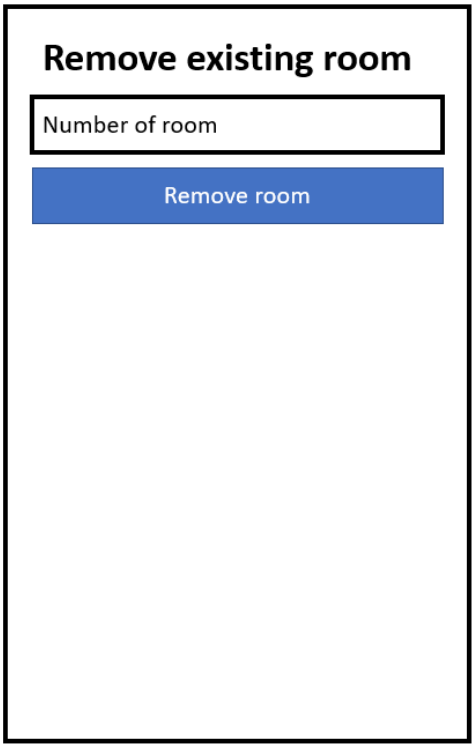

Рисунок 2.10 – Ескіз екрану видалення кімнати з БД

Схематичне зображення екрану додавання бронювання до БД подано на рисунку 2.11

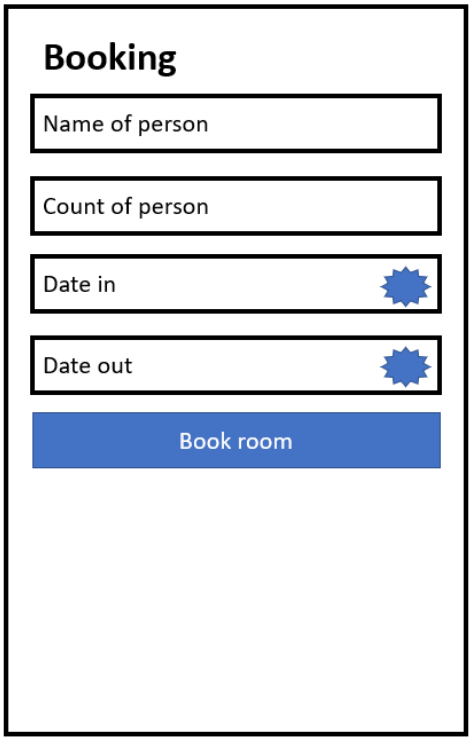

Рисунок 2.12 – Ескіз екрану додавання бронювання до БД

Схематичне зображення екрану видалення бронювання з БД подано на рисунку 2.13

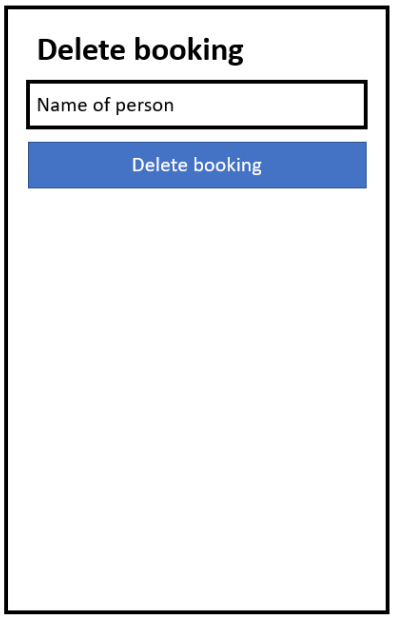

Рисунок 2.13 – Ескіз екрану видалення бронювання з БД

Схематичне зображення екрану пошук бронювання у БД подано на рисунку 2.14

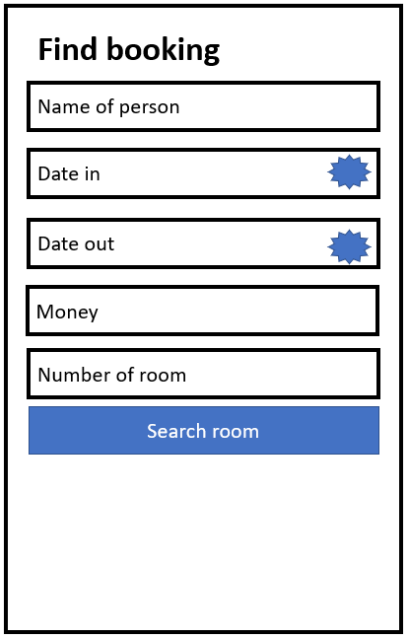

Рисунок 2.14 – Ескіз екрану пошук бронювання у БД

Схематичне зображення екрану звіту про прибуток за вказаний період часу подано на рисунку 2.15

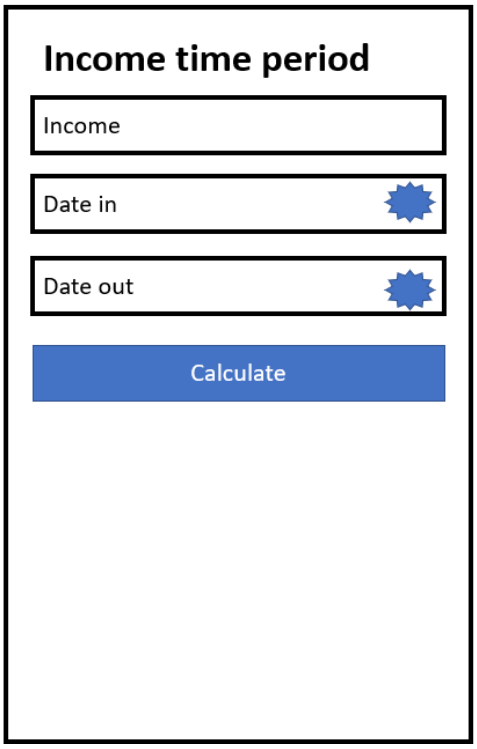

Рисунок 2.15 – Ескіз екрану звіту про прибуток за вказаний період часу

Схематичне зображення екрану звіту про прибуток за певну кімнату подано на рисунку 2.16

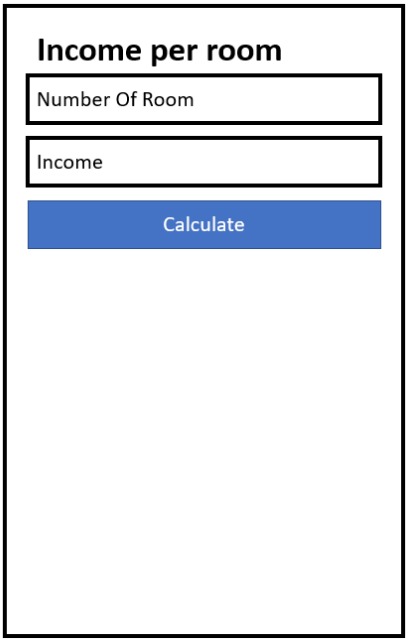

Рисунок 2.16 – Ескіз екрану про прибуток за певну кімнату

Схематичне зображення екрану про загальний прибуток подано на рисунку 2.17

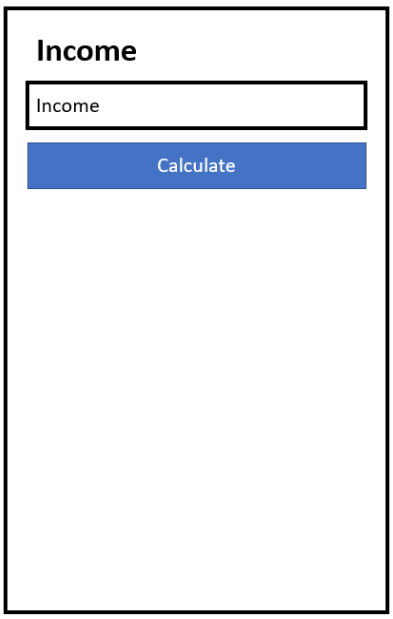

Рисунок 2.17 – Ескіз екрану про загальний прибуток

2.2.4 Проектування динаміки системи

Діаграма діяльності мобільного додатку при додаванні нової кімнати у БД зображена на рисунку 2.18

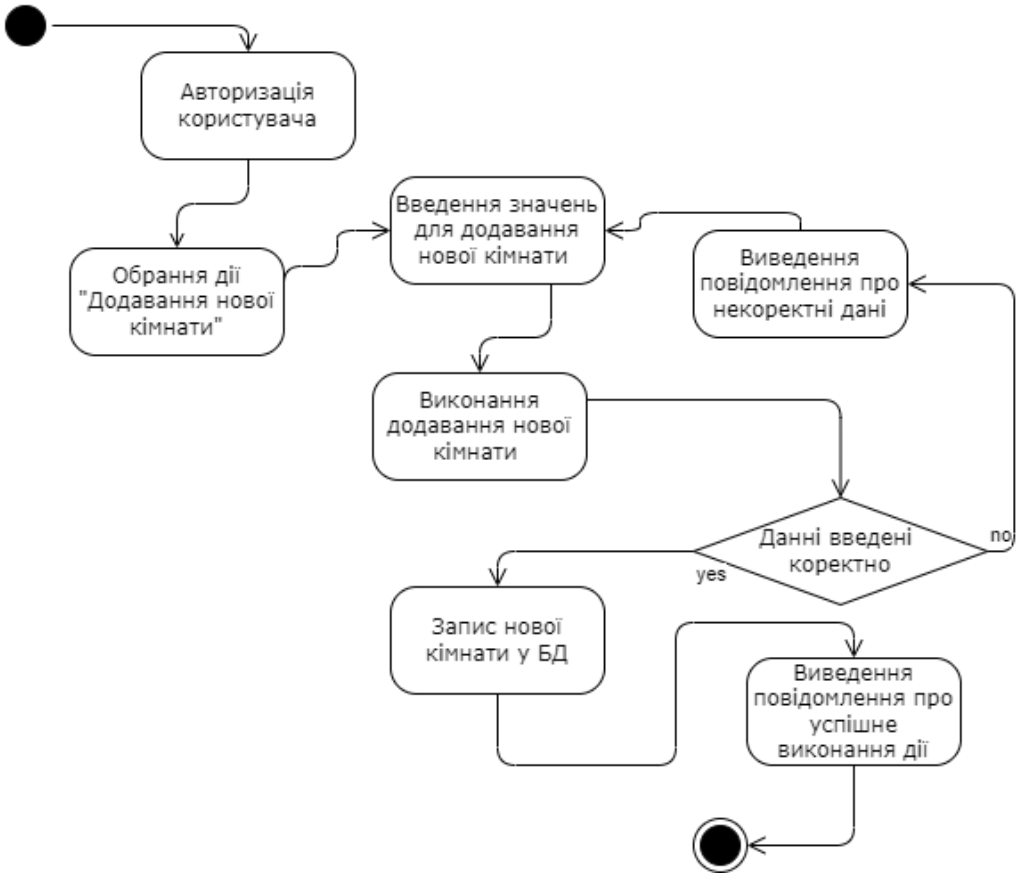

Рисунок 2.18 – Діаграма діяльності мобільного додатку при додаванні нової кімнати у БД

Початковим станом мобільної системи є запуск мобільного додатку. Після завантаження користувачу відображається екран авторизації. Після авторизації користувач потрапляє на головну сторінку, де відображаються пункти меню. При натисканні на основні пункти меню, відображаються підпункти, при натисканні на які, користувач переходить на сторінки, де виконує основні функції системи. Всі функції виконуються за схожими алгоритмами: користувач обирає дію, користувач вводить дані, дані перевіряються на коректність, якщо дані введені вірно, виконується функція і демострується результат, якщо дані невірні на екран виводиться повідомлення про це.

2.2.5 Проектування системи на фізичному рівні

Передбачається, що систеема буде реалізована на мові javascript. Кожен модуль представляє собою окремий екран.

Діаграма артефактів зображена на рисунку 2.19

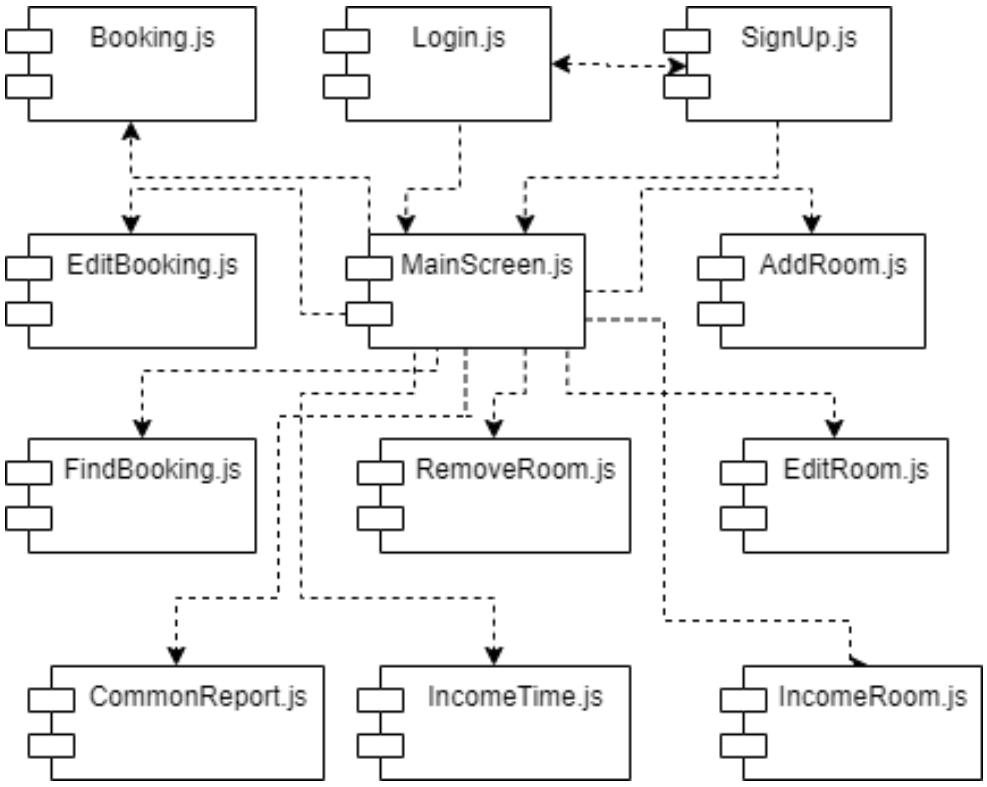

Рисунок 2.19 – Діаграма артефактів

3. РОЗРОБКА ПРОГРАМИ

3.1 Вибір мови програмування

Мобільний додаток — це тип програмного забезпечення, призначеного для запуску на мобільному пристрої, такому як смартфон або планшетний комп’ютер. Мобільні програми часто служать для надання користувачам послуг, подібних до послуг, доступних на ПК. Програми, як правило, є невеликими окремими програмними блоками з обмеженою функціональністю.

Типи мобільних додатків зображені на рисунку 3.1.1

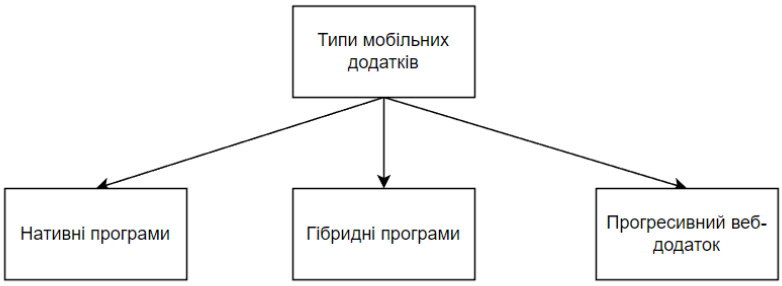

Рисунок 3.1 – Типи мобільних додатків

Нативні програми

Нативна розробка додатків кодується мовою, яка ізначально підтримується конкретною операційною системою мобільних пристроїв. Це використовується, якщо ви створюєте програму спеціально для Apple App Store або Google Play Store.

Це ідеально підходить для програм із високим рівнем налаштування, яким потрібно використовувати власні компоненти кожного пристрою. Нативна розробка чудово підходить для ігрових додатків, програм віртуальної реальності та додатків з великою графікою. Але один код не працюватиме на обох платформах.

Гібридні програми

Гібридні програми призначені для кросплатформної розробки. Вони закодовані однією мовою, яка може працювати на кількох як на iOS, так і на Android.

Це прискорює терміни розробки, оскільки вам доведеться кодувати лише один раз, а не двічі. Порівняно з нативною розробкою, ви трохи втратите гнучкість

щодо того, що ви можете робити з гібридними програмами. Але це добре для переважної більшості розробників.

Програми PWA

PWA (прогресивна веб-програма) — це легка програма, яка запускається в URL-адресі веб-переглядача пристрою. Він виглядає і відчувається як мобільний додаток, але не поставляється на пристрої.

Розробники, які мають досвід веб-розробки, можуть легко створювати PWA. Ви вже повинні бути знайомі з мовами кодування, які використовуються в процесі розробки.

3.1.1 React Native

React Native базується на бібліотеці JavaScript React від Facebook, яка відображає інтерфейс нативної платформи. React Native — чудовий вибір для простих додатків, де API мають чіткий міст між платформами. React Native зосереджен на інтерфейсі значною мірою, завдяки чому інтерфейс дуже гнучкий. Функція швидкого оновлення дозволяє розробникам бачити зміни, внесені в код за лічені секунди. Різноманітність вбудованих компонентів і API, керованих спільнотою, заощаджує час розробки.

3.1.2 Xamarin

Xamarin пропонує чудову перевірку під час компіляції, що дозволяє розробникам відчувати менше помилок під час виконання. Xamarin також спрощує розробку додатка, схожого на нативну, зі зручним інтерфейсом і елементами керування. Xamarin підтримує зв’язування з рідними бібліотеками та використовує API, що стосуються платформи. Дозволяє ділитися до 90% вашого коду між платформами для «Напишіть один раз, запустіть будь-де». Спеціальні платформні елементи інтерфейсу користувача з Xamarin.Forms забезпечують узгоджений вигляд у різних ОС.

3.1.3 Flutter

Flutter має сильних прихильників у кросплатформній спільноті, оскільки його універсальний SDK для розробки мобільних додатків. Фреймворк Flutter, як і

мова Dart, існує недовго, тому він досі не зовсім стабільний і зрілий. Flutter легко вивчати, легко створювати та налагоджувати, і він повністю скомпільований. У Flutter є повний набір віджетів у Material Design від Google і в стилі Apple із пакетом Cupertino. Функція Hot Reload дозволяє розробникам бачити зміни, внесені в код за секунди.

3.1.4 Qt

Qt використовується для розробки кросплатформних програм і графічних інтерфейсів користувача (GUI), які працюють на більшості мобільних, настільних або вбудованих платформ. Qt підтримує такі компілятори, як Visual Studio, компілятор GCC і PHP через розширення. Інструменти Qt включають IDE Qt Creator для C++ і Qt Quick, яка включає декларативну мову сценаріїв під назвою QML. Інші функції Qt включають аналіз XML, аналіз JSON, доступ до бази даних SQL та керування потоками. Це добре розроблена програма C++ GUI. Код скомпільовано у власні двійкові файли, які працюють на повній швидкості (не потрібно використовувати віртуальну машину). Qt має величезну базу користувачів і документацію. Отримати відповіді на запитання легко.

3.1.5 Cordova

Apache Cordova є форком з відкритим вихідним кодом проекту PhoneGap, який був придбаний Adobe у 2011 році. Пізніше Adobe припинив випуск PhoneGap у 2020 році. Програми, створені за допомогою Cordova, є гібридними, що означає, що вони не є ані по-справжньому нативними, ані суто веб-орієнтованими. Cordova дозволяє обгортати HTML, CSS та JavaScript залежно від платформи пристрою. Додатки Cordova покладаються на прив’язки API, що відповідають стандартам, для доступу до можливостей кожного пристрою, таких як дані, датчики, стан мережі тощо. Сумісний з різними сторонніми плагінами та API. Також підтримує спеціальні плагіни. Синтаксис командного рядка простий у освоєнні та застосуванні. Легко налаштовувати, розробляти, додавати платформи та розгортати програми для кількох платформ.

Спираючись на перераховані вище переваги кожної з мов програмування, мій вибір зупинився на React Native, тому що ця технологія дозволить швидко і легко створити заплановану систему.

3.2 Опис алгоритмічних структур

Однією з головних зачач створення даного додатку є створення ефективних алгоритмів додавання та пошуку бронювання та подальше збереження інформації про бронювання. Для візуалізації алгоритмів обран метод каскадного предствалення алгоритмів. Каскадне представлення алгоритму додавання нового бронювання представлено у таблиці 3.1

Таблиця 3.1 – Кскадне представлення алгоритму додавання нового бронювання

Задача	Підзадачі	
Додавання нового бронювання у БД	Введення данних	Введеня даних з клавіатури
		Вибір дати за допомогою елемента «календар»
	Перевірка даних на коректність	Перевірка на пусті рядки
		Перевірка на початок значення з «0»
		Перевірка на правильно встановленні дати
	Виведення повідомлення про некоректні данні	
	Пошук вільного номеру на вказані дати	Перевірка вільного місця на момент заїзду
		Перевірка вільного місця з моменту заїзду до момента віїзду
	Внесення бронювання до БД	Занесення записів про клієнта з моменту заїзду до моменту виїзду
	Виведення повідомлення	Виведення повідомлення про успішне виконання

Каскадне представлення алгоритму пошуку бронювання у таблиці 3.2.

Таблиця 3.2 – Каскадне представлення алгоритму пошуку бронювання

Задача	Підзадачі	
Пошук бронювання у БД	Введення данних	Введеня даних з клавіатури
	Перевірка даних на коректність	Перевірка на пусті рядки
	Виведення повідомлення про некоректні данні	
	Пошук бронювання	Пошук імені в бд бронювання.
	Підрахунок вартості перебування у готелі	Пошук дати заїзду
		Пошук дати виїзду
		Підрахунок кількості днів
		Підрахунок вартості перебування
	Виведення результатів на екран	Вартість перебування
		Номер на яке є бронювання
		Дати перебуванняя в готелі

4. ТЕСТУВАННЯ ТА ВІДЛАГОДЖЕННЯ

Тестування програмного забезпечення – це процес оцінки та перевірки того, що програмний продукт або додаток виконує те, що він повинен робити. Переваги тестування включають запобігання помилкам, зниження витрат на розробку та підвищення продуктивності.

Існує багато різних типів тестів програмного забезпечення, кожен з яких має певні цілі та стратегії:

- Приймальне тестування: перевірка, чи вся система працює належним чином.
- Інтеграційне тестування: переконання, що програмні компоненти або функції працюють разом.
- Модульне тестування: перевірка того, що кожен програмний блок працює належним чином. Одиниця — це найменший компонент програми, який можна перевірити.
- Функціональне тестування: перевірка функцій шляхом емуляції бізнес-сценаріїв на основі функціональних вимог. Тестування чорного ящика є поширеним способом перевірки функцій.
- Тестування продуктивності: перевірка роботи програмного забезпечення при різних навантаженнях. Навантажувальне тестування, наприклад, використовується для оцінки продуктивності в умовах реального навантаження.
- Регресійне тестування: перевірка, чи нові функції не порушують чи погіршують функціональність. Тестування на працездатність можна використовувати для перевірки меню, функцій і команд на поверхневому рівні, коли немає часу на повну регресійну перевірку.
- Стрес-тестування: перевірка, наскільки сильного навантаження може витримати система, перш ніж вийти з ладу. Вважається різновидом нефункціонального тестування.

– Тестування зручності використання: перевірка того, наскільки добре клієнт може використовувати систему або веб-додаток для виконання завдання.
– У випадку тестування системи треба поділити методи тестування програмної системи на такі категорії:
– Тестування чорною скринькою(без доступа до тексту програми);
– Тестування білою скринькою(з доступом до тексту програми).

4.2 Тестування методами білої скриньки

Для тестування функції IncomeTime() були обрані метод покриття операторів, та метод покриття умов.

```
const validateDate = () => {
    if (date > dateOut) {//1
      createTwoButtonAlert(
        "Invalid data",
        "Date in cannot be bigger than date out"
      );
      return false;//2
    } else {
      return true;//3
    }
  };

  const calculate = () => {
    if (validateDate()) {
      let income = 0, //4
        allIncome = 0,
        countDays = 0,
        cost = 0;
      let indexIn = 0,//5
        indexOut = 0;
      let dateIn =
        date.getDate() + "/" + (date.getMonth() + 1) + "/" +
date.getFullYear();//6
      let dateOUT =
        dateOut.getDate() +
        "/" +
        (dateOut.getMonth() + 1) +
        "/" +
        dateOut.getFullYear();//7
      for (let i = 0; i < rooms[0].dateOfBooking.length; i++) {//8
        if (rooms[0].dateOfBooking[i].dateCode == dateIn) {//9
          indexIn = i;//10
```

```
            for (let j = indexIn; j < rooms[0].dateOfBooking.length; j++) {//11
              if (rooms[0].dateOfBooking[j].dateCode == dateOUT) {//12
                indexOut = j;//13
              }
            }
          }
        }
        rooms.forEach((element) => {//14
          if (element.owner == global.owner) {//15
            cost = element.costOfRoom;//16
            for (let k = indexIn; k < indexOut; k++) {//17
              if (element.dateOfBooking[k].nameOfPerson !== "") {//18
                countDays++;//19
              }
            }
            income = countDays * cost;//20
          }
          allIncome = allIncome + income;//21
          countDays = 0;//22
          income = 0;//23
        });
        setIncome(allIncome);//24
      }
    };
```

Функція обчислює прибуток за певний період часу.

Заголовок const Calculate = ()

В явному вигляді функція не приймає змінних.

У явному вигляді функція не повертає нічого.

№	Вхідні дані	Вихідні дані
1	Date in: 1/1/2022 Date out: 30/6/2022	Income: 12000
2	Date in: 8/6/2022 Date out: 12/6/2022	Income: 10000
3	Date in: 7/6/2022 Date out: 7/6/2022	Income: 0
4	Date in: 7/6/2022 Date out: 3/6/2022	Error “Invalid date”

Тестування методом покриття операторів подано у табл. 4.1:

Таблиця 4.1 Тестування методом покриття операторів

№ теста	Номер оператора																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Тестування методом покриття умов подано у табл. 4.2:

Таблиця 4.2 – Тестування методом покриття умов

Умова / Тест	`date > dateOut`		`rooms[0].dateOfBooking[i].dateCode == dateIn`		`rooms[0].dateOfBooking[j].dateCode == dateOUT`		`element.owner == global.owner`		`element.dateOfBooking[k].nameOfPerson !== ""`	
	+	-	+	-	+	-	+	-	+	-
1		*	*		*	*	*		*	
2		*	*		*		*		*	
3		*	*		*		*			*
4	*			*		*		*		*

4.3 Тестування чорною скринькою

Основний метод цього виду тестування – це припущення про помилку

Ситуація №1. Невірно введений логін або пароль при авторизації користувача. Результат реакції програми подано на рисунку 4.1

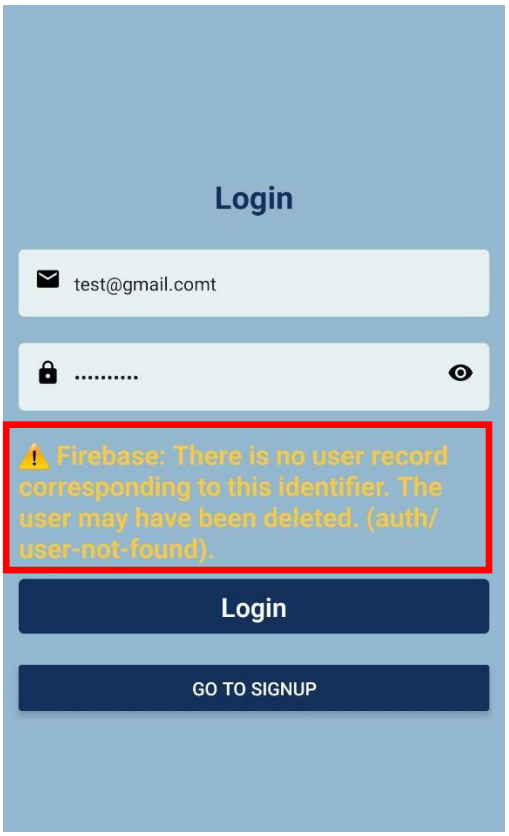

Рисунок 4.1 – Результат реакції програми на введення невірного логіну

Ситуація №2. Користувач намагається зареєструватися, використовуючи вже зареєстрований email. Результат реакції програми подано на рисунку 4.2

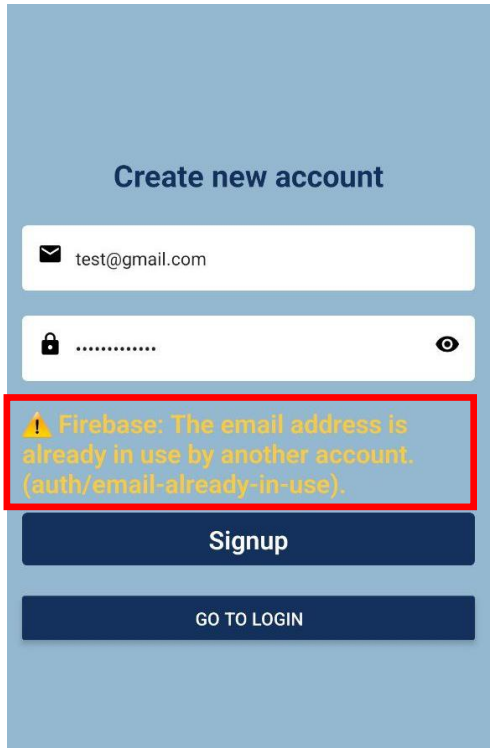

Рисунок 4.2 – Результат реакції програми

Ситуація №3. Додавання нової кімнати готелю. Результат реакції програми подано на рисунку 4.3

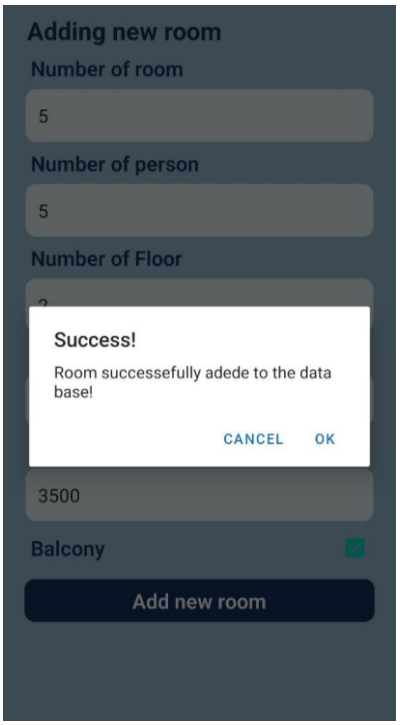

Рисунок 4.3 – Результат реакції програми

Ситуація №3. Додавання нового бронювання. Результат реакції програми подано на рисунку 4.4

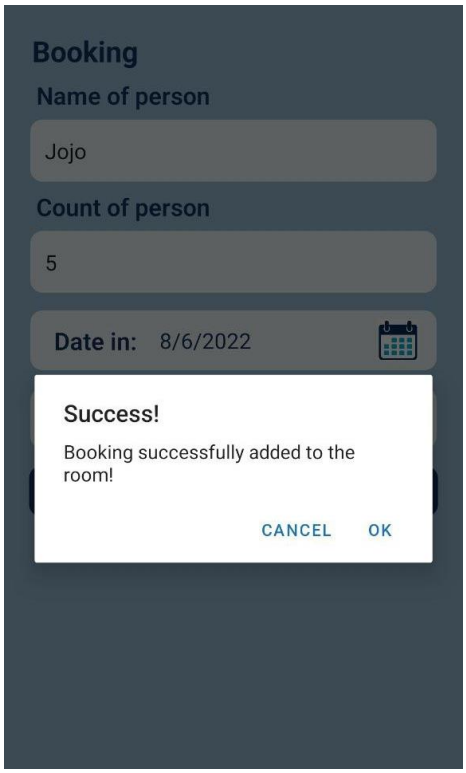

Рисунок 4.4 – Результат реакції програми

Ситуація №5. Пошук бронювання. Результат реакції програми подано на рисунку 4.5

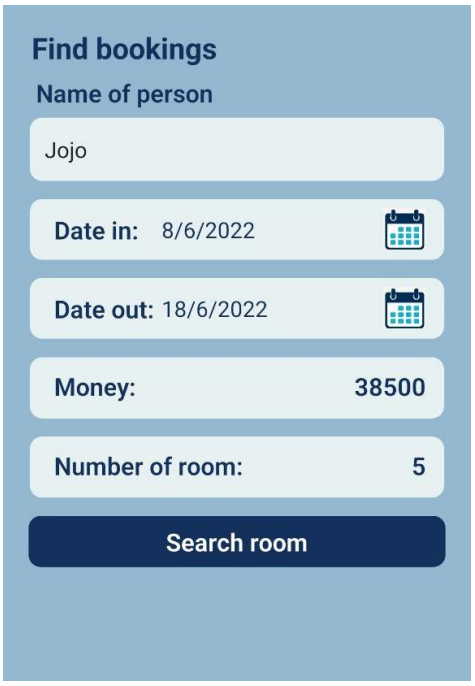

Рисунок 4.5 – Результат реакції програми

Ситуація №6. Розрахунок прибутку за певний період часу. Результат реакції програми подано на рисунку 4.6

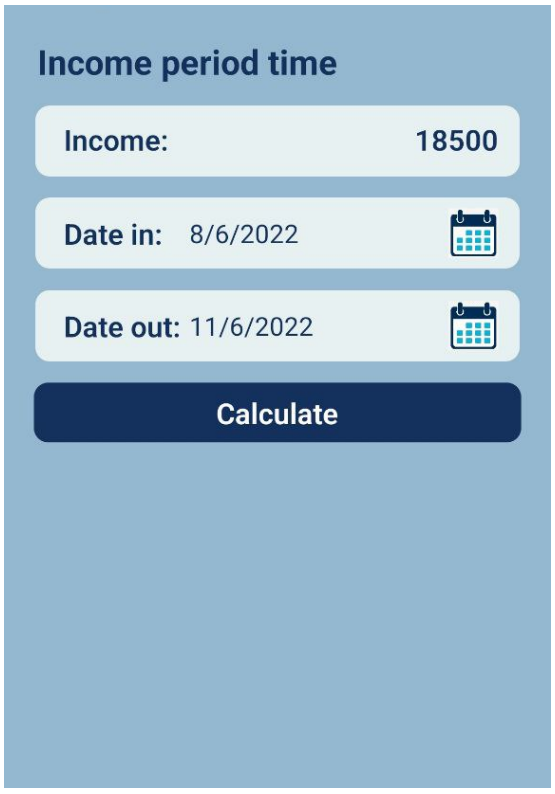

Рисунок 4.6 – Результат реакції програми

Ситуація №7. Редагування інформації про кімнату з пустими рядками. Результат реакції програми подано на рисунку 4.7

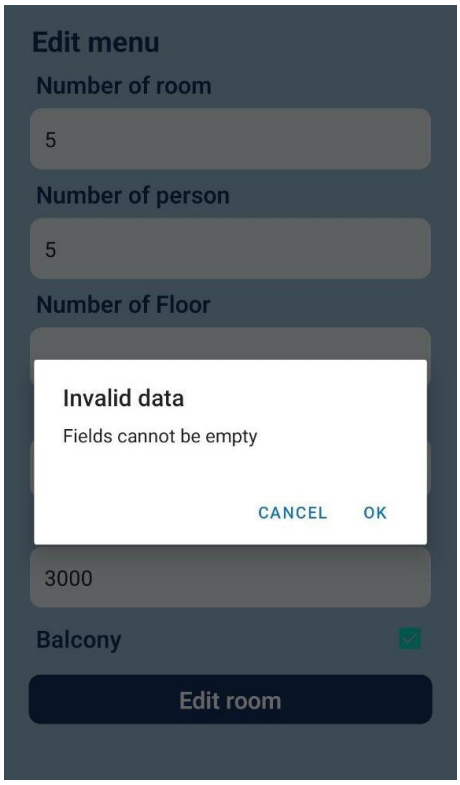

Рисунок 4.7 – Результат реакції програми

Ситуація №8. Редагування інформації про кімнату з невалідними даними. Результат реакції програми подано на рисунку 4.8

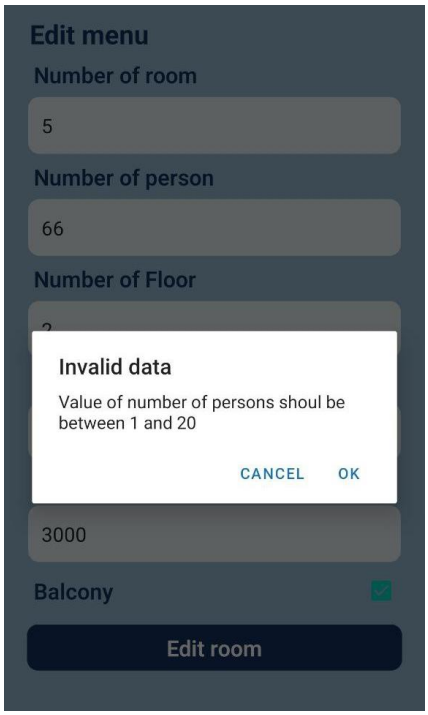

Рисунок 4.8 – Результат реакції програми

4.3 Налагодження програми

Налагодження – це процес виявлення та видалення наявних і потенційних помилок у програмному коді, які можуть призвести до несподіваної поведінки або збою. Щоб запобігти некоректній роботі програмного забезпечення або системи, налагодження використовується для пошуку та усунення помилок або дефектів. Коли різні підсистеми або модулі тісно пов’язані, налагодження стає важчим, оскільки будь-яка зміна в одному модулі може призвести до появи нових помилок в іншому. Іноді на налагодження програми потрібно більше часу, ніж на її кодування.

Протокол налагодження програми наведений у таблиці 4.3:

Таблиця 4.3. – Протокол налагодження програми

Опис помилки	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення
Невірний підрахунок прибутку за кімнату.	При виконуванні циклу неправильно зазначуєтся вартість за номер за добу	Надання змінній конкретної вартості	Створення локальної змінної для збереження вартості номеру за добу
Невірне додавання бронювання	Додавання здійснюється не на номер власника аккаунту, а на інший аккаунт	Додати перевірку на власника готелю, зберігаючи його як окреме поле документу.	Додання умовного оператора `if (element.owner == global.owner)`
Збереження даних, які переважають гранично допустимі норми	Додання нереальних даних(наприклад 1001 поверх кімнати)	Додати перевірку на гранично допустимі дані	Додавання функції валідації даних

ЗАГАЛЬНІ ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

В результаті виконання дипломної роботи був створений кросплатформений додаток для менеджменту готелю.

Було спроектовано та розроблено базу данних для збереження всієї необхідної інформації. Для ефективного опрацювання даних БД системою, були розроблені алгоритми додавання/редагування та пошуку бронювання і номерів в готелі. Також було створено зручну систему звітування про прибуток готелю.

Для реалізації кросплатформеного додатку було використовано технологію React Native. БД системи було створена на платформі Firebase, яка зручно дозволяє зберігати колекції документів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Аналоги мобільних додатків для менеджменту готелю [Електронний ресурс] – https://platforms.su/articles/2611 Дата звернення: 05.05.2022
2. What is user interface (UI)? [Електронний ресурс] – https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI Дата звернення: 08.05.2022
3. ReactJS CRUD application use firebase Firestore database [Електронний ресурс] – https://infotechwar.com/it-solutions/reactjs-crud-application-using-firebase-firestore-database/ Дата звернення: 15.05.2022
4. What Are the Different Types of Mobile Apps? And How Do You Choose? [Електронний ресурс] - https://clevertap.com/blog/types-of-mobile-apps/ Дата звернення: 23.05.2022
5. Top Most Popular Programming Languages for Mobile App Development [Електронний ресурс] - https://fireart.studio/blog/top-most-popular-programming-languages-for-mobile-app-development/ Дата звернення: 26.05.2022
6. 14 Programming Languages for Mobile App Development [Електронний ресурс] - https://buildfire.com/programming-languages-for-mobile-app-development/ Дата звернення: 27.05.2022
7. Top 12+ React Datepickers to Use in 2022 [Електронний ресурс] - https://flatlogic.com/blog/top-12-react-datepickers-to-use-in-2021/ Дата звернення: 31.05.2022
8. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с Дата звернення: 05.05.2022 – 06.05.2022
9. Інженерія програмного забезпечення [Текст]: навчальний посібник / В. М. Горячкін, О. В. Горбова, О. С. Куроп’ятник; Український державний

університет науки і технологій. – Дніп-ро, 2022. – 140 с. Дата звернення: 05.05.2022 – 06.05.2022

ДОДАТОК А

Текст програми

Buttons.js

```
import React from "react";
import { StyleSheet, Pressable, Text } from "react-
native";

const Button = ({
  title,
  backgroundColor = "#000",
  titleColor = "#fff",
  titleSize = 14,
  onPress,
  width = "100%",
  containerStyle,
}) => {
  return (
    <Pressable
      onPress={onPress}
      style={(args) => {
        if (args.pressed) {
          return [
            styles.base,
            {
              opacity: 0.5,
              backgroundColor,
              width,
            },
            containerStyle,
          ];
        }

        return [
          styles.base,
          {
            opacity: 1,
            backgroundColor,
            width,
          },
          containerStyle,
        ];
      }}
    >
      <Text style={[styles.text, { color: titleColor,
fontSize: titleSize }]}>
        {title}
      </Text>
    </Pressable>
  );
};

const styles = StyleSheet.create({
  text: {
    fontWeight: "600",
  },
  base: {
    alignItems: "center",
    justifyContent: "center",
    minHeight: 42,
    borderRadius: 4,
    paddingHorizontal: 12,
  },
});

export default Button;
```

ErrorMessage.js

```
import React from "react";
import { StyleSheet, Text } from "react-native";

const ErrorMessage = ({ error, visible }) => {
  if (!error || !visible) {
    return null;
  }

  return <Text style={styles.errorText}>⚠
{error}</Text>;
};

const styles = StyleSheet.create({
  errorText: {
    color: "#fdca40",
    fontSize: 20,
    marginBottom: 10,
    fontWeight: "600",
  },
});

export default ErrorMessage;
```

IconButton.js

```
import React from "react";
import { Pressable, StyleSheet } from "react-native";
import { AntDesign } from "@expo/vector-icons";
```

```
const IconButton = ({ color, size, onPress, name })
=> {
  return (
    <Pressable
      style={(args) => {
        if (args.pressed) {
          return [
            styles.base,
            {
              opacity: 0.5,
              backgroundColor: "transparent",
            },
          ];
        }

        return [styles.base, { opacity: 1,
backgroundColor: "transparent" }];
      }}
      onPress={onPress}
    >
      <AntDesign name={name} size={size}
color={color} />
    </Pressable>
  );
};

const styles = StyleSheet.create({
  base: {
    alignItems: "center",
    justifyContent: "center",
  },
});

export default IconButton;
```

index.js

```
import IconButton from "./IconButton";
import Button from "./Buttons";
import ErrorMessage from "./ErrorMessage";
import InputField from "./InputField";

export { IconButton, Button, ErrorMessage, InputField
};
```

InputField.js

```
import React from "react";
import { View, StyleSheet, TextInput,
TouchableOpacity } from "react-native";
import { MaterialCommunityIcons } from "@expo/vector-
icons";

const InputField = ({
  leftIcon,
  iconColor = "#000",
  rightIcon,
  inputStyle,
  containerStyle,
  placeholderTextColor = "#444",
  handlePasswordVisibility,
  ...rest
}) => {
  return (
    <View style={[styles.container, containerStyle]}>
      {leftIcon ? (
        <MaterialCommunityIcons
          name={leftIcon}
          size={20}
          color={iconColor}
          style={styles.leftIcon}
        />
      ) : null}
      <TextInput
        {...rest}
        placeholderTextColor={placeholderTextColor}
        style={[styles.input, inputStyle]}
      />
      {rightIcon ? (
        <TouchableOpacity
onPress={handlePasswordVisibility}>
          <MaterialCommunityIcons
            name={rightIcon}
            size={20}
            color={iconColor}
            style={styles.rightIcon}
          />
        </TouchableOpacity>
      ) : null}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    borderRadius: 4,
    flexDirection: "row",
    padding: 12,
  },
  leftIcon: {
    marginRight: 10,
  },
```

```
  input: {
    flex: 1,
    width: "100%",
    fontSize: 18,
  },
  rightIcon: {
    alignSelf: "center",
    marginLeft: 10,
  },
});

export default InputField;
```

firebase.js

```
// import firebase from "firebase/app";
// import "firebase/auth";
// import Constants from "expo-constants";
import firebase from "firebase/compat/app";
import "firebase/compat/auth";
import "firebase/compat/firestore";

// Initialize Firebase
const firebaseConfig = {
  apiKey: "AIzaSyC9unMstYDFKw2CmzVxGKNz4QWYzC6bAsY",
  authDomain: "hotelmanagement-
344d9.firebaseapp.com",
  projectId: "hotelmanagement-344d9",
  storageBucket: "hotelmanagement-344d9.appspot.com",
  messagingSenderId: "305737607501",
  appId: "1:305737607501:web:fc499f06e32858a50d6f04",
};

let Firebase;

if (firebase.apps.length === 0) {
  Firebase = firebase.initializeApp(firebaseConfig);
}

export default Firebase;
```

firestore.jsx

```
// Import the functions you need from the SDKs you
need
import { initializeApp } from "firebase/app";

import { getFirestore } from "firebase/firestore";
// TODO: Add SDKs for Firebase products that you want
to use
//
https://firebase.google.com/docs/web/setup#available-
libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyC9unMstYDFKw2CmzVxGKNz4QWYzC6bAsY",
  authDomain: "hotelmanagement-
344d9.firebaseapp.com",
  projectId: "hotelmanagement-344d9",
  storageBucket: "hotelmanagement-344d9.appspot.com",
  messagingSenderId: "305737607501",
  appId: "1:305737607501:web:fc499f06e32858a50d6f04",
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

export const db = getFirestore(app);
```

AddScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useState } from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, InputField, IconButton } from
"../components";
import Firebase from "../config/firebase";
import { Checkbox } from "react-native-paper";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
```

```
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function AddScreen({ navigation }) {
  //balckony
  const [checked, setChecked] = useState(false);
  const [costOfRoom, setCostOfRoom] = useState("");
  const [numberOfFloor, setNumberOfFloor] =
useState("");
  const [numberOfPersons, setNumberOfPersons] =
useState("");
  const [numberOfRoom, setNumberOfRoom] =
useState("");
  const [countOfRooms, setCountOfRooms] =
useState("");

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  const correctData = () => {
    if (
      costOfRoom === "" ||
      countOfRooms === "" ||
      numberOfFloor === "" ||
      numberOfPersons === "" ||
      numberOfRoom === ""
    ) {
      createTwoButtonAlert("Invalid data", "Fields
cannot be empty");
      return false;
    }
    if (costOfRoom[0] === "0") {
      createTwoButtonAlert(
        "Invalid data",
        "Value of cost of room cannot begining with
zero"
      );
      return false;
    } else {
      if (numberOfFloor[0] === "0") {
        createTwoButtonAlert(
          "Invalid data",
          "Value of number of floor cannot begining
with zero"
        );
        return false;
      } else {
        if (numberOfPersons[0] === "0") {
          createTwoButtonAlert(
            "Invalid data",
            "Value of number of person cannot
begining with zero"
          );
          return false;
        } else {
          if (numberOfRoom[0] === "0") {
            createTwoButtonAlert(
              "Invalid data",
              "Value of number of room cannot
begining with zero"
            );
            return false;
          } else {
            if (countOfRooms[0] === "0") {
              createTwoButtonAlert(
                "Invalid data",
                "Value of count of rooms cannot
begining with zero"
              );
              return false;
            } else {
              return true;
            }
          }
        }
      }
    }
  };

  const validateData = () => {
    if (countOfRooms > 5) {
      createTwoButtonAlert(
        "Invalid data",
        "Value of count of rooms shoul be between 1
and 5"
      );
      return false;
    } else {
```

```
      if (numberOfFloor > 20) {
        createTwoButtonAlert(
          "Invalid data",
          "Value of number of floor shoul be between
1 and 20"
        );
        return false;
      } else {
        if (numberOfPersons > 10) {
          createTwoButtonAlert(
            "Invalid data",
            "Value of number of person shoul be
between 1 and 10"
          );
          return false;
        } else {
          return true;
        }
      }
    }
  };

  function Create() {
    if (correctData() && validateData()) {
      const dateOfBookingEmpty = [];
      let day = 1,
        month = 1;
      for (let i = 1; i < 366; i++) {
        let dateOfBookingRoom = {};
        if (i === 32) {
          month = 2;
          day = 1;
        }
        if (i === 60) {
          month = 3;
          day = 1;
        }
        if (i === 91) {
          month = 4;
          day = 1;
        }
        if (i === 121) {
          month = 5;
          day = 1;
        }
        if (i === 152) {
          month = 6;
          day = 1;
        }
        if (i === 182) {
          month = 7;
          day = 1;
        }
        if (i === 213) {
          month = 8;
          day = 1;
        }
        if (i === 244) {
          month = 9;
          day = 1;
        }
        if (i === 274) {
          month = 10;
          day = 1;
        }
        if (i === 305) {
          month = 11;
          day = 1;
        }
        if (i === 335) {
          month = 12;
          day = 1;
        }
        //придумать как делать шифр
        dateOfBookingRoom.dateCode = day + "/" +
month + "/2022";
        dateOfBookingRoom.nameOfPerson = "";
        dateOfBookingEmpty.push(dateOfBookingRoom);
        day++;
      }
      setDoc(doc(db, "rooms", numberOfRoom + "_" +
global.owner), {
        balcony: checked,
        costOfRoom: costOfRoom,
        numberOfFloor: numberOfFloor,
        numberOfRooms: numberOfRoom,
        numberOfPersons: numberOfPersons,
        countOfRooms: countOfRooms,
        dateOfBooking: dateOfBookingEmpty,
        owner: global.owner,
      })
        .then(() => {
          // Data saved successfully!
          console.log("data submitted");
          createTwoButtonAlert(
            "Success!",
            "Room successefully adede to the data
base!"
          );
        })
```

```
        .catch((error) => {
          // The write failed...
          console.log(error);
        });
    }
  }

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Adding new
room</Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
room</Text>
        <InputField
          onChangeText={(numberOfRoom) => {
            setNumberOfRoom(numberOfRoom);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
person</Text>
        <InputField
          onChangeText={(numberOfPersons) => {
            setNumberOfPersons(numberOfPersons);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
Floor</Text>
        <InputField
          onChangeText={(numberOfFloor) => {
            setNumberOfFloor(numberOfFloor);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Count of
rooms</Text>
        <InputField
          onChangeText={(countOfRooms) => {
            setCountOfRooms(countOfRooms);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Cost of
room</Text>
        <InputField
          onChangeText={(costOfRoom) => {
            setCostOfRoom(costOfRoom);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.CheckContainer}>
        <Text
style={styles.inputHeader}>Balcony</Text>
        <Checkbox
          status={checked ? "checked" : "unchecked"}
          onPress={() => {
            setChecked(!checked);
          }}
        />
      </View>
      <Button
        onPress={Create}
        style={styles.button}
        title="Add new room"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
```

```
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  CheckContainer: {
    marginTop: 5,
    marginHorizontal: 10,
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
  },
  buttonContainer: {
    borderRadius: 10,
    marginTop: 10,
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
});
```

BookingEditScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useState, useEffect } from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
  Image,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import DateTimePicker from "@react-native-
community/datetimepicker";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();
```

```
export default function BookingEditScreen({
navigation }) {
  const [nameOfPerson, setNameOfPerson] =
useState("");
  const [rooms, setRooms] = useState([]);
  const roomsCollectionsRef = collection(db,
"rooms");

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  useEffect(() => {
    const getRooms = async () => {
      const data = await
getDocs(roomsCollectionsRef);
      setRooms(data.docs.map((doc) => ({
...doc.data() })));
    };
    getRooms();
  }, []);

  const deleteBooking = () => {
    console.log("delete");
    rooms.forEach((element) => {
      if (element.owner == global.owner) {
        let newDateOfBooking = [];
        for (let i = 0; i <
element.dateOfBooking.length; i++) {
          if (element.dateOfBooking[i].nameOfPerson
!= nameOfPerson) {

newDateOfBooking.push(element.dateOfBooking[i]);
          } else {
            let newElem = {
              dateCode:
element.dateOfBooking[i].dateCode,
              nameOfPerson: "",
            };
            newDateOfBooking.push(newElem);
          }
        }
        updateDoc(
          doc(db, "rooms", element.numberOfRooms +
"_" + global.owner),
          {
            dateOfBooking: newDateOfBooking,
          }
        );
        createTwoButtonAlert("Success!", "Booking
successfully deleted!");
      }
    });
  };

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Delete booking
</Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Name of
person</Text>
        <InputField
          onChangeText={(nameOfPerson) => {
            setNameOfPerson(nameOfPerson);
          }}
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <Button
        onPress={deleteBooking}
        style={styles.button}
        title="Delete booking"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
```

```
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  buttonContainer: {
    marginTop: 15,
    borderRadius: 10,
    alignSelf: "center",
    width: "95%",
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  dateButtonContainer: {
    borderRadius: 10,
    alignSelf: "center",
    width: "50%",
    marginBottom: 15,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  imageCalendar: {
    width: 35,
    height: 35,
  },
  dateContainer: {
    marginTop: 15,
    paddingHorizontal: 15,
    borderRadius: 10,
    height: 50,
    backgroundColor: inputColor,
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
  pickedDateContainer: {
    backgroundColor: inputColor,
  },
  pickedDate: {
    fontSize: 18,
    color: buttonColor,
  },
  dateContainerText: {
    width: "70%",
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
});
```

BookingFindScrren.js

```
import { StatusBar } from "expo-status-bar";
import React, { useState, useEffect } from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
  Image,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import DateTimePicker from "@react-native-
community/datetimepicker";
import {
  collection,
```

```
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";
import { elementsThatOverlapOffsets } from "react-
native/Libraries/Lists/VirtualizeUtils";

const auth = Firebase.auth();

export default function BookingFindScreen({
navigation }) {
  //nameOfPerson and replace datapicker
  const [nameOfPerson, setNameOfPerson] =
useState("");
  const [numberOfRoom, setNumberOfRoom] =
useState("");
  const [isPickerShow, setIsPickerShow] =
useState(false);
  const [date, setDate] = useState("");
  const [isPickerShowOut, setIsPickerShowOut] =
useState(false);
  const [dateOut, setDateOut] = useState("");
  const [cost, setCost] = useState("");

  const [rooms, setRooms] = useState([]);
  const roomsCollectionsRef = collection(db,
"rooms");

  useEffect(() => {
    const getRooms = async () => {
      const data = await
getDocs(roomsCollectionsRef);
      setRooms(data.docs.map((doc) => ({
...doc.data() })));
    };
    getRooms();
  }, []);

  const showPicker = () => {
    setIsPickerShow(true);
  };

  const onChange = (event, value) => {
    setDate(value);
    if (Platform.OS === "android") {
      setIsPickerShow(false);
    }
  };

  const showPickerOut = () => {
    setIsPickerShowOut(true);
  };

  const onChangeOut = (event, value) => {
    setDateOut(value);
    if (Platform.OS === "android") {
      setIsPickerShowOut(false);
    }
  };

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  const findBooking = () => {
    let indexIn,
      countDay = 0,
      cost = 0;
    for (let i = 0; i < rooms.length; i++) {}
    rooms.forEach((element) => {
      if (element.owner == global.owner) {
        for (let i = 0; i <
element.dateOfBooking.length; i++) {
          if (element.dateOfBooking[i].nameOfPerson
== nameOfPerson) {
            cost = element.costOfRoom;
            setNumberOfRoom(element.numberOfRooms);

setDate(element.dateOfBooking[i].dateCode);
            indexIn = i;
            break;
          }
        }
        for (let i = indexIn; i <
element.dateOfBooking.length; i++) {
```

```
          if (element.dateOfBooking[i].nameOfPerson
== nameOfPerson) {
            countDay++;

setDateOut(element.dateOfBooking[i].dateCode);
          } else {
            break;
          }
        }
        setCost(countDay * cost);
      }
    });
    if (indexIn == undefined) {
      createTwoButtonAlert("No booking", "No booking
found for this name");
    }
  };
  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Find bookings
</Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Name of
person</Text>
        <InputField
          onChangeText={(nameOfPerson) => {
            setNameOfPerson(nameOfPerson);
          }}
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <View style={styles.dateContainer}>
          <View style={styles.dateContainerText}>
            <TouchableOpacity onPress={showPicker}>
              <Image
                source={require("../assets/calendar-
icon.png")}
                style={styles.imageCalendar}
              ></Image>
            </TouchableOpacity>
            <View style={styles.pickedDateContainer}>
              <Text
style={styles.pickedDate}>{date}</Text>
            </View>

            <Text style={styles.inputHeader}>Date
in:</Text>
          </View>
        </View>
        <View style={styles.inputContainer}>
          <View style={styles.dateContainer}>
            <View style={styles.dateContainerText}>
              <TouchableOpacity
onPress={showPickerOut}>
                <Image
                  source={require("../assets/calendar-
icon.png")}
                  style={styles.imageCalendar}
                ></Image>
              </TouchableOpacity>
              <View style={styles.pickedDateContainer}>
                <Text
style={styles.pickedDate}>{dateOut}</Text>
              </View>
            </View>
            <Text style={styles.inputHeader}>Date
out:</Text>
          </View>
        </View>
        <View style={styles.containerInfo}>
          <Text style={styles.textInfo}> Money: </Text>
          <Text style={styles.textInfo}> {cost}</Text>
        </View>
        <View style={styles.containerInfo}>
          <Text style={styles.textInfo}> Number of
room: </Text>
          <Text style={styles.textInfo}>
{numberOfRoom}</Text>
        </View>

        <Button
          onPress={findBooking}
          style={styles.button}
          title="Search room"
          titleSize={20}
          containerStyle={styles.buttonContainer}
        ></Button>
        {!isPickerShow && <View
style={styles.btnContainer}></View>}
        {isPickerShow && (
          <DateTimePicker
            value={date}
            mode={"date"}
```

```
          is24Hour={true}
          onChange={onChange}
          style={styles.datePicker}
        />
      )}
      {!isPickerShowOut && <View
style={styles.btnContainer}></View>}
      {isPickerShowOut && (
        <DateTimePicker
          value={dateOut}
          mode={"date"}
          is24Hour={true}
          onChange={onChangeOut}
          style={styles.datePicker}
        />
      )}
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  textInfo: {
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  containerInfo: {
    flexDirection: "row",
    justifyContent: "space-between",
    marginTop: 15,
    marginHorizontal: 10,
    borderRadius: 10,
    backgroundColor: inputColor,
    height: 50,
    paddingVertical: 10,
    paddingHorizontal: 15,
  },
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  buttonContainer: {
    marginTop: 15,
    borderRadius: 10,
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  dateButtonContainer: {
    borderRadius: 10,
    marginTop: 5,
    alignSelf: "center",
    width: "50%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  imageCalendar: {
    width: 35,
    height: 35,
  },
  dateContainer: {
    marginTop: 15,
    paddingHorizontal: 15,
    borderRadius: 10,
```

```
    height: 50,
    backgroundColor: inputColor,
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
  pickedDateContainer: {
    backgroundColor: inputColor,
  },
  pickedDate: {
    fontSize: 18,
    color: buttonColor,
  },
  dateContainerText: {
    width: "70%",
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
});
```

BookingScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useState, useEffect }
from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
  Image,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import DateTimePicker from "@react-native-
community/datetimepicker";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function BookingScreen({ navigation })
{
  const [isPickerShow, setIsPickerShow] =
useState(false);
  const [date, setDate] = useState(new
Date(Date.now()));
  const [isPickerShowOut, setIsPickerShowOut] =
useState(false);
  const [dateOut, setDateOut] = useState(new
Date(Date.now()));
  const [nameOfPerson, setNameOfPerson] =
useState("");
  const [numberOfPersons, setNumberOfPersons] =
useState("");

  //rooms
  const [rooms, setRooms] = useState([]);
  const roomsCollectionsRef = collection(db,
"rooms");

  // const createUsers = async () => {};

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  useEffect(() => {
    const getRooms = async () => {
      const data = await
getDocs(roomsCollectionsRef);
      setRooms(data.docs.map((doc) => ({
...doc.data() })));
```

```
    };
    getRooms();
  }, []);

  const showPicker = () => {
    setIsPickerShow(true);
  };

  const onChange = (event, value) => {
    setDate(value);
    if (Platform.OS === "android") {
      setIsPickerShow(false);
    }
  };

  const showPickerOut = () => {
    setIsPickerShowOut(true);
  };

  const onChangeOut = (event, value) => {
    setDateOut(value);
    if (Platform.OS === "android") {
      setIsPickerShowOut(false);
    }
  };

  const validateDate = () => {
    if (date > dateOut) {
      createTwoButtonAlert(
        "Invalid data",
        "Date in cannot be bigger than date out"
      );
      return false;
    } else {
      if (numberOfPersons[0] === "0") {
        createTwoButtonAlert(
          "Invalid data",
          "Value of number of person cannot begining
with zero"
        );
        return false;
      } else {
        if (numberOfPersons > 10) {
          createTwoButtonAlert(
            "Invalid data",
            "Value of number of person shoul be
between 1 and 10"
          );
          return false;
        } else {
          return true;
        }
      }
    }
  };

  const MakeBooking = () => {
    if (validateDate()) {
      let booking = {
        nameOfPerson: nameOfPerson,
        dateIN:
          date.getDate() +
          "/" +
          (date.getMonth() + 1) +
          "/" +
          date.getFullYear(),
        dateOUT:
          dateOut.getDate() +
          "/" +
          (dateOut.getMonth() + 1) +
          "/" +
          dateOut.getFullYear(),
      };
      let searchRoom = false;
      rooms.forEach((element) => {
        if (!searchRoom) {
          if (
            element.numberOfPersons ===
numberOfPersons &&
            element.owner == global.owner
          ) {
            let bookingDateIN = false,
              bookingDateOUT = false;
            let indexIN = 0,
              indexOUT = 1;
            for (let i = 0; i <
element.dateOfBooking.length; i++) {
              if (element.dateOfBooking[i].dateCode
== booking.dateIN) {
                if
(element.dateOfBooking[i].nameOfPerson == "") {
                  bookingDateIN = true;
                  indexIN = i;
                }
              }
            }
            if (bookingDateIN) {
              for (let i = 0; i <
element.dateOfBooking.length; i++) {
```

```
                if (element.dateOfBooking[i].dateCode
== booking.dateOUT) {
                  if
(element.dateOfBooking[i].nameOfPerson == "") {
                    bookingDateOUT = true;
                    indexOUT = i;
                  }
                }
              }
            }
            if (bookingDateOUT) {
              for (let i = indexIN; i < indexOUT;
i++) {
                element.dateOfBooking[i].nameOfPerson
= booking.nameOfPerson;
                searchRoom = true;
              }
            }
            updateDoc(
              doc(db, "rooms", element.numberOfRooms
+ "_" + global.owner),
              {
                dateOfBooking: element.dateOfBooking,
              }
            );
            createTwoButtonAlert(
              "Success!",
              "Booking successfully added to the
room!"
            );
          }
        }
      });
      if (!searchRoom) {
        createTwoButtonAlert(
          "Couldn't find a room",
          "There are no available rooms for the
selected dates. Please try other days."
        );
      }
    }
  };

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Booking </Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Name of
person</Text>
        <InputField
          onChangeText={(nameOfPerson) => {
            setNameOfPerson(nameOfPerson);
          }}
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Count of
person</Text>
        <InputField
          onChangeText={(numberOfPerson) => {
            setNumberOfPersons(numberOfPerson);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <View style={styles.dateContainer}>
          <View style={styles.dateContainerText}>
            <TouchableOpacity onPress={showPicker}>
              <Image
                source={require("../assets/calendar-
icon.png")}
                style={styles.imageCalendar}
              ></Image>
            </TouchableOpacity>
            <View style={styles.pickedDateContainer}>
              <Text style={styles.pickedDate}>
                {date.getDate() +
                  "/" +
                  (date.getMonth() + 1) +
                  "/" +
                  date.getFullYear()}
              </Text>
            </View>
          </View>

          <Text style={styles.inputHeader}>Date
in:</Text>
        </View>
```

```
      </View>
      <View style={styles.inputContainer}>
        <View style={styles.dateContainer}>
          <View style={styles.dateContainerText}>
            <TouchableOpacity
onPress={showPickerOut}>
              <Image
                source={require("../assets/calendar-
icon.png")}
                style={styles.imageCalendar}
              ></Image>
            </TouchableOpacity>
            <View style={styles.pickedDateContainer}>
              <Text style={styles.pickedDate}>
                {dateOut.getDate() +
                  "/" +
                  (dateOut.getMonth() + 1) +
                  "/" +
                  dateOut.getFullYear()}
              </Text>
            </View>
          </View>
          <Text style={styles.inputHeader}>Date
out:</Text>
        </View>
      </View>
      <Button
        onPress={MakeBooking}
        style={styles.button}
        title="Book room"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      {!isPickerShow && <View
style={styles.btnContainer}></View>}
      {isPickerShow && (
        <DateTimePicker
          value={date}
          mode={"date"}
          is24Hour={true}
          onChange={onChange}
          style={styles.datePicker}
        />
      )}
      {!isPickerShowOut && <View
style={styles.btnContainer}></View>}
      {isPickerShowOut && (
        <DateTimePicker
          value={dateOut}
          mode={"date"}
          is24Hour={true}
          onChange={onChangeOut}
          style={styles.datePicker}
        />
      )}
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  buttonContainer: {
    marginTop: 15,
    borderRadius: 10,
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
    backgroundColor: buttonColor,
```

```
    color: buttonTextColor,
  },
  dateButtonContainer: {
    borderRadius: 10,
    marginTop: 5,
    alignSelf: "center",
    width: "50%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  imageCalendar: {
    width: 35,
    height: 35,
  },
  dateContainer: {
    marginTop: 15,
    paddingHorizontal: 15,
    borderRadius: 10,
    height: 50,
    backgroundColor: inputColor,
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
  pickedDateContainer: {
    backgroundColor: inputColor,
  },
  pickedDate: {
    fontSize: 18,
    color: buttonColor,
  },
  dateContainerText: {
    width: "70%",
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
});
```

CommonReportScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useEffect, useState }
from "react";
import {
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function CommoReportScreen({
navigation }) {
  const [income, setIncome] = useState("");
  const [rooms, setRooms] = useState([]);
  const roomsCollectionsRef = collection(db,
"rooms");

  useEffect(() => {
    const getRooms = async () => {
      const data = await
getDocs(roomsCollectionsRef);
      setRooms(data.docs.map((doc) => ({
...doc.data() })));
    };
    getRooms();
  }, []);

  const calculate = () => {
    let income = 0,
      countDays,
```

```
      incomePerRoom,
      cost;
    rooms.forEach((element) => {
      if (element.owner == global.owner) {
        countDays = 0;
        incomePerRoom = 0;
        cost = element.costOfRoom;
        for (let i = 0; i <
element.dateOfBooking.length; i++) {
          if (element.dateOfBooking[i].nameOfPerson
!= "") {
            countDays++;
          }
        }
        incomePerRoom = cost * countDays;
        income = income + incomePerRoom;
      }
    });
    setIncome(income);
  };

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Common
Report</Text>
      </View>
      <View style={styles.containerInfo}>
        <Text style={styles.textInfo}> Income:
</Text>
        <Text style={styles.textInfo}>
{income}</Text>
      </View>
      <Button
        onPress={calculate}
        style={styles.button}
        title="Calculate income"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  textInfo: {
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  containerInfo: {
    flexDirection: "row",
    justifyContent: "space-between",
    marginTop: 15,
    marginHorizontal: 10,
    borderRadius: 10,
    backgroundColor: inputColor,
    height: 50,
    paddingVertical: 10,
    paddingHorizontal: 15,
  },
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  buttonContainer: {
    marginTop: 15,
    borderRadius: 10,
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
```

```
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  dateButtonContainer: {
    borderRadius: 10,
    marginTop: 5,
    alignSelf: "center",
    width: "50%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  imageCalendar: {
    width: 35,
    height: 35,
  },
  dateContainer: {
    marginTop: 15,
    paddingHorizontal: 15,
    borderRadius: 10,
    height: 50,
    backgroundColor: inputColor,
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
  pickedDateContainer: {
    backgroundColor: inputColor,
  },
  pickedDate: {
    fontSize: 18,
    color: buttonColor,
  },
  dateContainerText: {
    width: "70%",
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
});
```

EditScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useState } from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import { Checkbox } from "react-native-paper";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function EditScreen({ navigation }) {
  //balckony
  const [checked, setChecked] = useState(false);
  const [costOfRoom, setCostOfRoom] = useState("");
  const [numberOfFloor, setNumberOfFloor] =
useState("");
  const [numberOfPersons, setNumberOfPersons] =
useState("");
  const [numberOfRoom, setNumberOfRoom] =
useState("");
  const [countOfRooms, setCountOfRooms] =
useState("");

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
```

```
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  const correctData = () => {
    if (
      costOfRoom === "" ||
      countOfRooms === "" ||
      numberOfFloor === "" ||
      numberOfPersons === "" ||
      numberOfRoom === ""
    ) {
      createTwoButtonAlert("Invalid data", "Fields
cannot be empty");
      return false;
    }
    if (costOfRoom[0] === "0") {
      createTwoButtonAlert(
        "Invalid data",
        "Value of cost of room cannot begining with
zero"
      );
      return false;
    } else {
      if (numberOfFloor[0] === "0") {
        createTwoButtonAlert(
          "Invalid data",
          "Value of number of floor cannot begining
with zero"
        );
        return false;
      } else {
        if (numberOfPersons[0] === "0") {
          createTwoButtonAlert(
            "Invalid data",
            "Value of number of person cannot
begining with zero"
          );
          return false;
        } else {
          if (numberOfRoom[0] === "0") {
            createTwoButtonAlert(
              "Invalid data",
              "Value of number of room cannot
begining with zero"
            );
            return false;
          } else {
            if (countOfRooms[0] === "0") {
              createTwoButtonAlert(
                "Invalid data",
                "Value of count of rooms cannot
begining with zero"
              );
              return false;
            } else {
              return true;
            }
          }
        }
      }
    }
  };

  const validateData = () => {
    if (countOfRooms > 5) {
      createTwoButtonAlert(
        "Invalid data",
        "Value of count of rooms shoul be between 1
and 5"
      );
      return false;
    } else {
      if (numberOfPersons > 20) {
        createTwoButtonAlert(
          "Invalid data",
          "Value of number of persons shoul be
between 1 and 20"
        );
        return false;
      } else {
        if (numberOfPersons > 10) {
          createTwoButtonAlert(
            "Invalid data",
            "Value of number of person shoul be
between 1 and 10"
          );
          return false;
        } else {
          return true;
        }
      }
    }
  };

  function Update() {
    if (correctData() && validateData()) {
```

```
      updateDoc(doc(db, "rooms", numberOfRoom + "_" +
global.owner), {
        balcony: checked,
        costOfRoom: costOfRoom,
        numberOfFloor: numberOfFloor,
        numberOfRooms: numberOfRoom,
        numberOfPersons: numberOfPersons,
        countOfRooms: countOfRooms,
      })
        .then(() => {
          // Data saved successfully!
          console.log("data submitted");
          console.log(typeof numberOfRoom);
        })
        .catch((error) => {
          // The write failed...
          console.log(error);
        });
    }
  }
  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Edit menu</Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
room</Text>
        <InputField
          onChangeText={(numberOfRoom) => {
            setNumberOfRoom(numberOfRoom);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
person</Text>
        <InputField
          onChangeText={(numberOfPersons) => {
            setNumberOfPersons(numberOfPersons);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
Floor</Text>
        <InputField
          onChangeText={(numberOfFloor) => {
            setNumberOfFloor(numberOfFloor);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Count of
rooms</Text>
        <InputField
          onChangeText={(countOfRooms) => {
            setCountOfRooms(countOfRooms);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Cost of
room</Text>
        <InputField
          onChangeText={(costOfRoom) => {
            setCostOfRoom(costOfRoom);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.CheckContainer}>
        <Text
style={styles.inputHeader}>Balcony</Text>
        <Checkbox
          status={checked ? "checked" : "unchecked"}
```

```
          onPress={() => {
            setChecked(!checked);
          }}
        />
      </View>
      <Button
        onPress={Update}
        style={styles.button}
        title="Edit room"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  CheckContainer: {
    marginTop: 5,
    marginHorizontal: 10,
    flexDirection: "row",
    justifyContent: "space-between",
    alignItems: "center",
  },
  buttonContainer: {
    borderRadius: 10,
    marginTop: 10,
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
});
```

HomeScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useState } from "react";
import {
  FlatList,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton } from "../components";
import Firebase from "../config/firebase";
import { AuthenticatedUserContext } from
"../navigation/AuthenticatedUserProvider";

const auth = Firebase.auth();

export default function HomeScreen({ navigation }) {
  const [openEdit, setOpenEdit] = useState(false);
  const toggleEdit = () => setOpenEdit(!openEdit);
  const [open, setOpen] = useState(false);
```

```
  const toggle = () => setOpen(!open);
  const [openBooking, setOpenBooking] =
useState(false);
  const toggleBooking = () =>
setOpenBooking(!openBooking);
  const { user } =
useContext(AuthenticatedUserContext);
  const handleSignOut = async () => {
    try {
      await auth.signOut();
    } catch (error) {
      console.log(error);
    }
  };

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Main menu</Text>
        <IconButton
          name="logout"
          size={24}
          color={buttonColor}
          onPress={handleSignOut}
        />
      </View>
      <View style={styles.containerDropDownEdit}>
        <TouchableOpacity
          style={styles.headerContainerDropDownEdit}
          onPress={() => toggle(!open)}
        >
          <Text
style={styles.optionHeaderDropDownEdit}> Edit data
base</Text>
        </TouchableOpacity>
        {open && (
          <FlatList
            data={[
              { key: "Edit existing room" },
              { key: "Remove existing room" },
              { key: "Add new room" },
            ]}
            renderItem={({ item }) => (
              <TouchableOpacity

style={styles.itemContainerDropDownEdit}
                onPress={() => {
                  if (item.key == "Edit existing
room") {
                    navigation.navigate("Edit");
                  }
                  if (item.key == "Remove existing
room") {
                    navigation.navigate("Remove");
                  }
                  if (item.key == "Add new room") {
                    navigation.navigate("Add");
                  }
                }}
              >
                <Text
style={styles.itemDropDownEdit}>{item.key}</Text>
              </TouchableOpacity>
            )}
          />
        )}
      </View>
      <View style={styles.containerDropDownEdit}>
        <TouchableOpacity
          style={styles.headerContainerDropDownEdit}
          onPress={() => toggleBooking(!openBooking)}
        >
          <Text
style={styles.optionHeaderDropDownEdit}>
Booking</Text>
        </TouchableOpacity>
        {openBooking && (
          <FlatList
            data={[
              { key: "Make new booking" },
              { key: "Edit existing booking" },
              { key: "Find booking" },
            ]}
            renderItem={({ item }) => (
              <TouchableOpacity

style={styles.itemContainerDropDownEdit}
                onPress={() => {
                  if (item.key == "Make new booking")
{

navigation.navigate("BookingNew");
                  }
                  if (item.key == "Edit existing
booking") {

navigation.navigate("BookingEdit");
                  }
                  if (item.key == "Find booking") {
```

```
navigation.navigate("BookingFind");
                  }
                }}
              >
                <Text
style={styles.itemDropDownEdit}>{item.key}</Text>
              </TouchableOpacity>
            )}
          />
        )}
      </View>
      <View style={styles.containerDropDownEdit}>
        <TouchableOpacity
          style={styles.headerContainerDropDownEdit}
          onPress={() => toggleEdit(!openEdit)}
        >
          <Text
style={styles.optionHeaderDropDownEdit}> Create
reports</Text>
        </TouchableOpacity>
        {openEdit && (
          <FlatList
            data={[
              { key: "All earnings report" },
              { key: "Income report per room" },
              { key: "Income period time" },
            ]}
            renderItem={({ item }) => (
              <TouchableOpacity

style={styles.itemContainerDropDownEdit}
                onPress={() => {
                  if (item.key == "All earnings
report") {

navigation.navigate("CommonReport");
                  }
                  if (item.key == "Income report per
room") {

navigation.navigate("IncomeRoom");
                  }
                  if (item.key == "Income period
time") {

navigation.navigate("IncomeTime");
                  }
                }}
              >
                <Text
style={styles.itemDropDownEdit}>{item.key}</Text>
              </TouchableOpacity>
            )}
          />
        )}
      </View>

      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: "column",
    justifyContent: "flex-start",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 28,
    fontWeight: "700",
    color: buttonColor,
  },
  listItem: {
    padding: 15,
    marginTop: 20,
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  optionHeader: {
    fontSize: 24,
    color: buttonColor,
    fontWeight: "600",
  },
  optionDescription: {
    marginTop: 5,
    paddingHorizontal: 5,
    fontSize: 18,
    color: buttonColor,
    textAlign: "justify",
```

```
  },
  containerDropDownEdit: {
    marginTop: 20,
    backgroundColor: pageBackground,
  },
  headerContainerDropDownEdit: {
    borderRadius: 10,
    padding: 10,
    paddingHorizontal: 15,
    width: "100%",
    backgroundColor: inputColor,
  },
  optionHeaderDropDownEdit: {
    height: 30,
    fontSize: 24,
    color: buttonColor,

    fontWeight: "600",
  },
  itemContainerDropDownEdit: {
    alignSelf: "flex-end",
    width: "92%",
    padding: 10,
    paddingLeft: 15,
    backgroundColor: inputColor,
    borderRadius: 10,
    marginTop: 15,
  },
  itemDropDownEdit: {
    fontSize: 18,
    color: buttonColor,
  },
});
```

IncomeRoomScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useEffect, useState }
from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function IncomeRoomScreen({ navigation
}) {
  const [numberOfRoom, setNuberOfRoom] =
useState("");
  const [income, setIncome] = useState("");
  const [rooms, setRooms] = useState([]);
  const roomsCollectionsRef = collection(db,
"rooms");

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  useEffect(() => {
    const getRooms = async () => {
      const data = await
getDocs(roomsCollectionsRef);
      setRooms(data.docs.map((doc) => ({
...doc.data() })));
    };
    getRooms();
  }, []);
```

```
  const correctData = () => {
    if (numberOfRoom === "") {
      createTwoButtonAlert("Invalid data", "Fields
cannot be empty");
      return false;
    }
    if (numberOfRoom[0] === "0") {
      createTwoButtonAlert(
        "Invalid data",
        "Value of number of room cannot begining with
zero"
      );
      return false;
    }
  };

  const calculate = () => {
    if (correctData()) {
      let income = 0,
        countDays = 0,
        cost = 0;
      rooms.forEach((element) => {
        if (element.owner == global.owner) {
          if (element.numberOfRooms == numberOfRoom)
{
            cost = element.costOfRoom;
            for (let i = 0; i <
element.dateOfBooking.length; i++) {
              if
(element.dateOfBooking[i].nameOfPerson != "") {
                countDays++;
              }
            }
          }
        }
      });
      income = countDays * cost;
      setIncome(income);
    }
  };

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Incom Room</Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
room</Text>
        <InputField
          keyboardType="numeric"
          onChangeText={(numberOfRoom) => {
            setNuberOfRoom(numberOfRoom);
          }}
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <View style={styles.containerInfo}>
        <Text style={styles.textInfo}> Income:
</Text>
        <Text style={styles.textInfo}>
{income}</Text>
      </View>
      <Button
        onPress={calculate}
        style={styles.button}
        title="Calculate income"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      <StatusBar style="dark-content" />
    </View>
  );
}
const styles = StyleSheet.create({
  textInfo: {
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  containerInfo: {
    flexDirection: "row",
    justifyContent: "space-between",
    marginTop: 15,
    marginHorizontal: 10,
    borderRadius: 10,
    backgroundColor: inputColor,
    height: 50,
    paddingVertical: 10,
    paddingHorizontal: 15,
  },
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
```

```
  paddingHorizontal: 12,
},
containerMenuTitle: {
  alignSelf: "center",
  width: "97%",
  flexDirection: "row",
  justifyContent: "space-between",
},
title: {
  fontSize: 24,
  fontWeight: "700",
  color: buttonColor,
},
inputHeader: {
  marginVertical: 7,
  marginHorizontal: 5,
  fontSize: 20,
  color: buttonColor,
  fontWeight: "600",
},
inputContainer: {
  marginHorizontal: 10,
},
input: {
  backgroundColor: inputColor,
  borderRadius: 10,
},
buttonContainer: {
  marginTop: 15,
  borderRadius: 10,
  alignSelf: "center",
  width: "95%",
  marginBottom: 24,
  backgroundColor: buttonColor,
  color: buttonTextColor,
},
dateButtonContainer: {
  borderRadius: 10,
  marginTop: 5,
  alignSelf: "center",
  width: "50%",
  marginBottom: 24,
  backgroundColor: buttonColor,
  color: buttonTextColor,
},
button: {
  backgroundColor: buttonColor,
  color: buttonTextColor,
},
imageCalendar: {
  width: 35,
  height: 35,
},
dateContainer: {
  marginTop: 15,
  paddingHorizontal: 15,
  borderRadius: 10,
  height: 50,
  backgroundColor: inputColor,
  flexDirection: "row-reverse",
  justifyContent: "space-between",
  alignItems: "center",
},
pickedDateContainer: {
  backgroundColor: inputColor,
},
pickedDate: {
  fontSize: 18,
  color: buttonColor,
},
dateContainerText: {
  width: "70%",
  flexDirection: "row-reverse",
  justifyContent: "space-between",
  alignItems: "center",
},
});
```

IncomeTimeScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useContext, useState, useEffect }
from "react";
import {
  Alert,
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
  Image,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
```

```
import DateTimePicker from "@react-native-
community/datetimepicker";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function IncomeTimeScreen({ navigation
}) {
  const [isPickerShow, setIsPickerShow] =
useState(false);
  const [date, setDate] = useState(new
Date(Date.now()));
  const [isPickerShowOut, setIsPickerShowOut] =
useState(false);
  const [dateOut, setDateOut] = useState(new
Date(Date.now()));
  const [income, setIncome] = useState("");
  //rooms
  const [rooms, setRooms] = useState([]);
  const roomsCollectionsRef = collection(db,
"rooms");

  // const createUsers = async () => {};

  const createTwoButtonAlert = (title, message) =>
    Alert.alert(title, message, [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel",
      },
      { text: "OK", onPress: () => console.log("OK
Pressed") },
    ]);

  useEffect(() => {
    const getRooms = async () => {
      const data = await
getDocs(roomsCollectionsRef);
      setRooms(data.docs.map((doc) => ({
...doc.data() })));
    };
    getRooms();
  }, []);

  const showPicker = () => {
    setIsPickerShow(true);
  };

  const onChange = (event, value) => {
    setDate(value);
    if (Platform.OS === "android") {
      setIsPickerShow(false);
    }
  };

  const showPickerOut = () => {
    setIsPickerShowOut(true);
  };

  const onChangeOut = (event, value) => {
    setDateOut(value);
    if (Platform.OS === "android") {
      setIsPickerShowOut(false);
    }
  };

  const validateDate = () => {
    if (date > dateOut) {
      //1
      createTwoButtonAlert(
        "Invalid data",
        "Date in cannot be bigger than date out"
      );
      return false; //2
    } else {
      return true; //3
    }
  };

  const Calculate = () => {
    if (validateDate()) {
      let income = 0, //4
        allIncome = 0,
        countDays = 0,
        cost = 0;
      let indexIn = 0, //5
```

```
        indexOut = 0;
      let dateIn =
        date.getDate() + "/" + (date.getMonth() + 1)
+ "/" + date.getFullYear(); //6
      let dateOUT =
        dateOut.getDate() +
        "/" +
        (dateOut.getMonth() + 1) +
        "/" +
        dateOut.getFullYear(); //7
      for (let i = 0; i <
rooms[0].dateOfBooking.length; i++) {
        //8
        if (rooms[0].dateOfBooking[i].dateCode ==
dateIn) {
          //9
          indexIn = i; //10
          for (let j = indexIn; j <
rooms[0].dateOfBooking.length; j++) {
            //11
            if (rooms[0].dateOfBooking[j].dateCode ==
dateOUT) {
              //12
              indexOut = j; //13
            }
          }
        }
      }
      rooms.forEach((element) => {
        //14
        if (element.owner == global.owner) {
          //15
          cost = element.costOfRoom; //16
          for (let k = indexIn; k < indexOut; k++) {
            //17
            if (element.dateOfBooking[k].nameOfPerson
!== "") {
              //18
              countDays++; //19
            }
          }
          income = countDays * cost; //20
        }
        allIncome = allIncome + income; //21
        countDays = 0; //22
        income = 0; //23
      });
      setIncome(allIncome); //24
    }
  };

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Income period
time </Text>
      </View>

      <View style={styles.containerInfo}>
        <Text style={styles.textInfo}> Income:
</Text>
        <Text style={styles.textInfo}>
{income}</Text>
      </View>
      <View style={styles.inputContainer}>
        <View style={styles.dateContainer}>
          <View style={styles.dateContainerText}>
            <TouchableOpacity onPress={showPicker}>
              <Image
                source={require("../assets/calendar-
icon.png")}
                style={styles.imageCalendar}
              ></Image>
            </TouchableOpacity>
            <View style={styles.pickedDateContainer}>
              <Text style={styles.pickedDate}>
                {date.getDate() +
                  "/" +
                  (date.getMonth() + 1) +
                  "/" +
                  date.getFullYear()}
              </Text>
            </View>
          </View>

          <Text style={styles.inputHeader}>Date
in:</Text>
        </View>
      </View>
      <View style={styles.inputContainer}>
        <View style={styles.dateContainer}>
          <View style={styles.dateContainerText}>
            <TouchableOpacity
onPress={showPickerOut}>
              <Image
                source={require("../assets/calendar-
icon.png")}
                style={styles.imageCalendar}
              ></Image>
            </TouchableOpacity>
```

```
            <View style={styles.pickedDateContainer}>
              <Text style={styles.pickedDate}>
                {dateOut.getDate() +
                  "/" +
                  (dateOut.getMonth() + 1) +
                  "/" +
                  dateOut.getFullYear()}
              </Text>
            </View>
          </View>
          <Text style={styles.inputHeader}>Date
out:</Text>
        </View>
      </View>
      <Button
        onPress={Calculate}
        style={styles.button}
        title="Calculate"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      {!isPickerShow && <View
style={styles.btnContainer}></View>}
      {isPickerShow && (
        <DateTimePicker
          value={date}
          mode={"date"}
          is24Hour={true}
          onChange={onChange}
          style={styles.datePicker}
        />
      )}
      {!isPickerShowOut && <View
style={styles.btnContainer}></View>}
      {isPickerShowOut && (
        <DateTimePicker
          value={dateOut}
          mode={"date"}
          is24Hour={true}
          onChange={onChangeOut}
          style={styles.datePicker}
        />
      )}
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  textInfo: {
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  containerInfo: {
    flexDirection: "row",
    justifyContent: "space-between",
    marginTop: 15,
    marginHorizontal: 10,
    borderRadius: 10,
    backgroundColor: inputColor,
    height: 50,
    paddingVertical: 10,
    paddingHorizontal: 15,
  },
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    marginHorizontal: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  buttonContainer: {
    marginTop: 15,
    borderRadius: 10,
```

```
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  dateButtonContainer: {
    borderRadius: 10,
    marginTop: 5,
    alignSelf: "center",
    width: "50%",
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  imageCalendar: {
    width: 35,
    height: 35,
  },
  dateContainer: {
    marginTop: 15,
    paddingHorizontal: 15,
    borderRadius: 10,
    height: 50,
    backgroundColor: inputColor,
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
  pickedDateContainer: {
    backgroundColor: inputColor,
  },
  pickedDate: {
    fontSize: 18,
    color: buttonColor,
  },
  dateContainerText: {
    width: "70%",
    flexDirection: "row-reverse",
    justifyContent: "space-between",
    alignItems: "center",
  },
});
```

LoginScreen.js

```
import { StatusBar } from "expo-status-bar";
import React from "react";
import { useState } from "react";
import { StyleSheet, Text, View, Button as RNButton }
from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, InputField, ErrorMessage } from
"../components";
import Firebase from "../config/firebase";

const auth = Firebase.auth();

const LoginScreen = ({ navigation }) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [passwordVisibility, setPasswordVisibility] =
useState(true);
  const [rightIcon, setRightIcon] = useState("eye");
  const [loginError, setLoginError] = useState("");
  global.owner = email;
  const handlePasswordVisibility = () => {
    if (rightIcon === "eye") {
      setRightIcon("eye-off");
      setPasswordVisibility(!passwordVisibility);
    } else if (rightIcon === "eye-off") {
      setRightIcon("eye");
      setPasswordVisibility(!passwordVisibility);
    }
  };

  const onLogin = async () => {
    try {
      if (email !== "" && password !== "") {
        await auth.signInWithEmailAndPassword(email,
password);
      }
    } catch (error) {
      setLoginError(error.message);
    }
  };

  return (
    <View style={styles.container}>
      <StatusBar style="dark-content" />
      <Text style={styles.title}>Login</Text>
      <InputField
```

```
        inputStyle={{
          fontSize: 14,
        }}
        containerStyle={styles.input}
        leftIcon="email"
        placeholder="Enter email"
        autoCapitalize="none"
        keyboardType="email-address"
        textContentType="emailAddress"
        autoFocus={true}
        value={email}
        onChangeText={(text) => {
          setEmail(text);
        }}
      />
      <InputField
        inputStyle={{
          fontSize: 14,
        }}
        containerStyle={styles.input}
        leftIcon="lock"
        placeholder="Enter password"
        autoCapitalize="none"
        autoCorrect={false}
        secureTextEntry={passwordVisibility}
        textContentType="password"
        rightIcon={rightIcon}
        value={password}
        onChangeText={(text) => setPassword(text)}

handlePasswordVisibility={handlePasswordVisibility}
      />
      {loginError ? <ErrorMessage error={loginError}
visible={true} /> : null}
      <Button
        onPress={onLogin}
        style={styles.button}
        title="Login"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      />
      <RNButton
        onPress={() => navigation.navigate("Signup")}
        title="Go to Signup"
        color={buttonColor}
      />
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    justifyContent: "center",
    alignContent: "center",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    alignSelf: "center",
    paddingBottom: 24,
    color: buttonColor,
  },
  buttonContainer: {
    marginBottom: 24,
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
  input: {
    backgroundColor: inputColor,
    marginBottom: 20,
  },
});

export default LoginScreen;
```

RemoveScreen.js

```
import { StatusBar } from "expo-status-bar";
import React, { useState } from "react";
import {
  TouchableOpacity,
  StyleSheet,
  Text,
  View,
  Button as RNButton,
} from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
```

```
import { Button, IconButton, InputField } from
"../components";
import Firebase from "../config/firebase";
import {
  collection,
  doc,
  setDoc,
  addDoc,
  updateDoc,
  deleteDoc,
  getDoc,
  getDocs,
  where,
  query,
} from "firebase/firestore";
import { db } from "../config/firestore";

const auth = Firebase.auth();

export default function RemoveScreen({ navigation })
{
  const [numberOfRoom, setNumberOfRoom] =
useState("");

  function Delete() {
    deleteDoc(doc(db, "rooms", numberOfRoom));
  }

  return (
    <View style={styles.container}>
      <View style={styles.containerMenuTitle}>
        <Text style={styles.title}> Remove
menu</Text>
      </View>
      <View style={styles.inputContainer}>
        <Text style={styles.inputHeader}>Number of
room</Text>
        <InputField
          onChangeText={(numberOfRoom) => {
            setNumberOfRoom(numberOfRoom);
          }}
          keyboardType="numeric"
          nputStyle={{
            fontSize: 12,
          }}
          containerStyle={styles.input}
        ></InputField>
      </View>
      <Button
        onPress={Delete}
        style={styles.button}
        title="Remove room"
        titleSize={20}
        containerStyle={styles.buttonContainer}
      ></Button>
      <StatusBar style="dark-content" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  containerMenuTitle: {
    alignSelf: "center",
    width: "97%",
    flexDirection: "row",
    justifyContent: "space-between",
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    color: buttonColor,
  },
  inputHeader: {
    marginVertical: 7,
    marginHorizontal: 5,
    fontSize: 20,
    color: buttonColor,
    fontWeight: "600",
  },
  inputContainer: {
    margin: 10,
  },
  input: {
    backgroundColor: inputColor,
    borderRadius: 10,
  },
  buttonContainer: {
    borderRadius: 10,
    marginTop: 5,
    alignSelf: "center",
    width: "95%",
    marginBottom: 24,
    backgroundColor: buttonColor,
```

```
    color: buttonTextColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
});
```

SignupScreen.js

```
import { StatusBar } from "expo-status-bar";
import React from "react";
import { useState } from "react";
import { StyleSheet, Text, View, Button as RNButton }
from "react-native";
import {
  pageBackground,
  buttonColor,
  buttonTextColor,
  inputColor,
} from "../colors";
import { Button, InputField, ErrorMessage } from
"../components";
import Firebase from "../config/firebase";

const auth = Firebase.auth();

export default function SignupScreen({ navigation })
{
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [passwordVisibility, setPasswordVisibility] =
useState(true);
  const [rightIcon, setRightIcon] = useState("eye");
  const [signupError, setSignupError] = useState("");

  const handlePasswordVisibility = () => {
    if (rightIcon === "eye") {
      setRightIcon("eye-off");
      setPasswordVisibility(!passwordVisibility);
    } else if (rightIcon === "eye-off") {
      setRightIcon("eye");
      setPasswordVisibility(!passwordVisibility);
    }
  };

  const onHandleSignup = async () => {
    try {
      if (email !== "" && password !== "") {
        await
auth.createUserWithEmailAndPassword(email, password);
      }
    } catch (error) {
      setSignupError(error.message);
    }
  };

  return (
    <View style={styles.container}>
      <StatusBar style="dark-content" />
      <Text style={styles.title}>Create new
account</Text>
      <InputField
        inputStyle={{
          fontSize: 14,
        }}
        containerStyle={{
          backgroundColor: "#fff",
          marginBottom: 20,
        }}
        leftIcon="email"
        placeholder="Enter email"
        autoCapitalize="none"
        keyboardType="email-address"
        textContentType="emailAddress"
        autoFocus={true}
        value={email}
        onChangeText={(text) => setEmail(text)}
      />
      <InputField
        inputStyle={{
          fontSize: 14,
        }}
        containerStyle={{
          backgroundColor: "#fff",
          marginBottom: 20,
        }}
        leftIcon="lock"
        placeholder="Enter password"
        autoCapitalize="none"
        autoCorrect={false}
        secureTextEntry={passwordVisibility}
        textContentType="password"
        rightIcon={rightIcon}
        value={password}
        onChangeText={(text) => setPassword(text)}

handlePasswordVisibility={handlePasswordVisibility}
      />
      {signupError ? <ErrorMessage
error={signupError} visible={true} /> : null}
```

```
      <Button
        onPress={onHandleSignup}
        containerStyle={styles.buttonContainer}
        title="Signup"
        tileColor="#fff"
        titleSize={20}
        style={styles.button}
      />
      <RNButton
        onPress={() => navigation.navigate("Login")}
        title="Go to Login"
        color={buttonColor}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    display: "flex",
    justifyContent: "center",
    alignContent: "center",
    backgroundColor: pageBackground,
    paddingTop: 50,
    paddingHorizontal: 12,
  },
  title: {
    fontSize: 24,
    fontWeight: "700",
    alignSelf: "center",
    paddingBottom: 24,
    color: buttonColor,
  },
  buttonContainer: {
    marginBottom: 24,
    backgroundColor: buttonColor,
  },
  button: {
    backgroundColor: buttonColor,
    color: buttonTextColor,
  },
});
```

AuthenticatedUserProvider.js

```
import React, { useState, createContext } from
"react";

export const AuthenticatedUserContext =
createContext({});

export const AuthenticatedUserProvider = ({ children
}) => {
  const [user, setUser] = useState(null);

  return (
    <AuthenticatedUserContext.Provider value={{ user,
setUser }}>
      {children}
    </AuthenticatedUserContext.Provider>
  );
};
```

AuthStack.js

```
import React from "react";
import { createStackNavigator } from "@react-
navigation/stack";

import LoginScreen from "../screens/LoginScreen";
import SignupScreen from "../screens/SignupScreen";

const Stack = createStackNavigator();

export default function AuthStack() {
  return (
    <Stack.Navigator headerMode="none">
      <Stack.Screen name="Login"
component={LoginScreen} />
      <Stack.Screen name="Signup"
component={SignupScreen} />
    </Stack.Navigator>
  );
}
```

HomeStack.js

```
import React from "react";
import { createStackNavigator } from "@react-
navigation/stack";

import HomeScreen from "../screens/HomeScreen";
import EditScreen from "../screens/EditScreen";
import RemoveScreen from "../screens/RemoveScreen";
import AddScreen from "../screens/AddScreen";
import CommonReportScreen from
"../screens/CommonReportScreen";
import IncomeRoomScreen from
"../screens/IncomeRoomScreen";
import IncomeTimeScreen from
"../screens/IncomeTimeScreen";
import BookingScreen from "../screens/BookingScreen";
```

```
import BookingEditScreen from
"../screens/BookingEditScreen";
import BookingFindScreen from
"../screens/BookingFindScreen";

const Stack = createStackNavigator();

export default function HomeStack() {
  return (
    <Stack.Navigator headerMode="none">
      <Stack.Screen name="Home"
component={HomeScreen} />
      <Stack.Screen name="Edit"
component={EditScreen} />
      <Stack.Screen name="Remove"
component={RemoveScreen} />
      <Stack.Screen name="Add" component={AddScreen}
/>
      <Stack.Screen name="CommonReport"
component={CommonReportScreen} />
      <Stack.Screen name="IncomeRoom"
component={IncomeRoomScreen} />
      <Stack.Screen name="IncomeTime"
component={IncomeTimeScreen} />
      <Stack.Screen name="BookingNew"
component={BookingScreen} />
      <Stack.Screen name="BookingEdit"
component={BookingEditScreen} />
      <Stack.Screen name="BookingFind"
component={BookingFindScreen} />
    </Stack.Navigator>
  );
}
```

RootNavigator.js

```
import React, { useContext, useEffect, useState }
from "react";
import { NavigationContainer } from "@react-
navigation/native";
import { View, ActivityIndicator } from "react-
native";

import Firebase from "../config/firebase";
import { AuthenticatedUserContext } from
"./AuthenticatedUserProvider";
import AuthStack from "./AuthStack";
import HomeStack from "./HomeStack";

const auth = Firebase.auth();

export default function RootNavigator() {
  const { user, setUser } =
useContext(AuthenticatedUserContext);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    // onAuthStateChanged returns an unsubscriber
    const unsubscribeAuth = auth.onAuthStateChanged(
      async (authenticatedUser) => {
        try {
          await (authenticatedUser
            ? setUser(authenticatedUser)
            : setUser(null));
          setIsLoading(false);
        } catch (error) {
          console.log(error);
        }
      }
    );

    // unsubscribe auth listener on unmount
    return unsubscribeAuth;
  }, []);

  if (isLoading) {
    return (
      <View style={{ flex: 1, justifyContent:
"center", alignItems: "center" }}>
        <ActivityIndicator size="large" />
      </View>
    );
  }

  return (
    <NavigationContainer>
      {user ? <HomeStack /> : <AuthStack />}
    </NavigationContainer>
  );
}
```

Colors.js

```
export const pageBackground = "#95B8D1";
export const inputColor = "#E8F1F2";
export const buttonColor = "#13315C";
export const buttonTextColor = "#E8F1F2";
```

App.js

```
import React from "react";

import Routes from "./navigation/index";
```

```
import "intl";
import "intl/locale-data/jsonp/en";

import {
  enGB,
  registerTranslation,
} from "react-native-paper-dates";
registerTranslation("en-GB", enGB);

export default function App() {
  return <Routes />;
}
```