**Software Design & Architecture**

**Project Name: TravelEase**

**Team Members:**

| **1** | Muhammad Amir | BSE-F21-M013 |
| --- | --- | --- |
| **2** | Faiqa Nasir | BSE-F21-M016 |
| **3** | Huzaifa Riaz | BSE-F21-M026 |

**Project Topic:** TravelEase

**Description:** TravelEase aims to revolutionize the way people plan and manage their trips, offering streamlined and personalized travel solutions for both individuals and businesses. Designed to be the ultimate travel companion, TravelEase will provide a centralized platform to book flights, hotels, and activities. It will simplify itinerary creation, and deliver real-time travel updates - all aiming to enhance the overall travel experience.

**Scope:** TravelEase is an online travel management platform targeting a wide audience, including leisure travelers or business professionals. The platform offers a comprehensive suite of services covering the entire travel journey, from initial planning to booking, management, and post-trip support. It focuses on user-friendliness, personalization, and accessibility across various devices.

**Features:**

1. User Registration and Login
2. User Profile Management
3. Flight Search and Booking
4. Hotel Search and Booking
5. Activity/Tour Booking
6. Customizable Itinerary Builder
7. Travel Alerts and Updates
8. Expense Tracking
9. Destination Guides
10. In-App Messaging
11. CRUD Operations
12. Travel Reviews and Ratings
13. Travel Agency Portal
14. Multi-Language Support
15. Secure Payment Gateway

**Note:** “System” is not an external entity. However, the purpose of writing the system in the Actor column is meant to describe the **response of the system** to the action taken by an external entity.

| **Sr#** | **Actor** | **Requirement Details** |
| --- | --- | --- |
| 1 | Traveler | Users (travelers/agencies) "shall" register to access personalized features. |
| 1.1 | Traveler | Travelers "shall" register by providing essential details (name, email, password, etc.). |
| 1.2 | Travel Agency | Travel agencies "shall" register with business details and contact information. |
| 2 | Traveler/Agency | Users (travelers/agencies) "shall" log in with their credentials to access the platform. |
| 3 | Traveler | Travelers "shall" be able to search for flights with flexible criteria. |
| 3.1 | System | System "shall" present flight options with pricing, schedules, and details. |
| 3.2 | System | System "shall" support flight filtering (stops, airlines, timings, etc.). |
| 4 | Traveler | Travelers "shall" be able to search for hotels with flexible criteria. |
| 4.1 | System | System "shall" present hotel options with details, availability, and pricing. |
| 5 | Traveler | Travelers "shall" be able to search and book activities/experiences. |
| 5.1 | System | System "shall" display available activities with descriptions, dates, and pricing. |
| 5.2 | System | System "shall" enable filtering activities by type, location, and interests. |
| 6 | Traveler | Travelers "shall" be able to create itineraries by adding flights, hotels, and activities. |
| 6.1 | Admin | System "shall" provide intuitive tools for building and modifying itineraries. |
| 6.2 | System | System "shall" calculate and display estimated trip costs in real-time. |
| 7 | Traveler | Travelers "shall" receive real-time alerts for flight status changes/delays. |
| 7.1 | System | System "shall" automatically track flight information. |
| 7.2 | Traveler | Travelers "shall" be able to customize their preferences for receiving notifications. |
| 8 | Traveler | Travelers "shall" be able to track and categorize trip expenses. |
| 8.1 | System | System "shall" allow for manual expense entry and optional linking to banking accounts. |
| 8.2 | System | System "shall" generate spending reports with customizable filters. |
| 9 | Traveler | Travelers "shall" be able to access destination guides for inspiration and planning. |
| 9.1 | System | System "shall" curate destination guides with information on attractions, restaurants, etc. |
| 9.2 | System | System "shall" incorporate photos, videos, and maps within destination guides. |
| 10 | Traveler | Travelers "shall" be able to communicate with support or travel agents via in-app messaging. |
| 10.1 | System | System "shall" provide a secure messaging platform within the user interface |
| 11 | Admin | Admin “shall” be able to perform CRUD operations on entities interacting with the system. |
| 12 | Traveler | Travelers "shall" be able to submit and view ratings and reviews for flights, hotels, and activities. |
| 12.1 | System | System "shall" provide forms for submitting reviews and ratings. |
| 12.2 | System | System "shall" display aggregated reviews on relevant booking pages. |
| 13 | Travel Agency | Travel agencies "shall" have a dedicated portal to manage client bookings. |
| 13.1 | System | System "shall" provide secure authentication for agency accounts. |
| 13.2 | System | System "shall" enable agencies to search, book, and manage flights, hotels, and activities on behalf of clients. |
| 14 | Traveler/Agency | Users (travelers/agencies) "shall" be able to access multi-language support in the platform. |
| 15 | Traveler/Agency | Users (travelers/agencies) "shall" be able to make secure payments through a variety of payment gateways. |

**Use Case:**

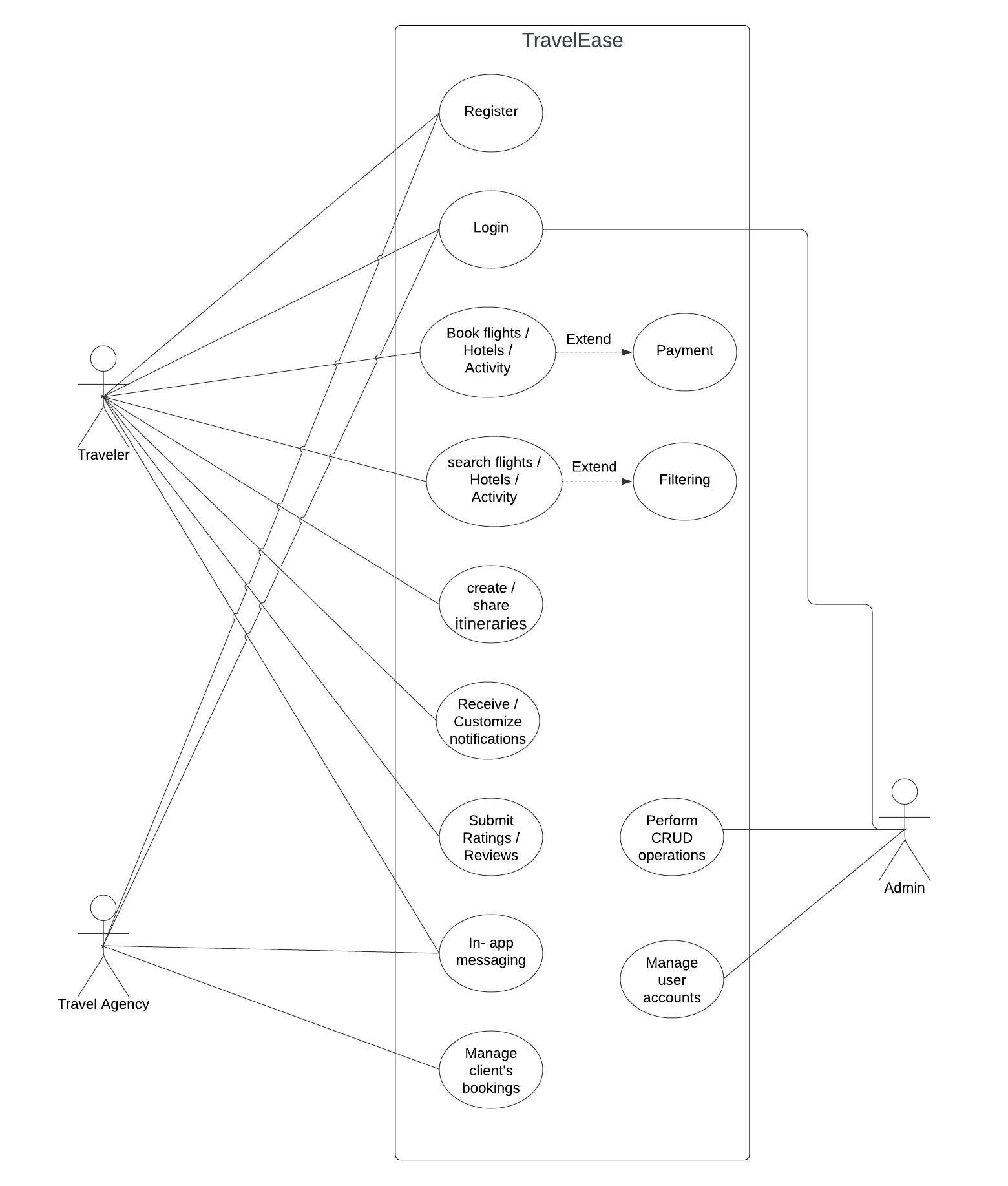
**Primary Actors:**

* **Traveler:** Represents individuals who utilize the travel management platform for planning, booking, and managing their travel experiences.
* **Travel Agency:**Represents organizations or agents who offer travel services and utilize the platform to facilitate bookings and interactions with users.

**Secondary Actor:**

* **Admin:**Represents the system administrator responsible for overseeing and managing the platform's operations, user accounts, and system settings.

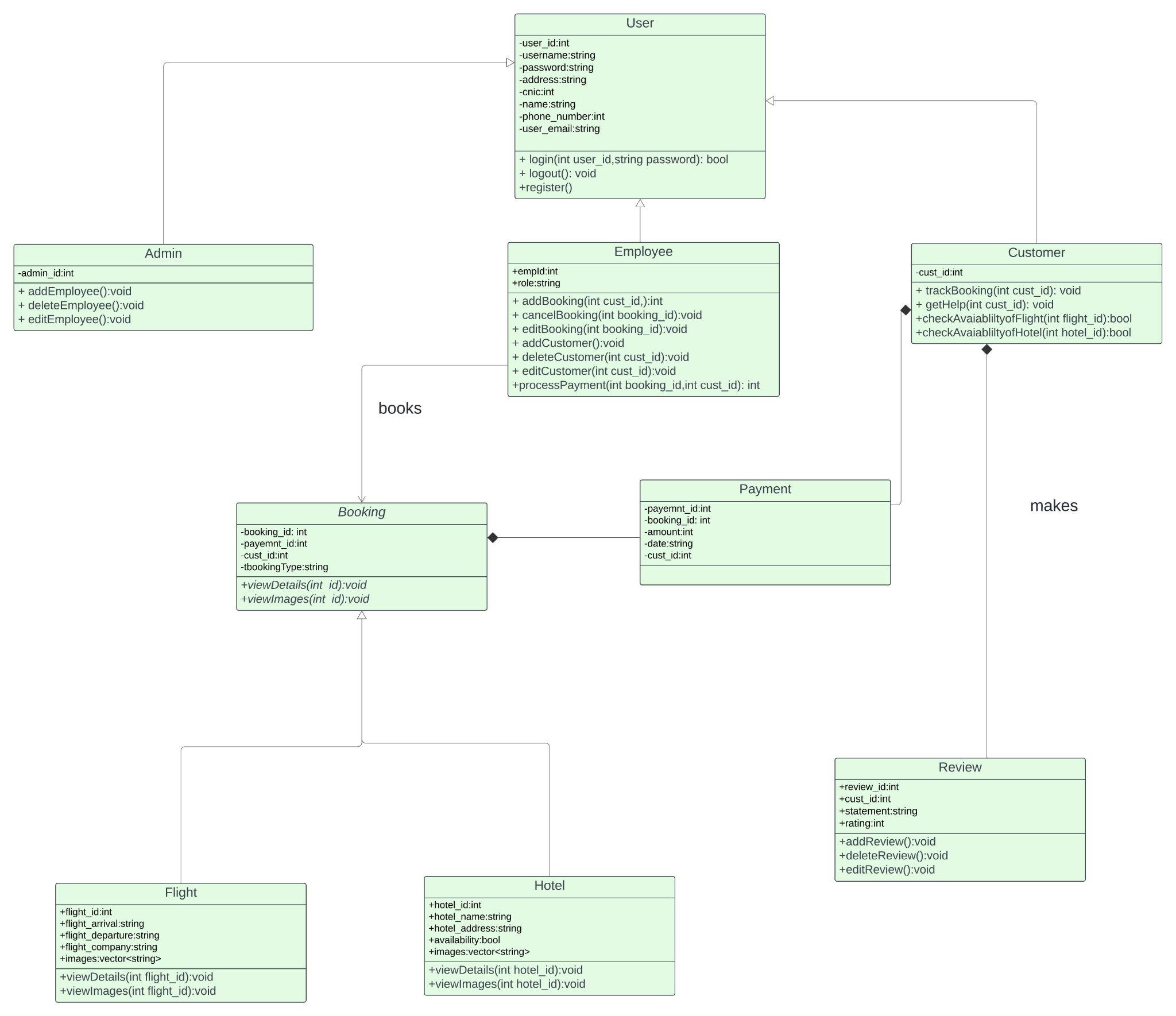
**Use Case Diagram**

****

**Description:**

The platform enables users to register, log in, and manage profiles. Users search, book flights, hotels, activities, create itineraries, share trips, track expenses, access guides, send messages, and make secure payments. Travel agencies manage bookings, access information, and use the agency portal for operations. Admin ensures security, configures settings, manages accounts, handles disputes, monitors activity, and performs administrative tasks. Overall, it offers efficient travel planning, booking, and management for users and agencies with administrative oversight.

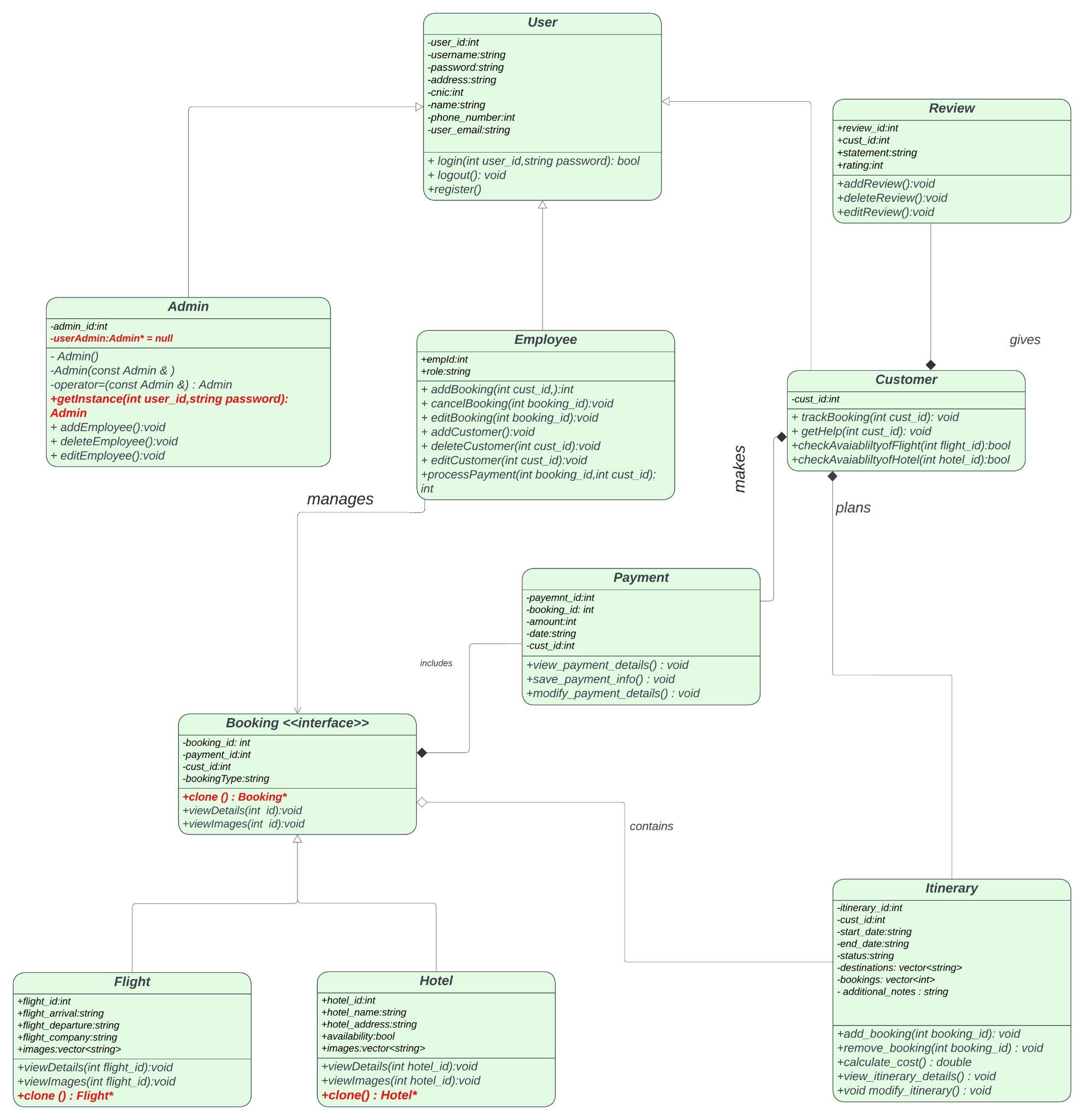
**Class Diagram**



**Refined Class Diagram**

**Changes:**

* Used singleton design pattern in Admin class.
* Used prototype design pattern in Booking & Descendant classes.



**Class Diagram Description**

**Object-Oriented Design (OOD) Integration with Integrated Database**

· The implementation of our system was based upon **OOD**, with an integrated database featuring a table corresponding to each object class.

**Singleton Pattern Implementation for Admin Class**

· The **Singleton** pattern was adopted specifically for the **Admin** class to restrict instantiation to a single instance at any given time, preventing conflicting instances that could lead to system discrepancies.

· To reduce redundancy in user roles and ensure system consistency, a User class served as the base class for Admin , Employee and customer classes.

· As we were confident that no new category of users would be added in the future, we opted not to use the abstract factory design pattern, thereby adhering to the **Open/Closed** Principle among the **SOLID** principles of OOD.

· This approach facilitates streamlined user management and access privileges.

**User Management and Access Privileges**

· Users are assigned access privileges based on their IDs, with their roles determined by an existing database table mapping user IDs to roles such as employee, admin, or customer.

· Notably, users are limited to registering solely as customers. Registration of employees is only possible by an admin.

**Functionality for Customers**

· Upon registration, customers gain the ability to request assistance, enabling CRUD operations on their bookings through employees.

· Validation of credentials like customer ID and booking ID is required for processing bookings.

· Bookings are processed only upon receipt of payment. After adding a booking, customers receive a booking ID and proceed to make the payment.

· If the payment is successfully validated, the booking is confirmed, and the customer is promptly notified.

**Booking Module with Flight and Hotel Subclasses**

· The **Booking** class is an **abstract** class that serves as the base for all types of bookings.

· It contains a pure virtual function called viewDetails(), which must be implemented by its derived classes.

· This abstraction allows for the creation of different types of bookings while ensuring a common interface for viewing booking details.

· Flight and Hotel are subclasses of the Booking class, representing specific types of bookings for flights and hotels, respectively.

· The **Prototype** design pattern is employed within the Booking module to facilitate the cloning of previous bookings.This allows for efficient handling of repeated travelers, as their previous bookings can be cloned and added to their record without the need for redundant data entry.By cloning existing booking objects, the system ensures consistency and accuracy while minimizing duplication effort.

In conclusion, the implementation follows a modular approach, ensuring that each class and component has a well-defined responsibility.The integration of OOD principles with database functionality enhances system flexibility, maintainability, and scalability.The use of design patterns like Singleton and Prototype ensures efficient resource utilization and system integrity.