Cover page

Faculty of Engineering & Information Technology

2024

Project title: Hotel booking management system

Academic Year: 5th year.

Group Members: Mohammad Moghrabi.

Mohammad Najjar.

Department Name: Computer Engineering.

Project Type: Software.

Supervisor Name: Dr. Suleiman Naeem Abu Kharmeh

Presented in partial fulfilment of the requirements for bachelor's degree in (Computer Engineering).

Table of Contents

Table Of Figures

Abstraction:

This project aims to address the growing demand for a seamless and efficient booking system by developing a comprehensive solution accessible through mobile platforms.

The project utilizes HTML5/CSS3 for structuring and styling, providing a visually appealing and user-friendly interface. JavaScript, coupled with React.js, empowers the system with dynamic functionalities, enhancing user interaction and experience. Asynchronous requests are facilitated through AJAX, ensuring smooth communication between the client-side and server-side components.

For mobile accessibility, React Native is employed, enabling the development of a native-like mobile application that shares the same codebase with the web counterpart, ensuring consistency across platforms and reducing development time and effort.

A restful API is built using ASP.net core to handle the business logic, data management, and integration with the database with SQL server as the database.

The admin web page was built using React.js where the admin can manage the application, A mobile application was built using React Native where the users can navigate through the hotels available and book the hotel they desire.

Introduction:

Every trip needs a lot of planning and most of it goes to plan where to stay and to find a good deal.

Book Wise helps find the best resident to stay in it could be a hotel, motel, apartment, villa, and more.

Book Wise also helps the service providers by allowing them to add their service easily to the system for the users to see and use.

Theoretical Background and Previous Work:

Booking applications are a will-know subject that has been implemented multiple times with different implementations for example www.booking.com.

Usually booking applications allow the user to book hotels and motels, which is functional but does not provide all the possible options for all the user's needs, that is why this idea came to mind where the user can look for any resident to book.

Methodology:

Standards and Specifications (Codes):

This project was built following the restful design pattern, and some other design patterns and standards.

The restful design pattern follows the REST standards building a REST API is not hard it is just following some patterns, it is just data modeling, which is a common skill in SOLID principles.

REST is an HTTP design pattern, it’s the correct way to implement HTTP. If implemented according to the REST design pattern HTTP is a client-server protocol with which clients can query and manipulate the resources of a server. For example, create a new resource using this URL. (Savalle, 2019).

Resources of the REST APIs can be anything cars, the content of webpages, databases, and much more, In the REST model they all have URLs to Create, Read, Update, and Delete (CRUD), and there is the HTTP verbs to do all the operations above. While building the URLs for the REST API there is a set of rules to follow to indicate the usage

of REST and to make the URL more readable and understandable (Reddy, 2018), these rules are:

- It is recommended to include API in the URL to indicate the usage of REST. For example, http://example.com/api/..., http://api.example.com/...
- Identify the API's version in the URL. For example, http://example.com/api/{API version}/{path for identifying a resource}, http://api.example.com/{API version}/{path for identifying a resource}
- Assigning a path for identifying resources, make sure not to include any verbs other than the HTTP verbs.

The REST API for this Project was Built using .net core clean architecture. The clean architecture is known for it is onion-like layers (Ltd.), it has four main layers which are infrastructure, domain, application, and web (API) layers, the names of these layers can vary but the purpose of each one is the same.

The layers in this architecture aim to reduce dependency which means more freedom for all components and improves application testability, maintainability, scalability, and flexibility.

Development process:

The project started by collecting some data from other existing booking applications to know how to handle our project and what to improve.

The database design came after that where we designed our database to be best fit to our project demands.

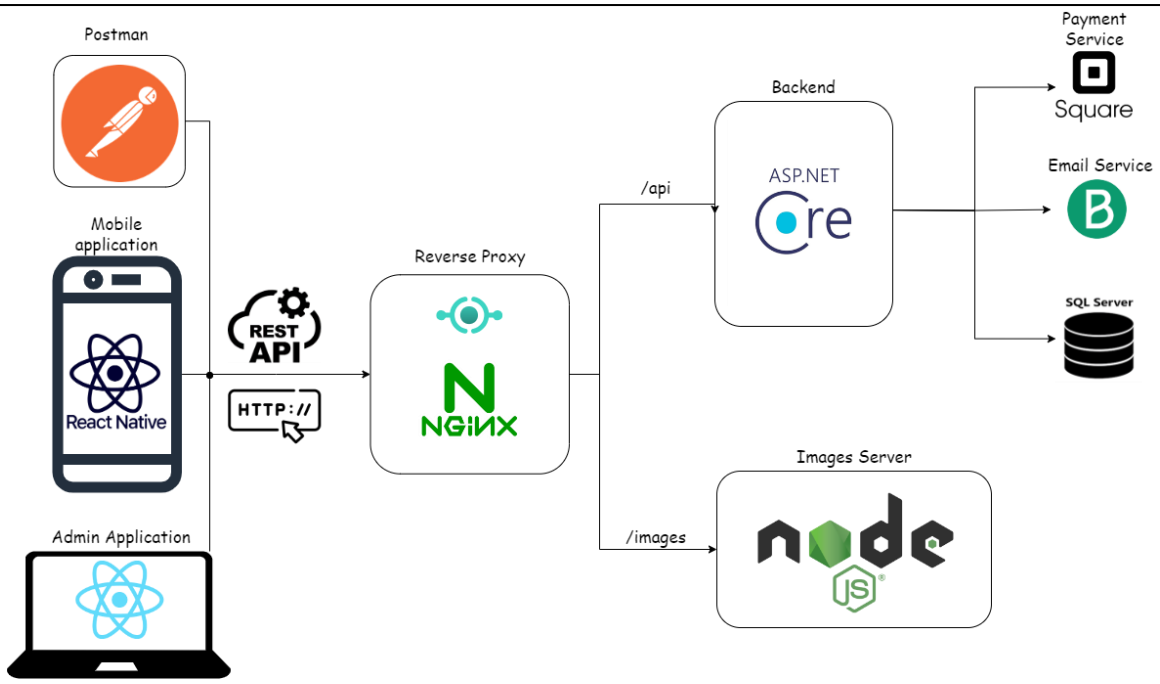


Figure 1

Figure 1 shows the application flow, with the tools necessary to implement our application in a will designed and structured manner.

REST APIs were used in this project to ensure simplicity, scalability and flexibility in our project.

A reveres proxy was made using nginx to map between the two servers we had the first one being the backend server with all the APIs and the second one being the images server where we convert the image path to a static URL.

The reverse proxy served another purpose to solve a problem we faced with visual studio discussed in detail in the constraints section.

The backend server was made using ASP.Net Core we used CQRS (Command Query Responsibility Segregation) to enhance our project's Separation of Concerns, Scalability and Performance.

CQRS is a design pattern that separates the reading (query) and writing (command). A command represents an action or change in the system's state. Commands are typically used to create, update, or delete data. Query represents a request to read or retrieve data from the system, queries do not change the state of the system; they only return data.

CQRS also helps with creating a simplified code base, Flexibility in Data Storage, and Event Sourcing which provides a full audit log of changes, making it easier to debug and replay events to reconstruct system state figure 2 shows how the CQRS works.

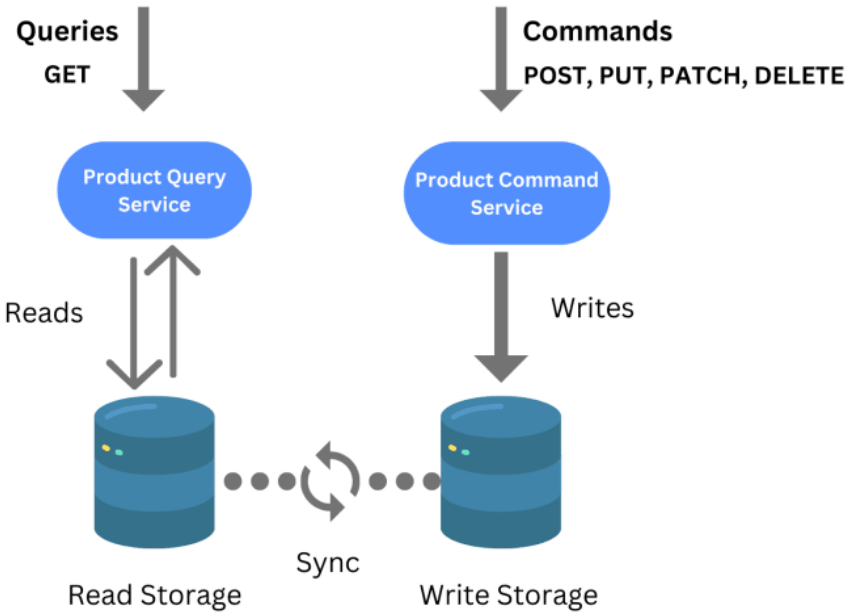


Figure 2

Three main external services were used square payments for payment service integration, Brevo for email service integration, and SQL server database as our system's database.

We used Entity Framework Core (EF Core) is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology (Microsoft., n.d.).

EF Core has a lot of benefits some of them are its cross-platform, Light weight, LINQ support, and has migration which allowed as to easily update the database.

Our APIs were developed following the REST design pattern and the clean architecture principles we also followed HATEOAS (Hypermedia As The Engine Of Application State) in returning the requests from the API.

Clean architecture is an onion like layered architecture with each layer being responsible for a specific part of the project with a set of rules to define how these layers communicate with each other to build a loosely coupled and will separate project figure 3 shows the layers of the clean architecture.

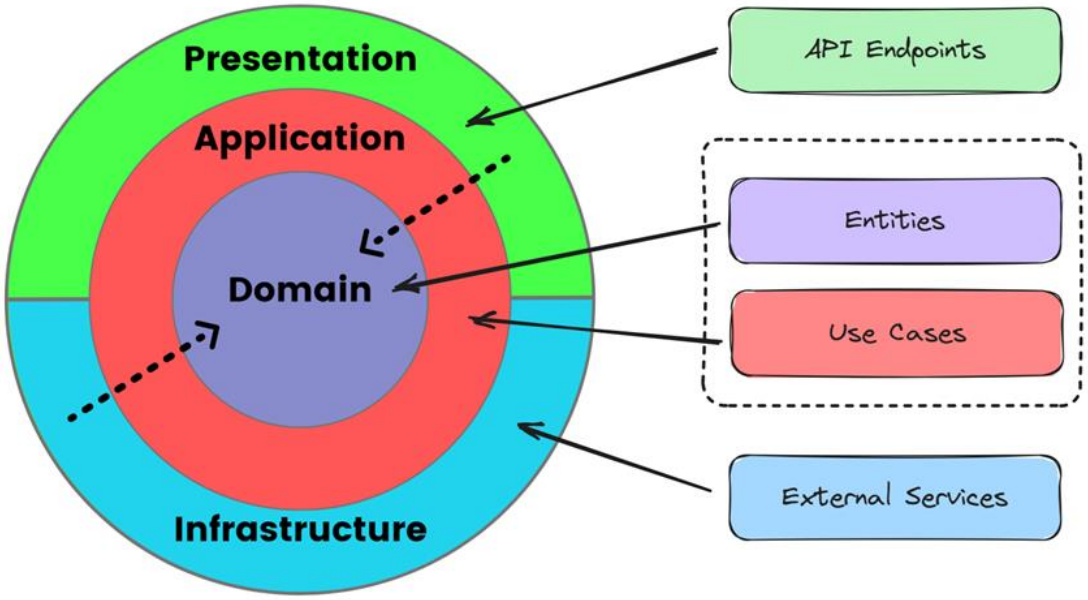


Figure 3

HATEOAS is a constraint of the REST application architecture. It ensures that a client interacts with the application entirely through hypermedia provided dynamically by application servers. This means that the client does not need to include hard coded links where the server drives the interactions figure 4 shows an example if the responses in our system.

```
"pagination": {
    "prevPage": null,
    "nextPage": "http://localhost:5092/api/hotel?page=2&pageSize=30",
    "count": 3
},
"hotels": [
    {
        "hotelId": "3858fa5b-9e7d-4cfd-48e6-08dc93dbc91c",
        "hotelName": "lalaland",
        "hotelDescription": "The best hotel ever.",
        "rating": 3.0,
        "links": [
            {
                "rel": "hotel type",
                "href": "/api/hotelType/3858fa5b-9e7d-4cfd-48e6-08dc93dbc91c",
                "method": "GET"
            },
            {
                "rel": "location",
                "href": "/api/hotel/3858fa5b-9e7d-4cfd-48e6-08dc93dbc91c/location",
                "method": "GET"
            },
            {
                "rel": "hotel images",
                "href": "/api/hotel/hotelImage/3858fa5b-9e7d-4cfd-48e6-08dc93dbc91c",
                "method": "GET"
```


Figure 4

JWT Bearer Token were used for authentication and authorization, to ensure the safety of the user's data. Role based authorization was used to separate the two roles we had in our system users and admins.

Finally, we used XUnit.net to test some of our features the most important of them being the payment feature where we wanted to ensure that the service integration was correct and because the payment processes being a sensitive operation needs to work as expected we added transaction to ensure that neither the user nor the service provider could have any problem with the payments.

Then the process of making the frontend started by creating a React project for the admin page where Vite was used to create this project.

The mobile application was built using React Native for the users, to interact with our server side.

The admin page is built using JavaScript, React, and Material-UI figure 5 to figure 13 shows the admin UI pages, and figure 14 to figure 25 show the user's UI pages.

Admin UI Pages.

Here the admin can add, delete and updates the information's about the stays and the rooms of it.

Admin login page: at first the admin should log in through Email and password.

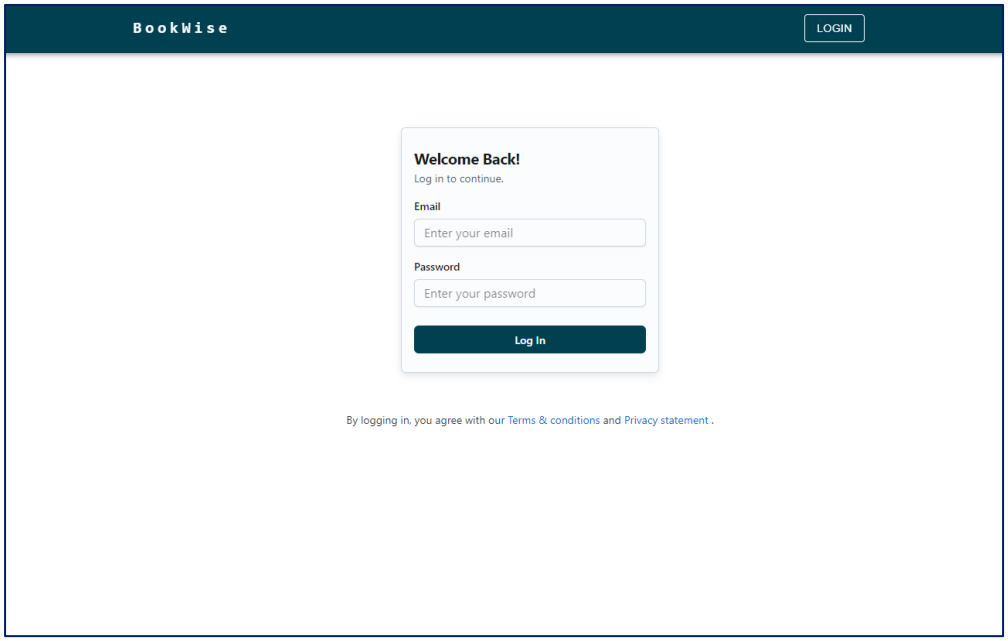


*Figure **5***

Hotel Management Page: here the admin can search between hotels, add new hotel, and can update or delete existing one.

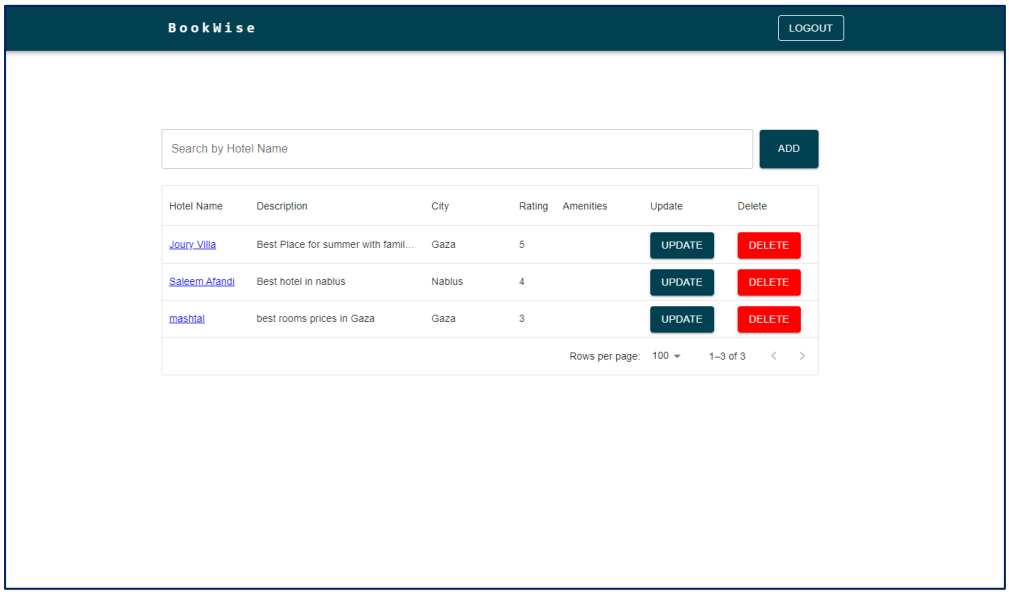


*Figure **6***

When click on ADD button there is a slide window appears to fill the information's about the new hotel such as name, Description, rate, Amenities, Type of stay (like hotel, resort, villa, …etc.), city name, street, postal code, and image of the hotel.

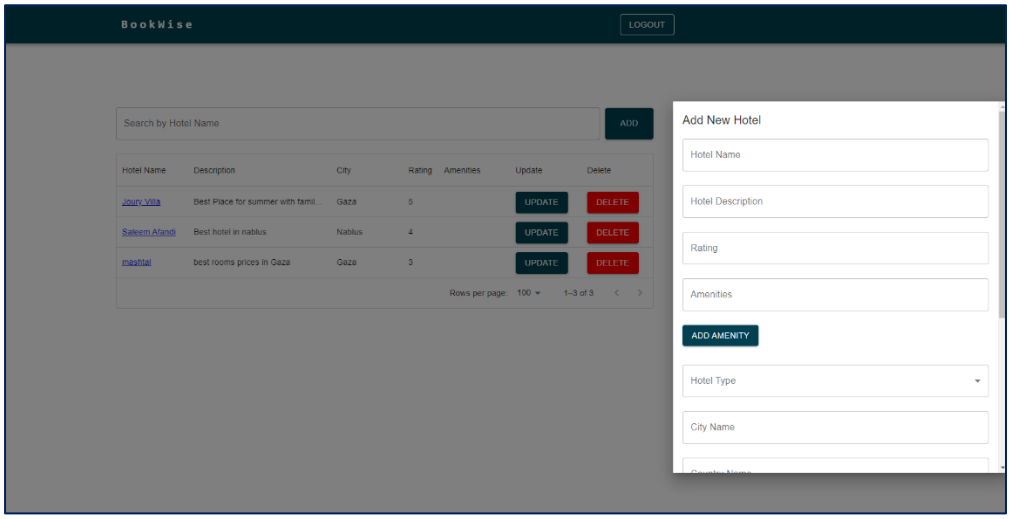


Figure 7

The admin can also updates or deletes existing hotel using UPDATE and DELETE buttons of each hotel hence when admin clicks on delete there is confirmation message appears to let admin knows that every thing about hotel will permanently deleted and when click confirm the hotel

will be deleted and when click on UPDATE there is a slide window appears with the current information's about hotel and the admin can change these information and click SAVE so the hotel will be updated

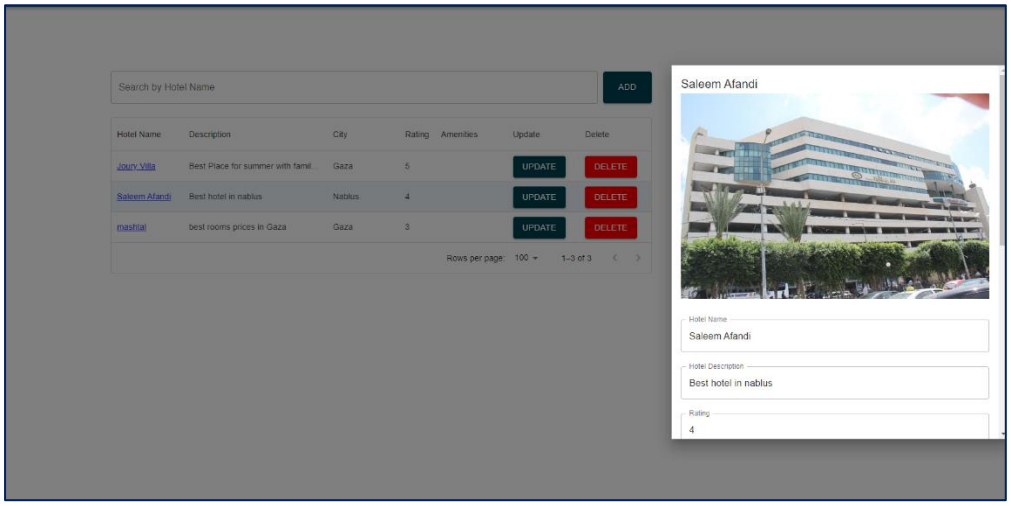


Figure 8

Rooms Management Page: here the admin can search between rooms of the stay through room number, add new room, update or delete existing room, and add new discount.

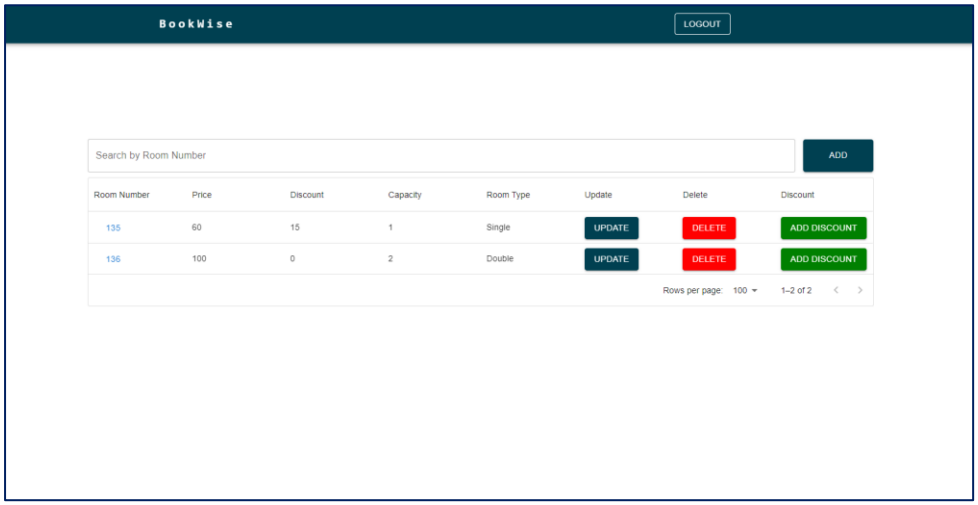


Figure 9

When click on ADD button there is a slide window appears to fill the information's about the new room such as room number, price, capacity, room type, and images of the room.

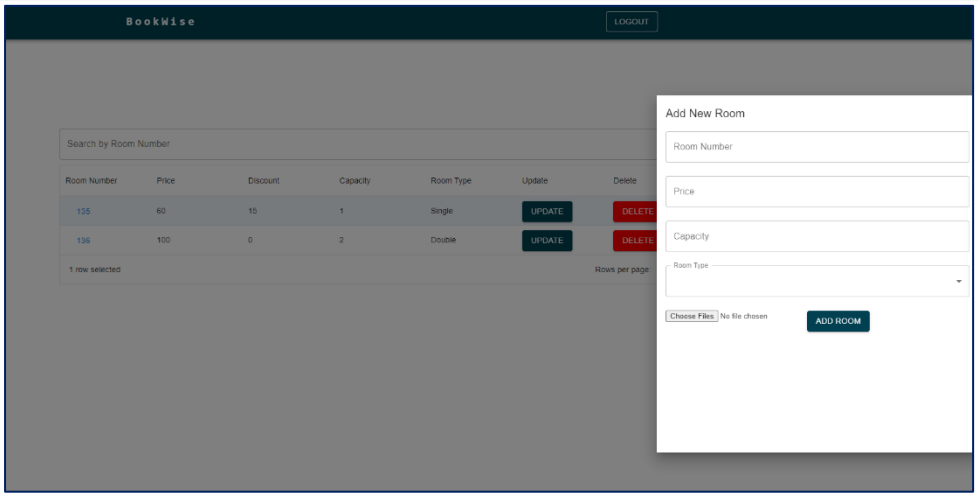


Figure 10

The admin can updates or deletes existing hotel using UPDATE and DELETE buttons just like in hotels but here the admin can also add discount for room and can add up to 5 images for room.

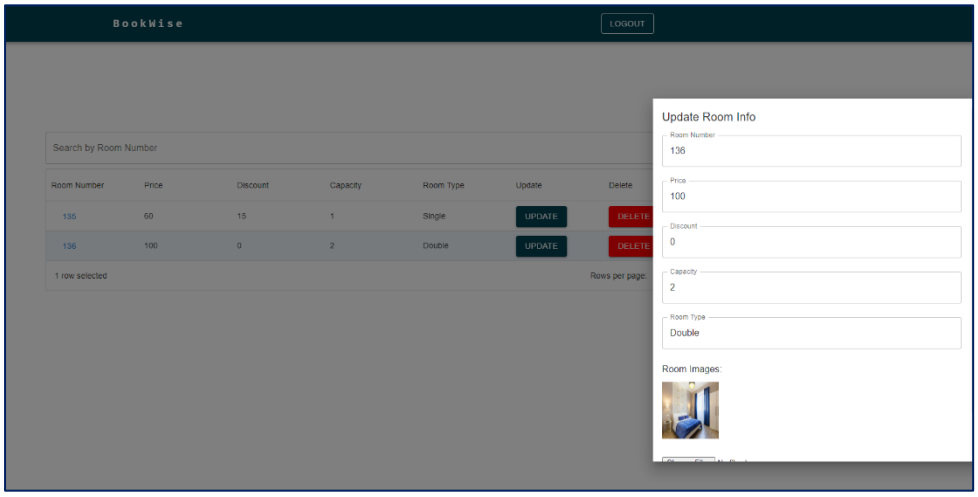


Figure 11

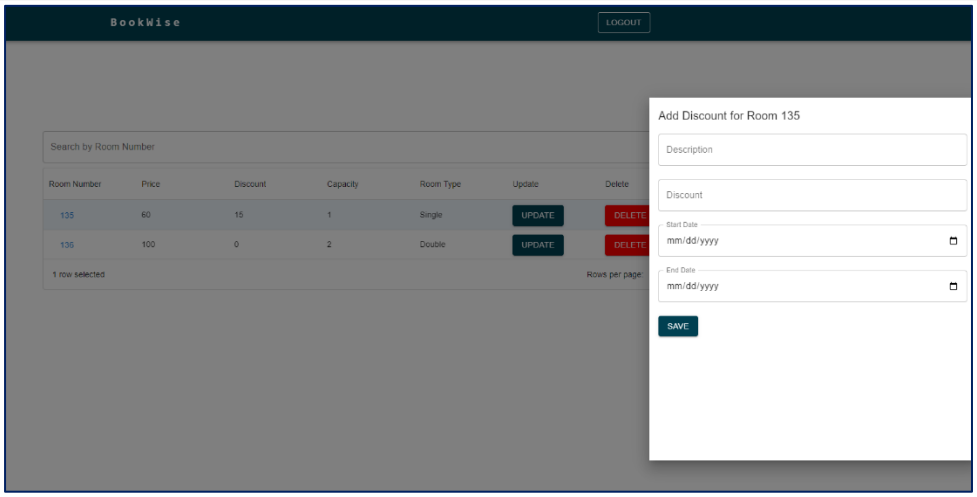


Figure 12

The admin can see the history of bookings of the room if click on room number there is a slide window appears and displays information about user who booked the room and the start/end date of book.

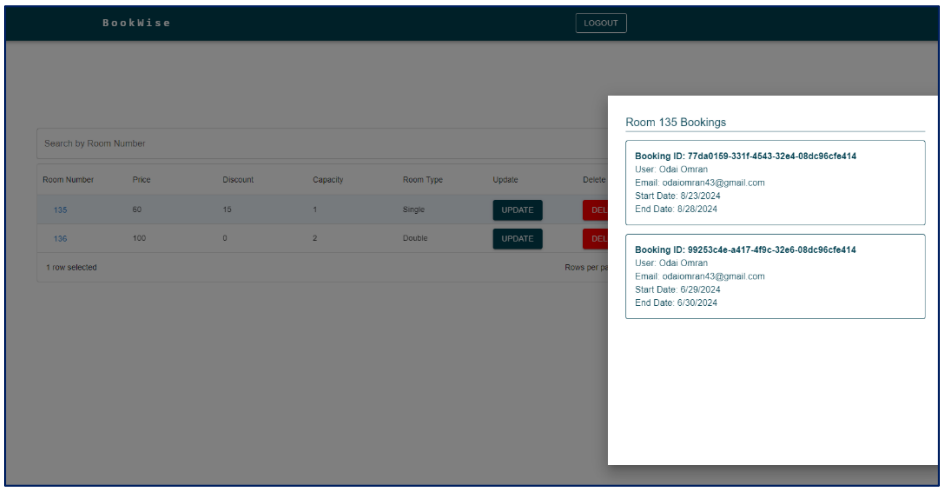
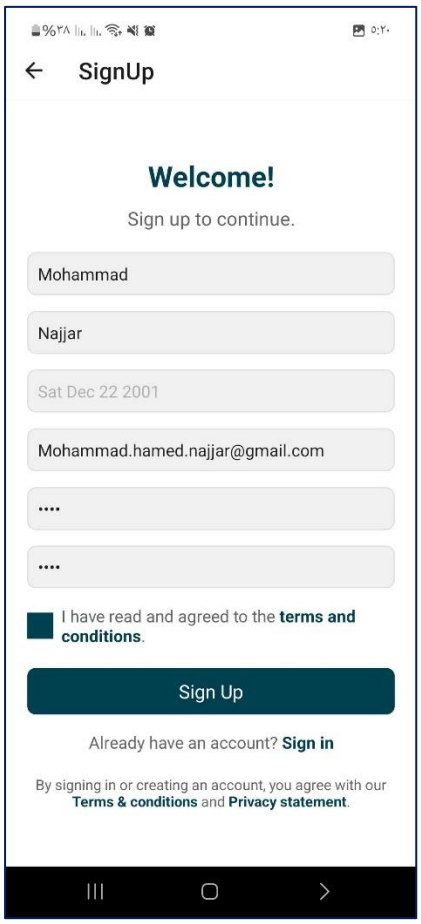


*Figure **13***

User UI Pages.

Here the user can register, reset his password if forgot, search for stays, see the best deals and latest visited hotel, add rooms to his cart, and pay for book the items of his cart.

User Registration Pages: the user can register through his email and some basic information's such as first name, last name, birthdate, and create password the user can log in through his email and password. If the user forgot his password, he can reset it by sending verification code to his email then he can reset his password figures 14, 15, and 16 shows the UI .



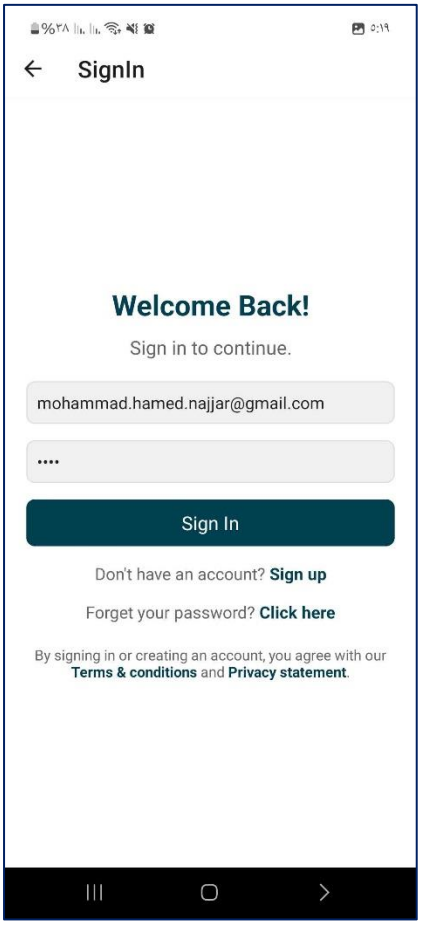


Figure 14

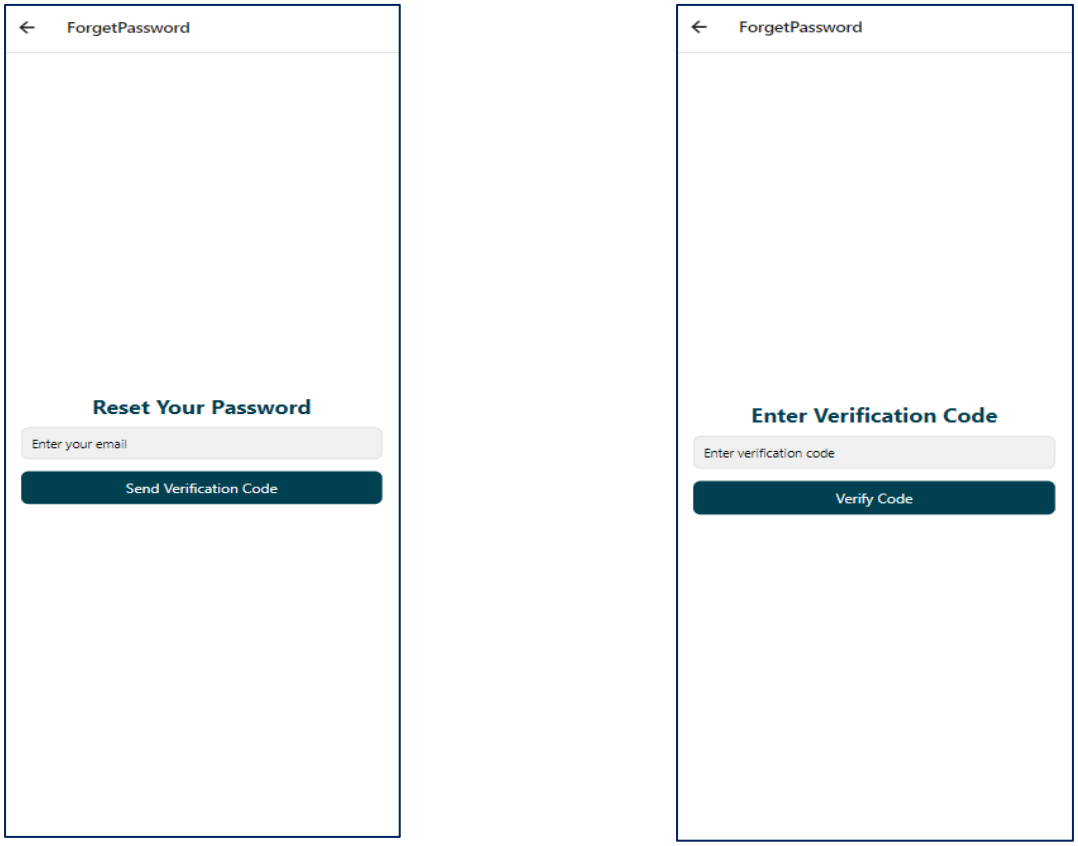

ForgetPassword
Reset Your Password
Enter your email
Send Verification Code
ForgetPassword
Enter Verification Code
Enter verification code
Verify Code

*Figure **15***

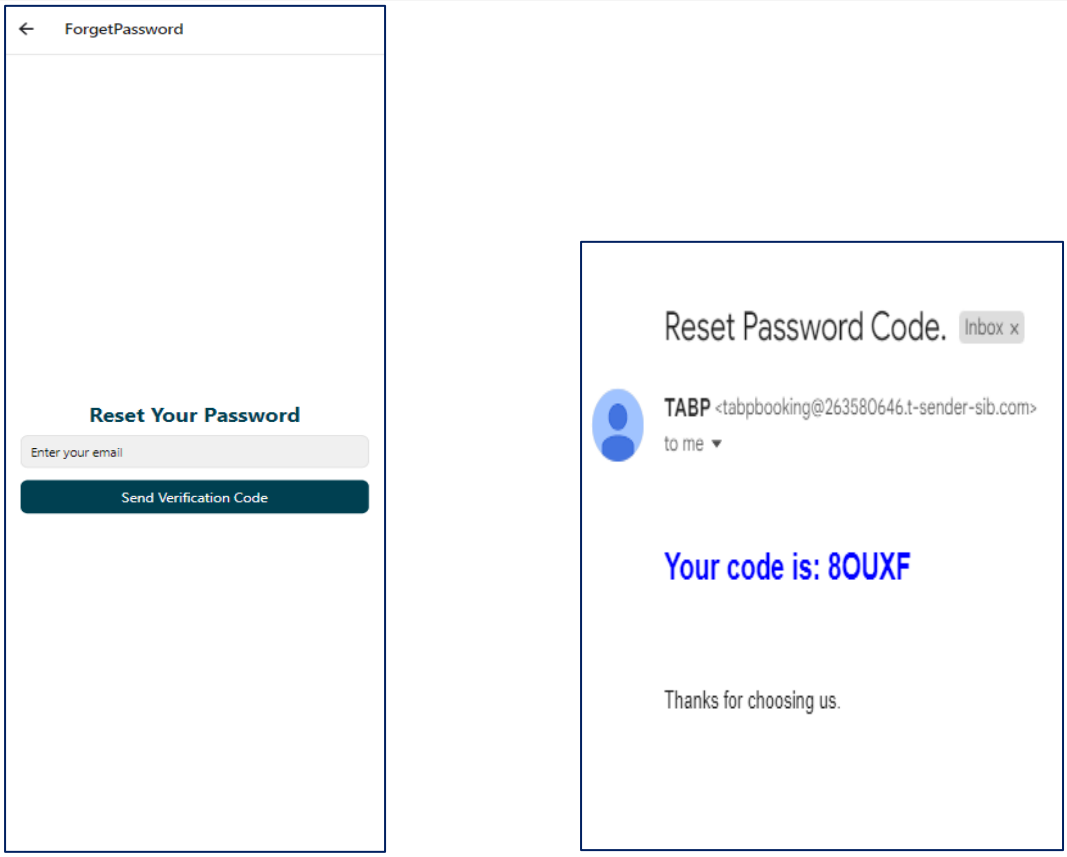


*Figure **16***

Before user register he can read the Terms & Conditions, Privacy Statement of the application

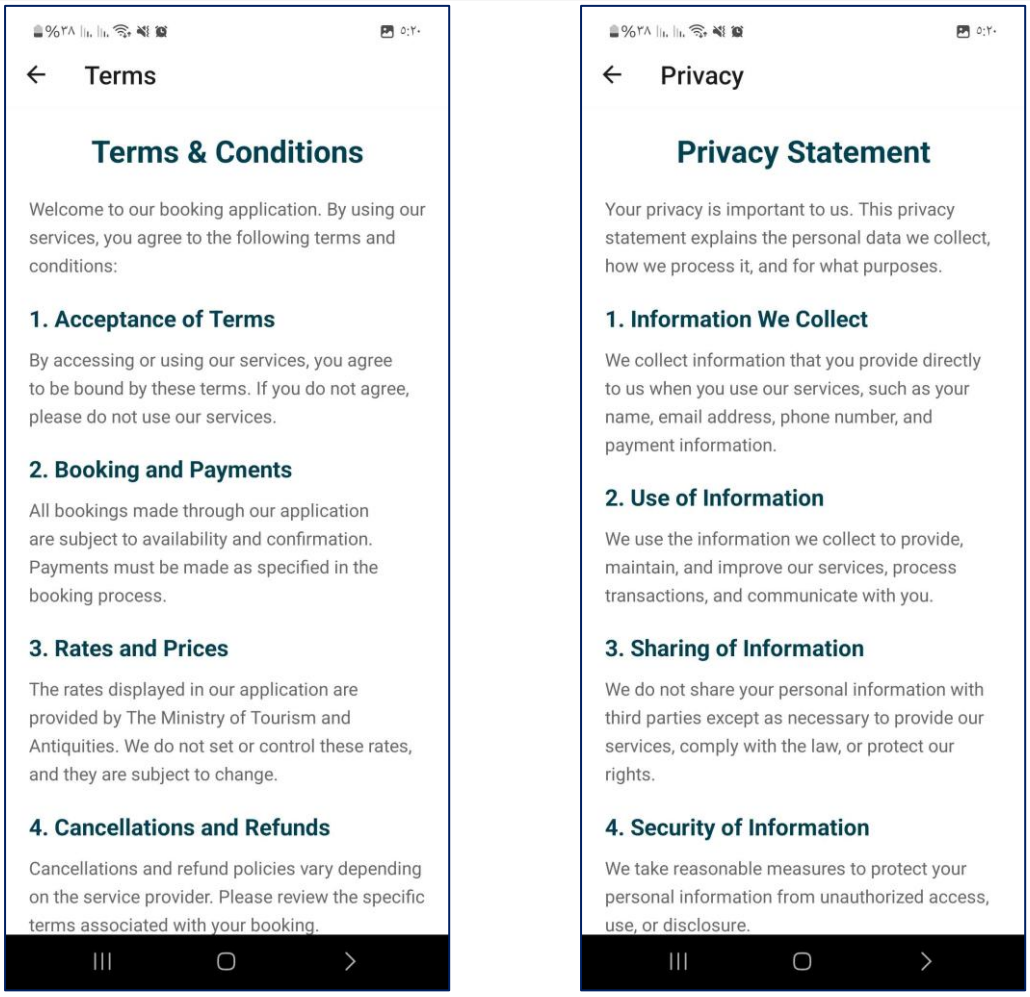
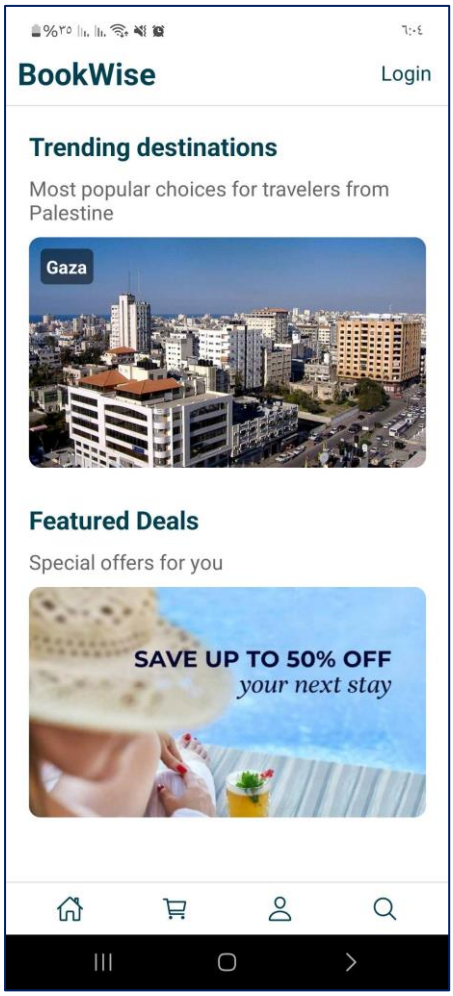

Terms
Terms & Conditions
Welcome to our booking application. By using our services, you agree to the following terms and conditions:
1. Acceptance of Terms
By accessing or using our services, you agree to be bound by these terms. If you do not agree, please do not use our services.
2. Booking and Payments
All bookings made through our application are subject to availability and confirmation. Payments must be made as specified in the booking process.
3. Rates and Prices
The rates displayed in our application are provided by The Ministry of Tourism and Antiquities. We do not set or control these rates, and they are subject to change.
4. Cancellations and Refunds
Cancellations and refund policies vary depending on the service provider. Please review the specific terms associated with your booking.
Privacy
Privacy Statement
Your privacy is important to us. This privacy statement explains the personal data we collect, how we process it, and for what purposes.
1. Information We Collect
We collect information that you provide directly to us when you use our services, such as your name, email address, phone number, and payment information.
2. Use of Information
We use the information we collect to provide, maintain, and improve our services, process transactions, and communicate with you.
3. Sharing of Information
We do not share your personal information with third parties except as necessary to provide our services, comply with the law, or protect our rights.
4. Security of Information
We take reasonable measures to protect your personal information from unauthorized access, use, or disclosure.

Figure 17

User Home Page: here the user can see most visited cities (the frame changes every 5 seconds to show next city), navigates to Featured Deals page, and see his latest visited hotel if he logged in, in guest mode the user can search any hotel and see his rooms, but cannot navigate to profile page, and cannot add anything to cart until log in.

Guest Mode



Home Page Mode

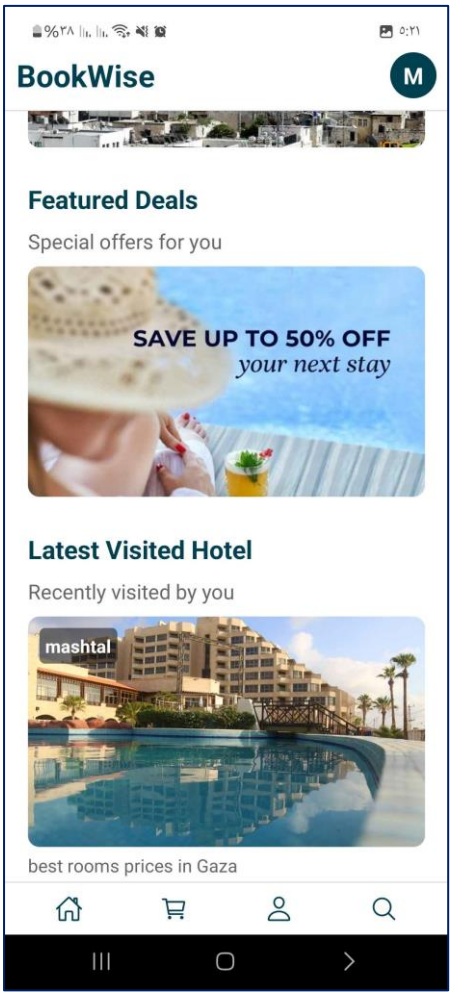


Figure 18

If user press on latest visited hotel he will directly navigates to that hotel page and shows the full info about it and its rooms.

Featured Deals Page: here the user explores the best deals for rooms from different stays and he can add any room to his cart.

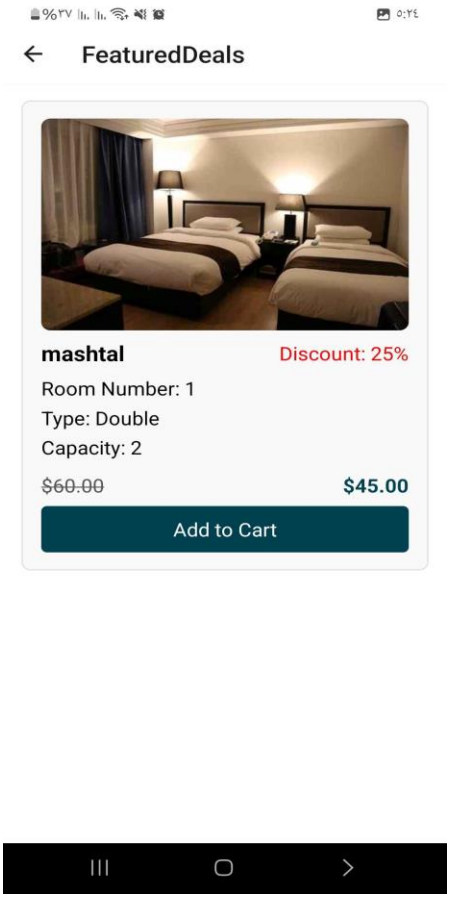


Figure 19

Search Page: here the user can search stays using some filters like name, Rating, Min/Max price, …etc., then when click search he will see the results of search and brief info for each result.

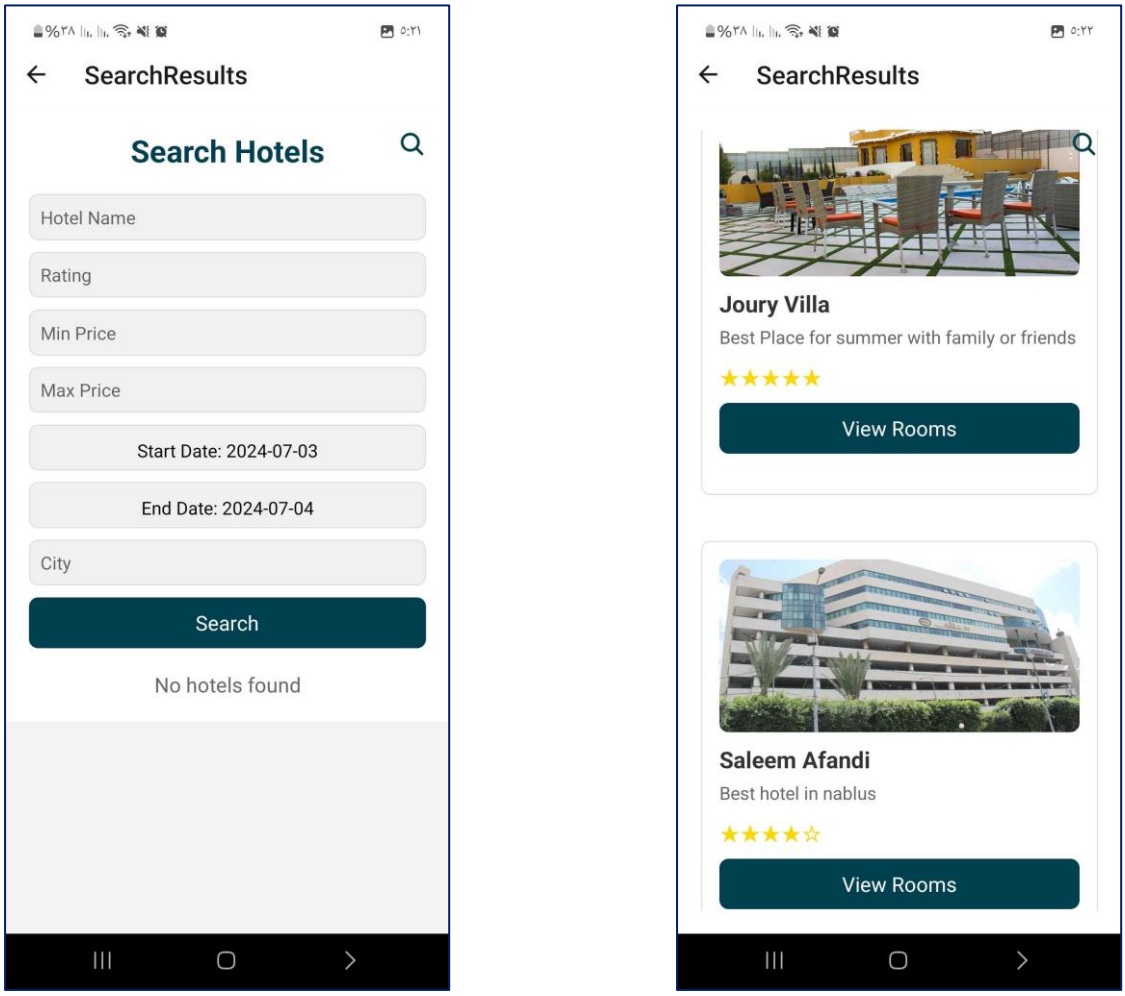


Figure 20

Rooms Page: if user want to see full information's of stay and explore its rooms so when press on View Rooms he will navigates to Rooms Page, in the Rooms Page the user can see more details about stay and its location. The user can also see the free rooms and add it to his cart.

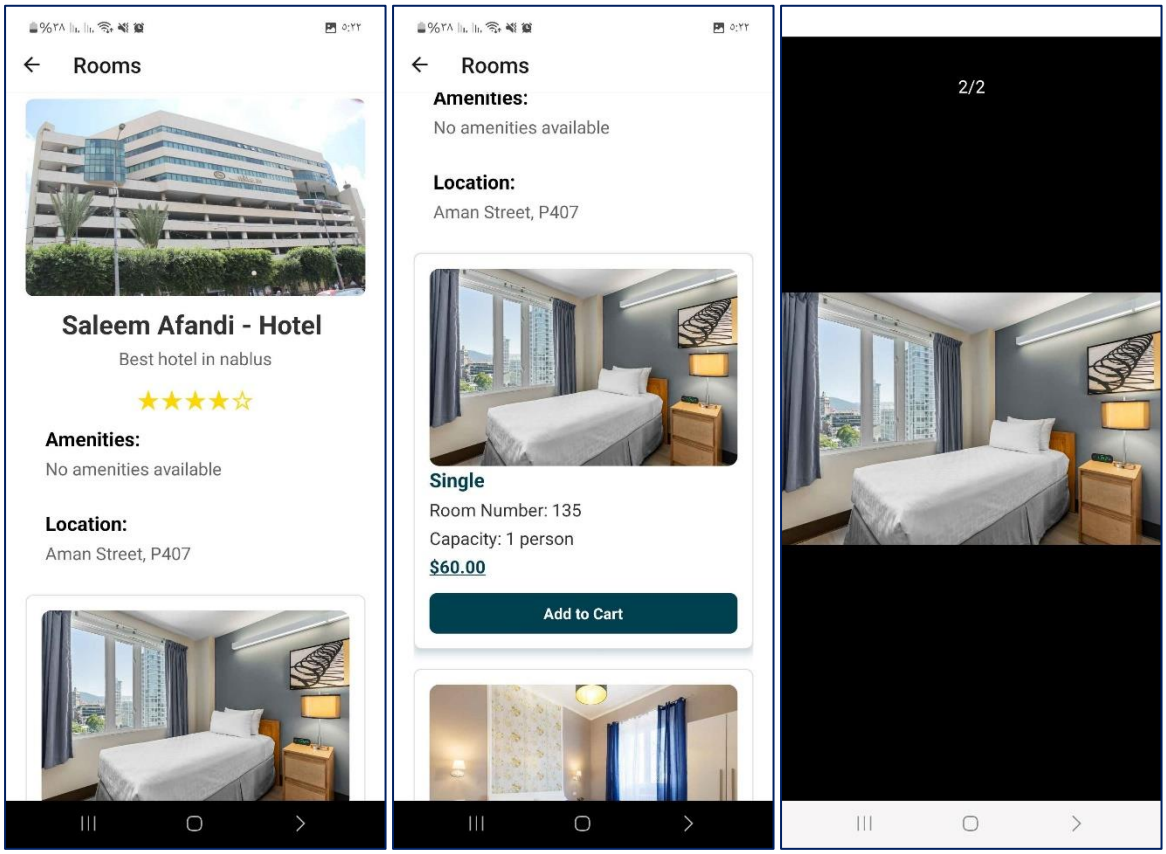


Figure 21

Add Room To The Cart Process:

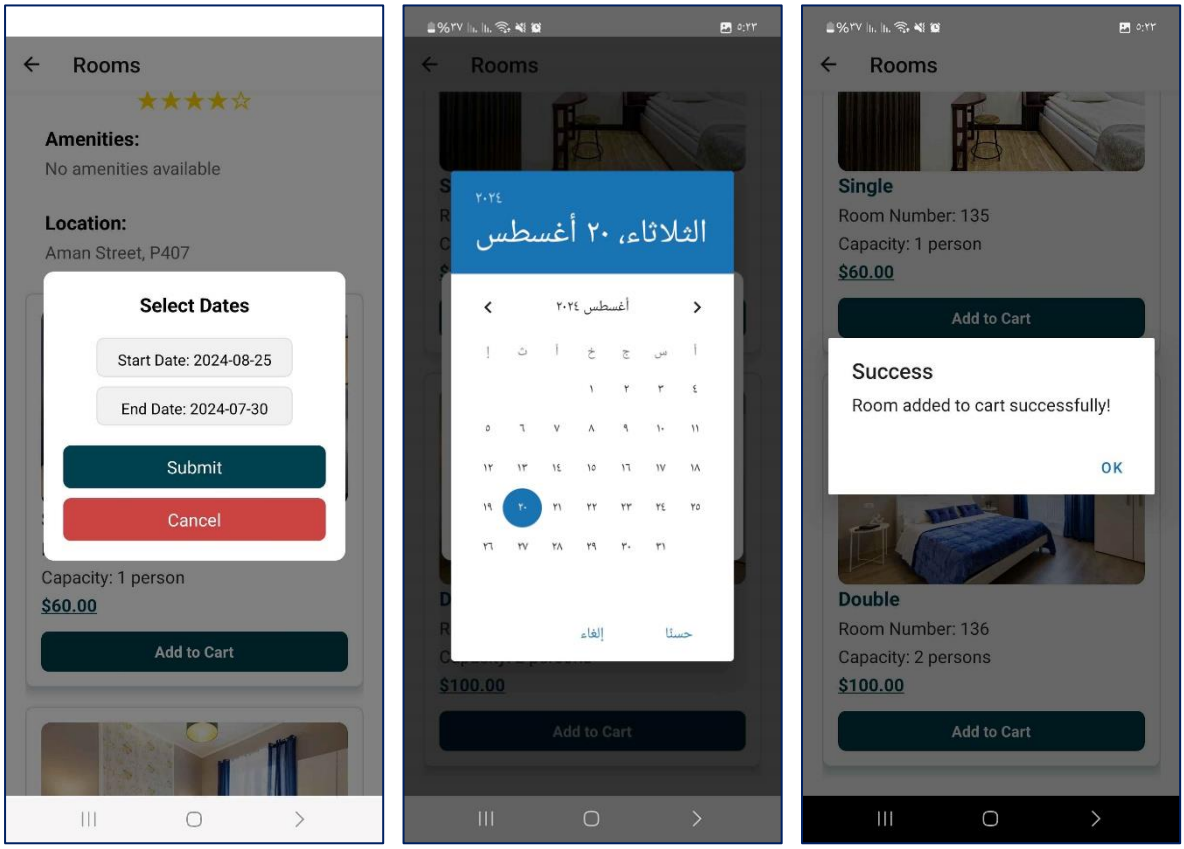
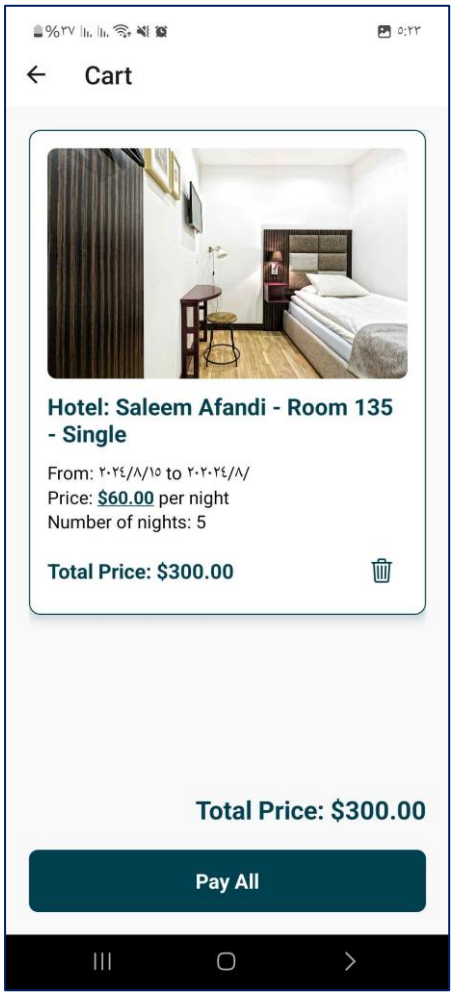
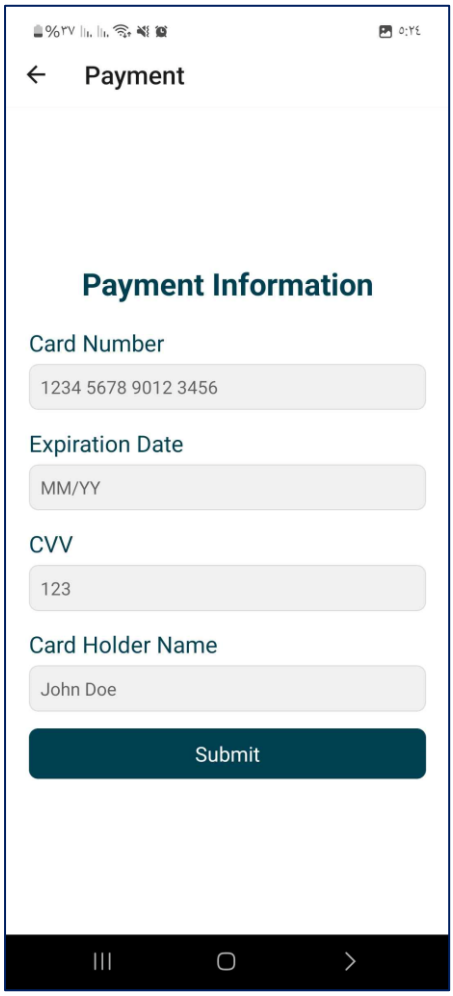


*Figure **22***

Cart Page: here the user can see items in his cart and can delete any item, then he can press on pay all to navigates to Payment page and when click on Submit the user will book all rooms in his cart then if payment success the user will receive invoice email.





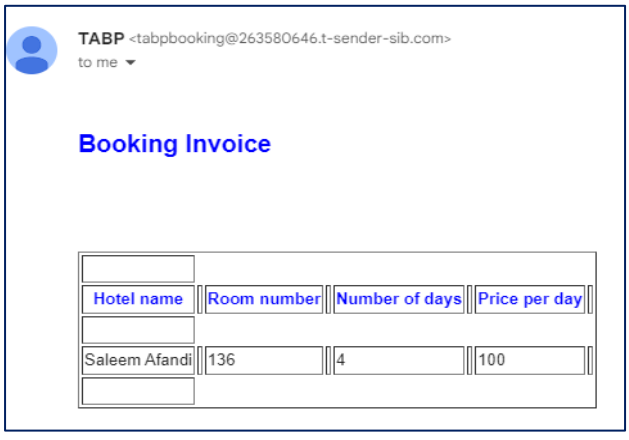


Figure 23

Finally, the user can logout from Profile Page through log out button

Figure 24

Constraints:

The project development process was not perfect hence we faced some constraints that prevented us from reaching the desired results such as technological limitations and time constraints.

1- Technological Limitations: We encountered problems working with Square for payment integration due to version differences. Additionally, the app was unable to perform updates effectively.

   Another technical issue we had was running our React Native application on the mobile the problem happened because visual studio 2022 did not allow any other devices other than the one running the server to access the API endpoints, we had two solutions for this problem either to deploy the application which needed a lot of time which we did not have.

   Another solution was to create a reverse proxy. We ended up making a reverse proxy that also had another advantage to map between our two servers.

2- Time Constraints: The development timeline was limited because of the demands of other university courses and their associated projects.

Results and Analysis:

In this project, we were able to build a booking system for different kinds of residents and include suggestions for trending destinations according to our system.

An ASP.net core API was built using clean architecture and used SQL server database to store and manage our data which includes, residents' data, rooms data, user data, location data, cities data, featured deals, and reviews, and images for residents, rooms, and cities.

Our system includes ratings for the hotels where each hotel will have it is rating according to some organizations that specialize in ranking the hotels more in hotel ratings can be found in (SiteMinder.).

For finding trending destinations our system uses the data collected from the user's bookings in the system and chooses the top locations of the bookings.

Discussion:

The main features of our project like booking searching and suggestions were correctly implemented and had the expected results, but some of the other features did not have the correct output like the hotel rating feature, and the payment feature.

Our system has implemented the functionality of having different resident types by adding hotel and room type entities which determine what is the resident type to book.

Each of the residents in our system has a list of amenities for the user to choose what is the best resident fitting for them.

Conclusion:

The idea of Book Wise came from the hard time people take to find nice places to go to especially in Palestine, where young like us can't find places to go out with friends, our system should solve this problem by making it possible to book villas and other types of entertainment places.

The system we built has the functionalities and features to solve the problem above, although comparing the result we had with the expected results in mind it did not include all the possible features and functionalities we intended to provide.

References:

Ltd., P. S. (n.d.). *Clean Architecture in .NET Core: The Complete Guide.* Retrieved from Positiwise. : https://positiwise.com/blog/clean-architecture-net-core

Reddy, R. (2018, June 17). *How to design URL to REST resource.* Retrieved from Java Guides.: https://www.javaguides.net/2018/06/how-to-design-url-to-rest-resource.html

Savalle, P. (2019, August 8). *REST API design as a craft, not an art.* Retrieved from medium.com: https://medium.com/@patricksavalle/rest-api-design-as-a-craft-not-an-art-a3fd97ed3ef4

SiteMinder. (n.d.). *Hotel star rating systems explained: How do they work?* Retrieved from SiteMinder.: https://www.siteminder.com/r/hotel-star-rating-systems/