NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

KNOWLEDGE * CHARACTER * UNITY

MINI PROJECT REPORT

on

HOTEL ROOM SEARCHING APP

Submitted in partial fulfilment of the requirement for the award of Degree of

Bachelor of Engineering

In

Information Science and Engineering

Submitted by:

Ananya Havinal	1NT22IS015
Chaithanya M	1NT22IS039
B Swathi	1NT22IS034
Dhruthi S L	1NT22IS049

Under the Guidance of

Mr. Prashanth B S

Assistant Professor, Dept. of ISE, NMIT

NITTE EDUCATION TRUST

Department of Information Science and Engineering
(Accredited by NBA Tier-1)

2024-2025

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

Department of Information Science and Engineering

(Accredited by NBA Tier-1)

CERTIFICATE

This is to certify that the Case Study Report on **"Hotel Room Searching App Using Flutter "**is an authentic work carried out by **Ananya Havinal(1NT22IS015), Chaithanya M(1NT22IS039), B Swathi (1NT22IS034), Dhruthi S L (1NT22IS049)** Bonafede students of Nitte Meenakshi Institute of Technology, Bangalore in partial fulfilment for the award of the degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2024-2025**.** It is certified that all corrections and suggestions indicated during the internal assessment has been incorporated in the report.

HAD Faculty

Mr. Prashanth B S

Assistant Professor, Dept. ISE,
NMIT Bangalore

Abstract

In the digital era, seamless and efficient hotel booking has become a crucial aspect of travel and hospitality services. This project presents a **Hotel Room Searching App** developed using **Flutter**, a powerful open-source UI toolkit by Google, aimed at providing cross-platform mobile applications with native performance. The app enables users to search, filter, and book hotel rooms based on various parameters such as location, check-in/check-out dates, price range, room type, and amenities.

The application leverages **Firebase** for backend services including authentication, real-time database, and cloud storage, ensuring secure and dynamic data management. It features an intuitive and responsive user interface, making the user experience smooth and accessible on Android platform. Key functionalities include user login/signup, interactive map integration, detailed hotel listings with images and reviews, booking confirmation, and user profile management.

By utilizing Flutter's widget-based architecture and state management techniques such as Provider or Riverpod, the app ensures efficient rendering and a responsive experience. This app serves as a scalable solution for travelers and hotel businesses, demonstrating the potential of modern mobile technologies in enhancing the travel industry.

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. I express my sincere gratitude to our Principal **Dr. H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing facilities.

We wish to thank our HoD**, Dr. Mohan S. G.** for the excellent environment created to further educational growth in our college. We also thank him for the invaluable guidance provided which has helped in the creation of a better project.

I hereby like to thank our ***Mr. Prashanth B S, Assistant Professor*** Department of Information Science & Engineering on their periodic inspection, time to time evaluation of the project and help to bring the project to the present form.

Table of Contents

List of Figures

Chapter-1

INTRODUCTION

The aim of this project is to offer travelers an authentic and transparent experience when searching for hotel rooms. Unlike generic listings, our app provides genuine insights and firsthand recommendations from fellow travelers and locals who have personally stayed at the hotels. Users can explore real-world reviews both positive and negative about their planned destinations, empowering them with trustworthy information. This approach filters out hype and misleading marketing, delivering a reliable source of truth. Ultimately, the app helps users make well-informed decisions, build trust in their booking choices, and avoid unexpected surprises.

Flutter Backend Architecture

The app is structured in a layered architecture similar to Clean Architecture, with Flutter handling the UI and a backend (like Firebase or a REST API) serving data and business logic:

Figure 1.1 – Flutter-Backend Architecture

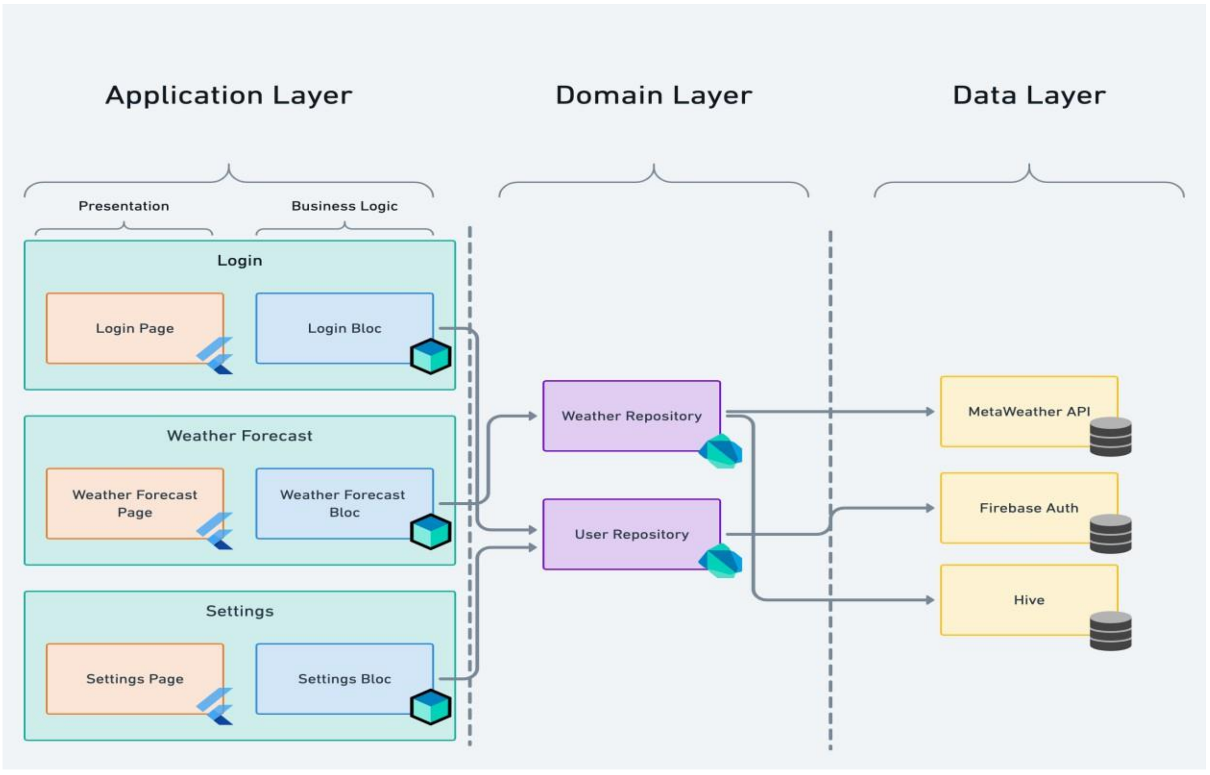


Figure 1.1 Flutter-Backend Architecture

1. Presentation Layer

- **UI Components**: Flutter screens (search form, hotel list, booking flow) are structured into reusable widgets for consistency and maintainability.
- **State Management**: Use Bloc, Provider, or Riverpod to handle UI state and events, keeping widgets “dumb” and logic-free.
- **Clean Separation**: Presentation delegates business logic to the Application layer and observes state changes without direct data handling.

2. Application Layer

- **Use Cases / Services**: Implements actions such as SearchHotels, BookRoom, SubmitReview, orchestrating domain workflows based on user commands.
- **Input Validation & Error Handling**: Validates user inputs (e.g., dates, guest count), manages errors, and transforms results for UI consumption .
- **Framework-Agnostic Logic**: Encapsulates control flow between Presentation and Domain, keeping UI and data layers decoupled

3. Domain Layer

- **Entities**: Core models like Hotel, Booking, Review, containing business rules.
- **Use Cases / Interactors**: Represent business actions, using repository interfaces to execute operations and return domain entities.
- **Abstraction & Testability**: Independent of Flutter or backend tech ideal for unit testing and core logic stability

4. Data Layer

- **Repository Implementations**: Provide concrete implementations of Domain interfaces (e.g., HotelRepository), hiding data source details.
- **Data Sources**: Fetch and transform data from Firebase, APIs, or local caches; map JSON/Firestore data to domain entities via DTOs.
- **Flexible Caching & Switching**: Enables seamless swaps between remote and local sources, with unified error handling and mapping logic.

Chapter-2

OBJECTIVES

 To build a secure database that stores:

- User login credentials, profiles, and booking history
- Hotel information: metadata, room types, amenities, images, real-time availability and pricing
- User reviews and ratings (genuine feedback from travelers)

 To design a dynamic, mobile-first frontend in Flutter that is:

- Crisp, responsive, and intuitive
- Includes key screens: Search, Results List, Hotel Details, Booking Summary, and Profile
- Employs state management for smooth, testable UI flows

 To implement real-time search and booking functionality that:

- Queries hotel availability dynamically with accurate filters.
- Reflects real-time availability updates and prevents booking errors/double bookings.
- Enables secure payment integration.

 To integrate a transparent review system, enabling:

- Submission and moderation of real reviews with ratings.
- Display of balanced feedback (positive & negative) to build user trust.

 To ensure seamless end-to-end flow, by:

- Connecting frontend, application logic, domain entities, and backend/data layer.
- Ensuring proper navigation, error handling, responsiveness, and performance.
- Including user registration, booking flow, review posting, and account management.

Chapter-3

PROBLEM STATEMENT

This project, developed using Flutter, aims to create an efficient and user-friendly hotel room searching app that provides insights into hotel availability, pricing, and user preferences across various locations. The app will identify the top hotel categories preferred by users, highlight the cities with the highest hotel listings and bookings, and track seasonal booking trends to understand peak demand periods. Additionally, it will enable comparison of hotels based on user ratings and amenities to assist users in making informed choices. By leveraging Flutter's cross-platform capabilities, the app seeks to deliver a seamless and interactive experience that simplifies the hotel search and booking process for users across multiple devices.

Chapter-4

SYSTEM DESIGN

System design talks about the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. Quality system design is essential in developing functional and lasting applications.

4.1 Architecture Design

A system's architecture outlines its main parts, their connections (structures), and how they work together. There are several contributing variables to software architecture and design, including business strategy, quality attributes, human dynamics, design, and IT environment.

Figure 4.1 below shows the project's architectural design.

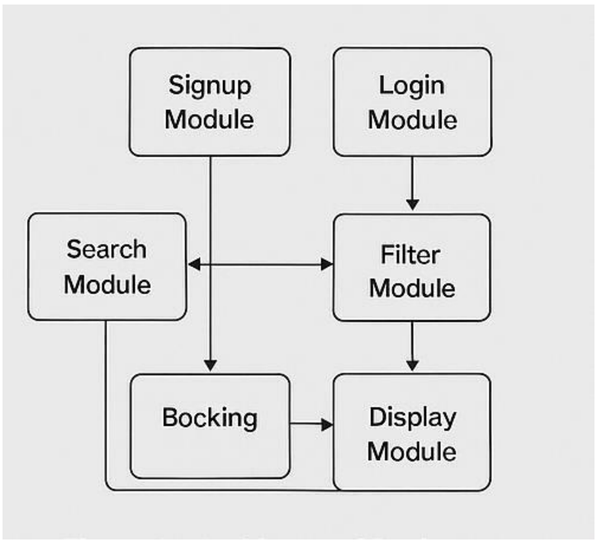


Figure 4.1 Architectural Design

1. **Signup Module: Allows new users to register using email and password (Firebase Auth or REST API).**
2. **Login Module: Handles authentication and session creation.**
3. **Search Module: Provides search functionality by city, hotel name, or current location.**
4. **Filter Module: Offers filtering by price, rating, amenities, etc.**
5. **Booking Module: Lets users book a selected room and confirm details.**

6. **Database Module: Stores user info, hotel listings, and bookings (Firestore or MongoDB).**
7. **Display Module: Renders hotel cards, filters, booking status, and maps using Flutter widgets.**

4.2 Software Architecture Block

A visual depiction that depicts the actual physical implementation of a software system's components is called an architecture diagram. It displays the relationships, constraints, and boundaries between each piece as well as the overall structure of the software system.

Figure 4.2 shows the project's software architecture block

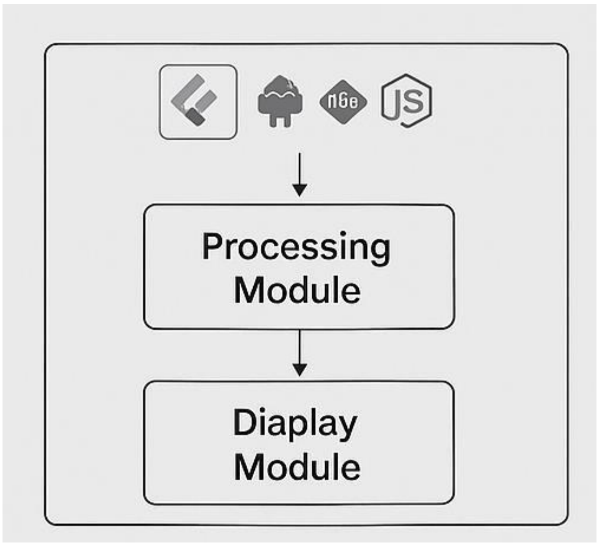


Figure 4.2 SAB

1. **Input Module:** This module has components that takes input from the user. Text inputs with coordinates of the pin on the map, title, type, ratings.
2. **Processing Module**: MERN based processing is done where React framework handles the states for UI rendering. Express has multiple endpoints for handling different functionalities which in turn communicates with the database to perform the specified operation. Processing module communicates with the DB to retrieve and write information.
3. **Display Module:** This module communicates with the Processing module by monitoring states and changing specific components on the viewport as text and Pins on a map

4.3 Flowchart

An algorithm is graphically represented by a flowchart. It is frequently used by programmers as a technique for planning programs to address issues. It uses interconnected symbols to represent the movement of information and processing. "Flowcharting" is the process of creating a flowchart for an algorithm.

Figure 4.3 shows the project application's flowchart.

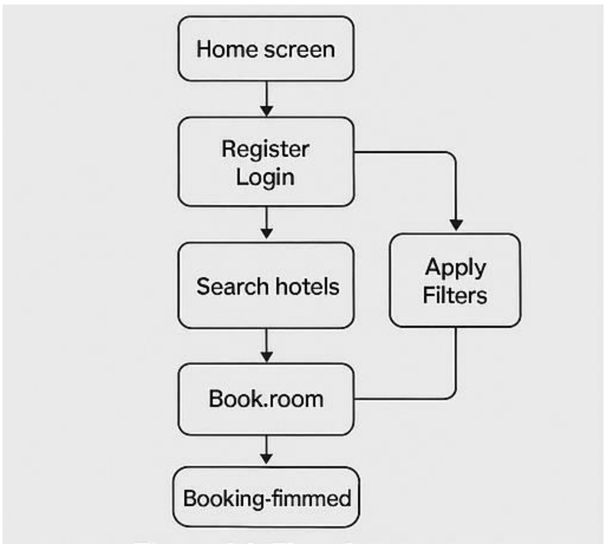


Figure 4.3 Flowchart

1. **Home Screen → Register / Login**

   - Users begin at the Home screen and must either **sign in** or **register** before accessing booking features.
   - This ensures secure access to personalized data like booking history and saved preferences.

2. **Search Hotels**

   - After authentication, users land on a **search interface** where they enter destination, check-in/check-out dates, and guest count.
   - The app triggers a search request to fetch hotel options dynamically.

3. **Apply Filters**

   - To refine results, users can apply filters such as price range, star ratings, or amenities which update the hotel list accordingly.

- This improves discoverability and user control over choices.

4. **Select & Book Room**

- Users choose a specific hotel room from the results, review details and pricing, then proceed to the **booking flow** (choose room options, enter guest info, confirm dates).

5. **Booking Confirmed**

- Once payment is processed, users receive a **confirmation screen** indicating successful booking.
- This completes the happy path with final confirmation and booking details.

4.4 Use case Diagram

The scope and high-level functions of a system are described in use-case diagrams. The interactions between the system and its actors are also depicted in these diagrams. Utilize-case diagrams show what the system does and how the actors use it, but they do not show how the system works within.

Figure 4.4 shows the use case diagram of the project.

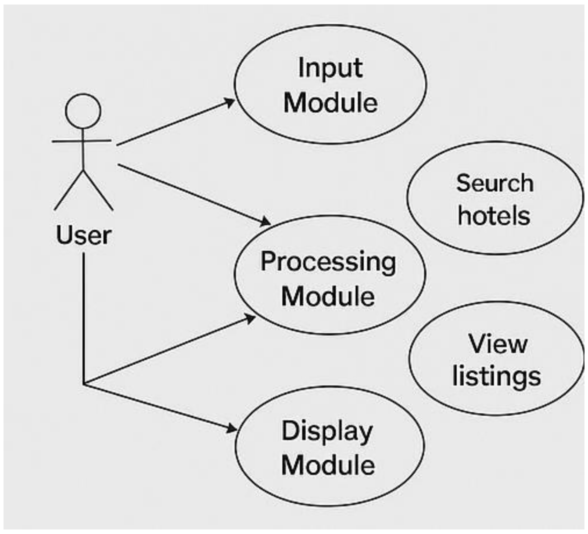


Figure 4.4 Use Case Diagram

1. **Input Module:** User will be able to register, login and add pins to the map by passing information about the place.
2. **Processing Module:** This module processes the data given by user, performs the required operations and passes it on to the storage module or display module.
3. **Display Module:** User will be able to see the outputs of the operations through this module in the form of text or map graphics.

Chapter-5

IMPLEMENTATION

The technical data package (TDP) for the software is converted into one or more fabricated, integrated, and tested software configuration items during the software implementation stage so that they are prepared for software acceptance testing.

5.1 Algorithm

An algorithm is a process used to carry out a computation or solve a problem. In either hardware-based or software-based routines, algorithms function as a detailed sequence of instructions that carry out predetermined operations sequentially. All aspects of information technology employ algorithms extensively.

```
login(username, password)
```

```
allow_search_access(true)
else return to login screen
```

```
show_search_bar(), display_featured_hotels()
```

```
fetch_filtered_hotels(city, checkin_date, price_range, ratin
```

```
show_hotel_details(hotel_id)
```

```
book_room(user_id, hotel_id, date_range)
```

5.2 Important Code Snippets (Flutter + Firebase/REST)

5.2.1 Firebase Authentication

```
final FirebaseAuth _auth = FirebaseAuth.instance;


Future<void> login(String email, String password) async {
  try {
    await _auth.signInWithEmailAndPassword(email: email, password:
password);
  } catch (e) {
    print("Login Failed: $e");
  }
}
```

5.2.2 Search Field and Filter in Flutter

```
TextField(
  controller: _searchController,
  onChanged: (value) => filterHotels(value),
  decoration: InputDecoration(
    hintText: 'Search by location or hotel name',
  ),
)
```

5.2.3 Hotel Card UI Widget

```
  ListView.builder(
  itemCount: hotels.length,
  itemBuilder: (context, index) {
    return Card(
      child: ListTile(
        title: Text(hotels[index].name),
        subtitle: Text('Rating: \${hotels[index].rating}'),
        onTap: () => showHotelDetails(hotels[index]),
      ),
    );
  },)
```

5.2.4 Booking a Room via HTTP API

```
  Future<void> bookRoom(hotelId, userId, dates) async {
  final response = await http.post(
    Uri.parse("https://yourapi.com/bookings"),
    body: jsonEncode({
      'hotelId': hotelId,
      'userId': userId,
      'dates': dates,
    }),
    headers: {'Content-Type': 'application/json'},
  );
  if (response.statusCode == 200) {
    print("Booking Confirmed");
  }
}
```

5.2.5 Firestore Data Structure (Example Entry)

```
{
  "_id": 1,
  "name": "Taj Hotel",
  "location": "Mumbai",
  "price": 3500,
  "rating": 4.8,
  "available": true,
  "coordinates": {
    "lat": 19.0760,
    "lng": 72.8777
  }
}
```

5.2.6 Flutter Hotel Detail Page

```
void showHotelDetails(Hotel hotel) {
  Navigator.push(context, MaterialPageRoute(builder: (context) =>
HotelDetailScreen(hotel: hotel)));
}
```

5.2.7 Booking Confirmation Dialog

```
   showDialog(
   context: context,
   builder: (context) => AlertDialog(
    title: Text('Booking Confirmed'),
    content: Text('Thank you for booking with us!'),
    actions: [
      TextButton(
        onPressed: () => Navigator.pop(context),
        child: Text('OK'),
      )
    ],
  ),
);
```

5.2.8 User Password Hash Validation

```
      const validPassword = await bcrypt.compare(req.body.password, user.password);
       if (!validPassword) return res.status(400).json("Invalid credentials");
```

Chapter-6

TESTING

The Software testing is a technique for determining whether the actual software product complies with expectations and is error-free. It involves executing software/system components either manually or automatically to evaluate key properties. The goal of testing is to detect defects, gaps, or unmet requirements.

Testing ensures early detection of issues before the product release. A well-tested Flutter app also boosts user satisfaction and reliability.

Table 6.1: Test Cases

Test Case No	Test Case Description	Expected Input	Expected Output	Pass/Fail
1	User Register	Wrong Email format	Fail	Pass
2	User Register	Correct Email format	Pass	Pass
3	User Login	Wrong Credentials	Fail	Pass
4	User Login	Correct Credentials	Pass	Pass
5	Search Hotel	Invalid location input	Fail	Pass
6	Search Hotel	Valid filters	Pass - Filtered Hotels List	Pass
7	Hotel Booking	Invalid Booking Data	Fail - Booking Rejected	Pass
8	Hotel Booking	Valid Booking Data	Pass - Booking Confirmed	Pass
9	View Hotel Details	Nonexistent Hotel ID	Fail - Not Found	Pass
10	View Hotel Details	Valid Hotel ID	Pass - Hotel Info Loaded	Pass

Unit testing is performed on the smallest testable components of the Flutter app, such as widgets, functions, or services. These are tested independently to verify correct behaviour.

Chapter-6

RESULTS AND SNAPHOTS

Snapshot 1: The following figure 6.1 shows the Hotel Searching Page

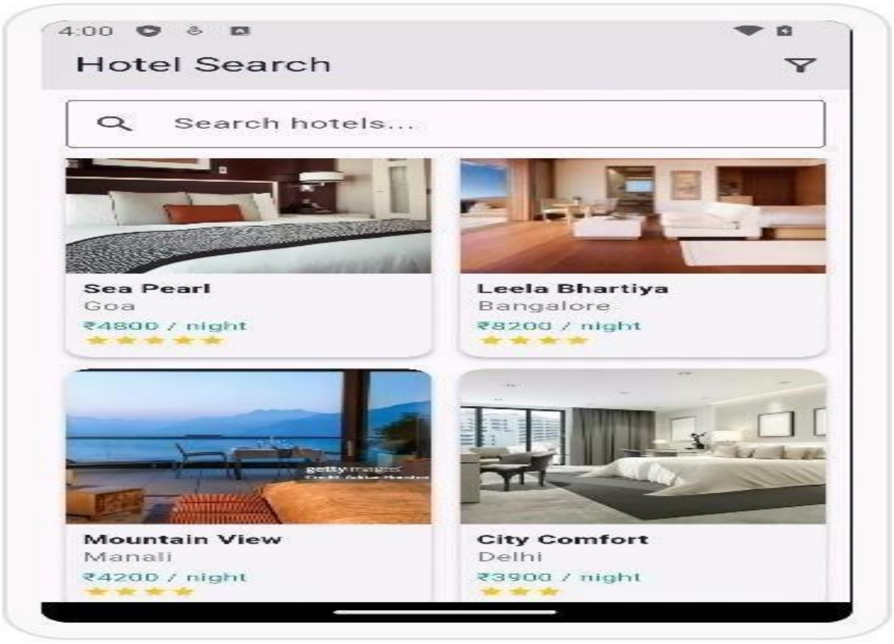


Figure 6.1 Search Page

Snapshot 2: The following figure 6.2 shows the searching hotels based on filters

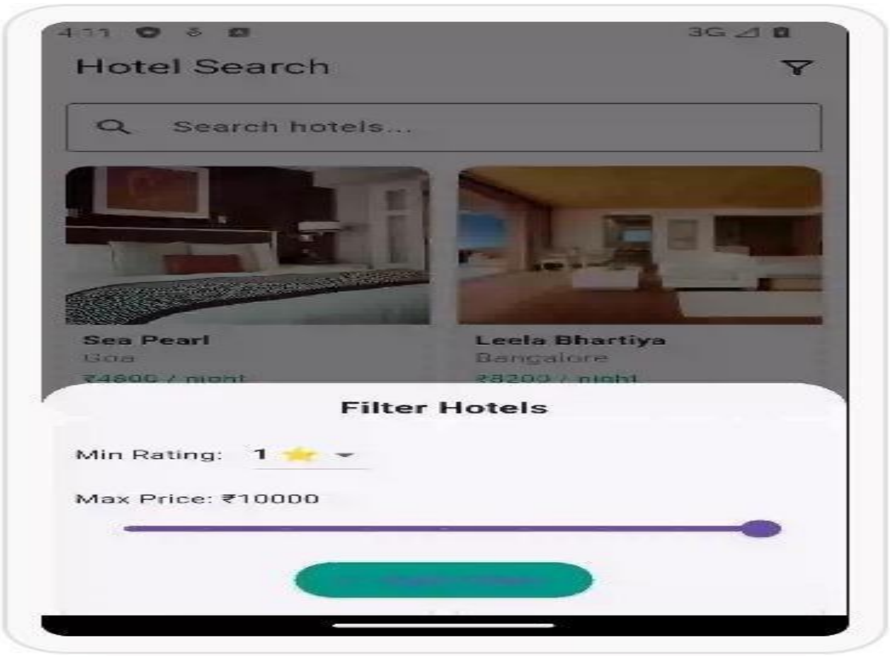
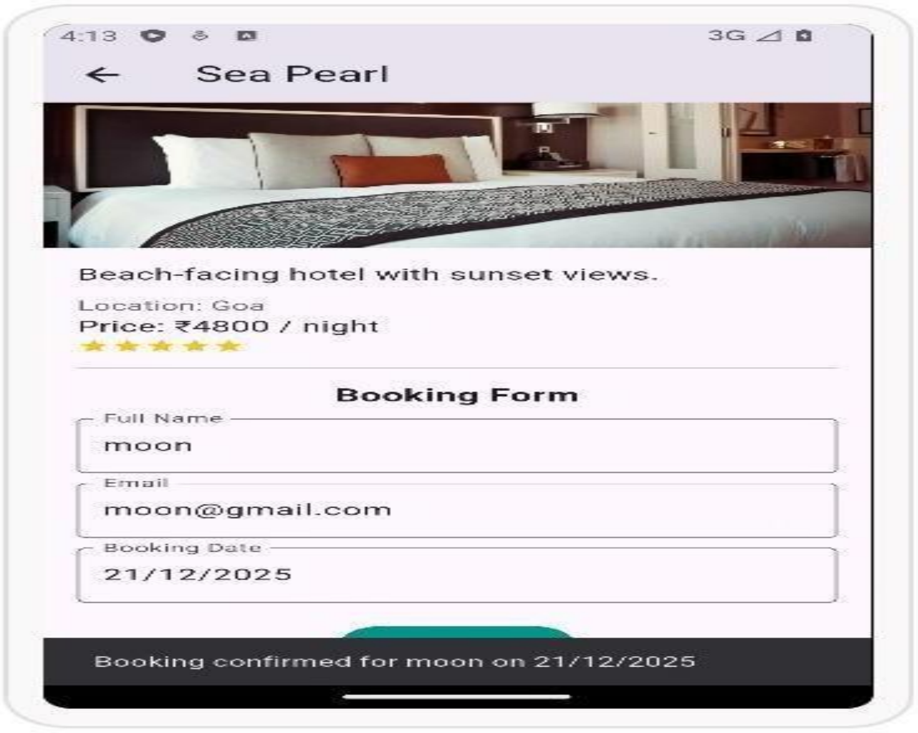


Figure 6.2 filter Hotels

Snapshot 3: The following figure 6.3 shows Hotels Booking Form



Beach-facing hotel with sunset views.
Location: Goa
Price: ₹4800 / night
★★★★★

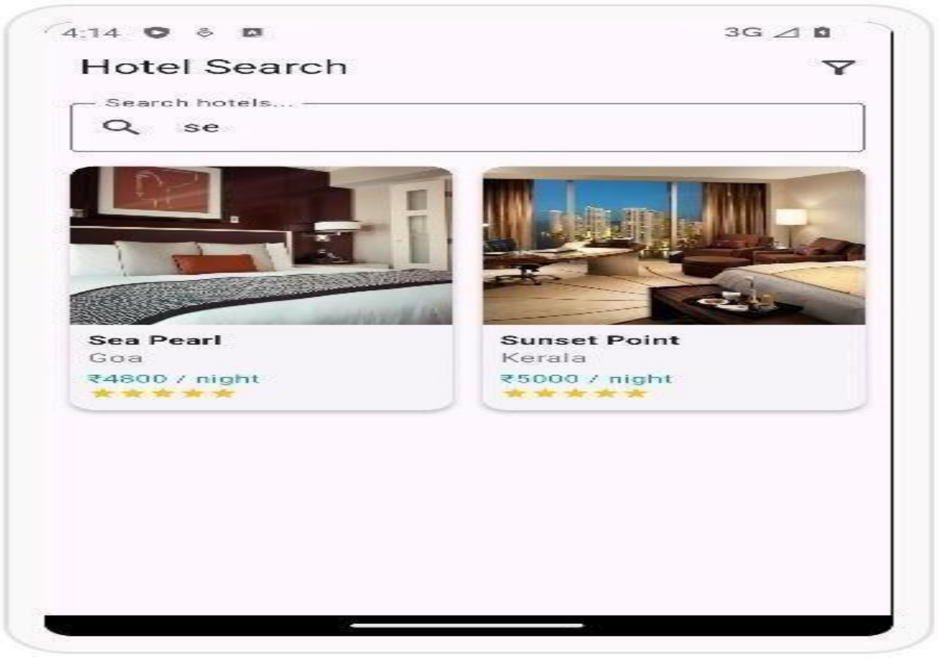
Booking Form

Full Name	moon
Email	moon@gmail.com
Booking Date	21/12/2025

Booking confirmed for moon on 21/12/2025

Figure 6.3 4 Booking Form

Snapshot 4: The following figure 6.4 shows Hotel searching based on ratings



Search hotels... se

Sea Pearl
Goa
₹4800 / night
★★★★★

Sunset Point
Kerala
₹5000 / night
★★★★★

Figure 6.4 Hotel Searching

Chapter-7

CONCLUSION

This hotel room searching app demonstrates how Flutter can be leveraged to build a responsive and user-friendly mobile application for discovering and booking hotel rooms. By integrating real-time search, filtering options based on rating and price, and detailed hotel views, the app offers an intuitive experience that helps users quickly find accommodations that match their preferences. The modular design using Flutter widgets like GridView, ModalBottomSheet, and Forms highlights Flutter’s flexibility in building rich UI components. Moreover, the booking form functionality showcases basic form validation and user interaction within the app.

Future improvements could include adding advanced features such as date pickers for selecting check-in/check-out dates, integration with backend services for live hotel availability, user authentication, and payment gateways to support end-to-end booking workflows. Additionally, incorporating personalized recommendations and user reviews could further enhance the user experience.

Overall, this app serves as a strong foundation for scalable hotel booking solutions using Flutter’s cross-platform capabilities.

REFERENCES

1. **Google Developers.** (n.d.). *Flutter Documentation.* Retrieved from: https://docs.flutter.dev
2. **Flutter Packages.** (n.d.). *pub.dev - Flutter Package Repository.* Retrieved from: https://pub.dev
3. **Udemy.** (2024). *The Complete Flutter Development Bootcamp Using Dart.* [Online Course] Retrieved from: https://www.udemy.com/course/flutter-bootcamp-with-dart/
4. **Raywenderlich.com.** (2023). *Flutter for Beginners: Learn App Development with Dart.* Retrieved from: https://www.raywenderlich.com/flutter
5. **GeeksforGeeks.** (2023). *Flutter - Build a Hotel Booking App UI.* Retrieved from: https://www.geeksforgeeks.org/flutter-build-a-hotel-booking-app-ui/