1.prepare a SRS document in line with the IEEE recommended standards.

Software Requirements Specification (SRS)

Off-Road Adventure

Document Version: 1.0 Date: April 26, 2025

1. Introduction

1.1 Purpose

The purpose of this document is to specify the functional, non-functional, and technical requirements for the development of the "Off-Road Adventure" platform. The website will serve as a hub for off-road travel enthusiasts, providing tools for discovering destinations, planning trips, and booking related services.

1.2 Scope

"Off-Road Adventure" is a web-based platform designed to offer:

- Exploration of off-road destinations globally.
- Travel guides, including safety tips, recommended gear, and itineraries.
- Secure booking of 4x4 vehicles, accommodations, and guided tours.
- A community forum for travelers to share experiences and advice. The platform will ensure an intuitive user experience, high security, and scalability to support a growing user base.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- UI: User Interface
- UX: User Experience
- API: Application Programming Interface

1.4 References

1. IEEE Std 830-1998, *Recommended Practice for Software Requirements Specifications*, IEEE, 1998.
2. Google Maps API Documentation. Retrieved from https://developers.google.com/maps/documentation.
3. Stripe API Documentation. Retrieved from https://stripe.com/docs/api.
4. Overland Journal, *Off-Road Travel Guidelines*. Retrieved from https://www.overlandjournal.com.
5. WCAG 2.1, *Web Content Accessibility Guidelines*, W3C Recommendation. Retrieved from https://www.w3.org/TR/WCAG21/.

2. Overall Description

2.1 Product Perspective

The "Off-Road Adventure" platform is a new web-based system aimed at creating a comprehensive travel solution for off-road enthusiasts. It integrates third-party APIs for booking and mapping, ensuring seamless trip planning and execution.

2.2 Product Functions

1. Destination Explorer: Search and filter destinations based on terrain type, location, and difficulty level.
2. Travel Guides: Articles on off-road driving techniques, survival tips, and recommended gear.
3. Booking System: Secure booking for vehicles, guided tours, and accommodations.
4. Community Forum: Users can connect, share experiences, and post travel itineraries.

2.3 User Characteristics

- Visitors: Unregistered users who explore public content.
- Registered Users: Enthusiasts who book services and contribute to forums.
- Administrators: Manage website content, user accounts, and booking systems.

2.4 Constraints

- The platform must support at least 5,000 concurrent users.
- Development time is limited to 6 months with a budget of $75,000.

2.5 Assumptions and Dependencies

- Reliable internet access for all users.
- Dependency on third-party APIs for mapping and booking functionalities.

3. Specific Requirements

3.1 Functional Requirements

1. Destination Search:
    - Users can explore off-road destinations and apply filters like terrain, difficulty, and region.
2. Travel Guide Section:
    - Users can access articles, maps, and safety tips related to off-road travel.
3. Booking Feature:
    - Integrated system for secure booking of vehicles, tours, and accommodations.
4. User Accounts:
    - Registered users can create profiles, save itineraries, and contribute to forums.
5. Community Forum:
    - An interactive space for users to share tips, experiences, and stories.

3.2 Non-Functional Requirements

1. Performance:
    - Average page load time under 2.5 seconds.
    - Handle 5,000+ concurrent users.
2. Security:
    - SSL encryption for all user data and transactions.
    - Two-factor authentication for user accounts.

3. Usability:
    - Responsive design for both mobile and desktop devices.
    - Accessible navigation with user-friendly filters and search options.
4. Accessibility:
    - Compliant with WCAG 2.1 to cater to a wide range of users.

4. External Interface Requirements

4.1 User Interfaces

- Web-based UI with dynamic, map-based exploration tools.
- Mobile-optimized UI for seamless access on-the-go.

4.2 Hardware Interfaces

- Cloud-based server infrastructure to ensure high availability and scalability.

4.3 Software Interfaces

- Integration with Google Maps API for route planning.
- Integration with Stripe/PayPal for secure payments.

5. System Features

1. Destination Explorer:
    - Features an interactive, map-based interface for finding off-road trails.
2. Travel Gear Store:
    - Sell off-road travel gear via an integrated e-commerce system.
3. Guided Tours:
    - Real-time booking for guided tours and training programs.

6. Other Non-Functional Requirements

6.1 Reliability

- 99.9% system uptime to ensure consistent user access.

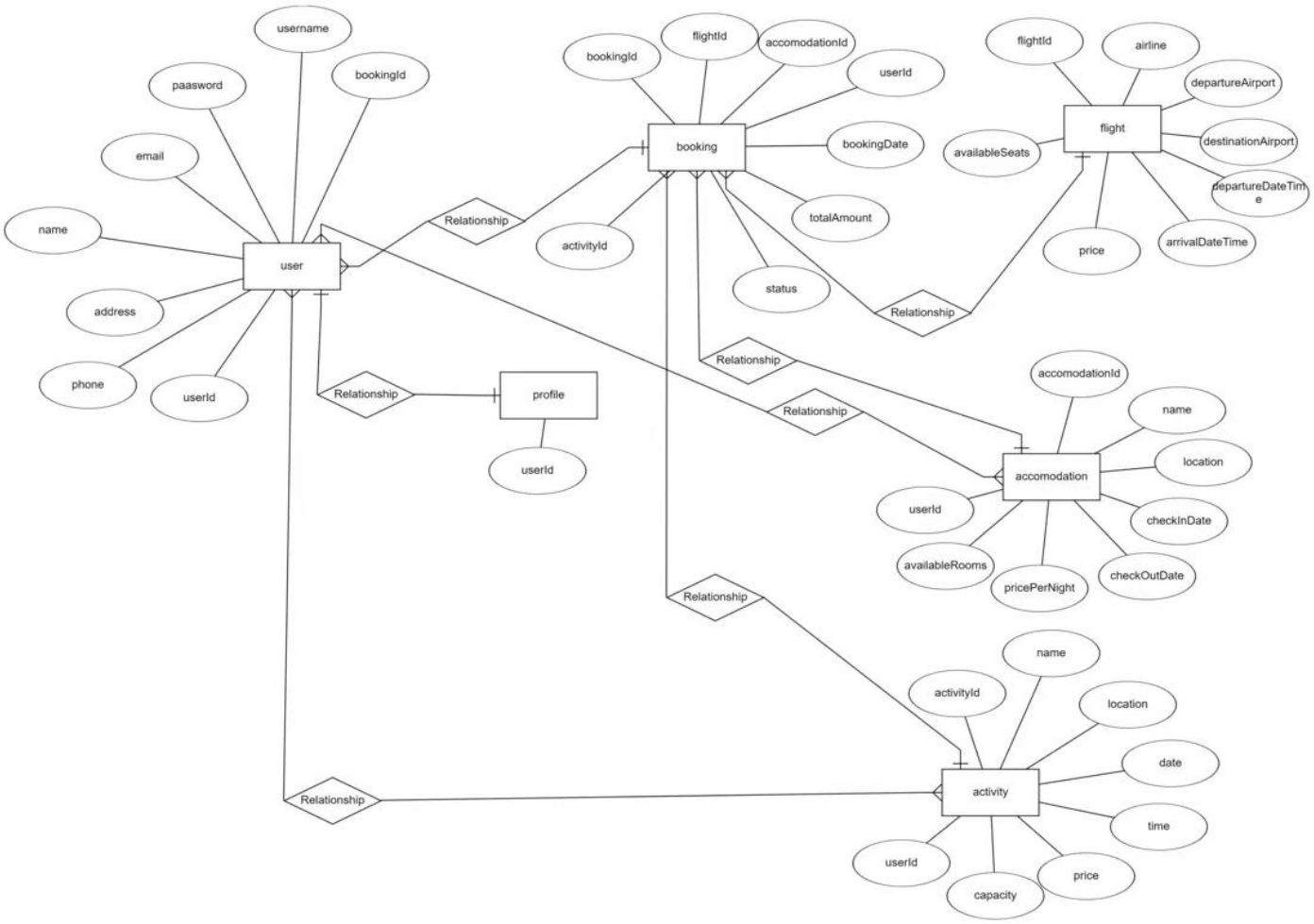
6.2 Maintainability

- Modular architecture for easy updates and scalability.

6.3 Portability

- Compatible across all major browsers and devices.

2.Draw the entity relationship diagram of a project.



Entities and Attributes

1. User: Represents individuals who interact with the system

- **UserID (Primary Key):** Unique identifier for each user.
- **Username:** Name chosen by the user for login.
- **Password:** Securely stored password for login authentication.
- **Email:** Email address associated with the user's account.
- **Name:** Full name of the user.
- **Address:** Physical address of the user.
- **Phone:** Contact the phone number of the user.

2. Booking: Records details of each reservation made by users

- **BookingID (Primary Key):** Unique identifier for each booking.
- **UserID (Foreign Key):** References the user who made the booking.
- **BookingDate:** Date when the booking was made.
- **TotalAmount:** Total amount payable for the booking.
- **Status:** Status of the booking (e.g., pending, confirmed, cancelled).

3. Flight: Stores information about available flights

- **FlightID (Primary Key):** Unique identifier for each flight.
- **Airline:** Name of the airline operating the flight.
- **DepartureAirport:** Departure airport for the flight.
- **DestinationAirport:** Destination airport for the flight.
- **DepartureDateTime:** Date and time of departure.
- **ArrivalDateTime:** Date and time of arrival.
- **Price:** Price of the flight ticket.
- **AvailableSeats**: Number of available seats on the flight.

4. Accomodation: Represents available lodging options

- **AccommodationID (Primary Key):** Unique identifier for each accommodation.
- **Name:** Name or title of the accommodation.
- **Location:** Location or address of the accommodation.
- **CheckInDate:** Date for check-in.
- **CheckOutDate:** Date for check-out.
- **PricePerNight:** Price per night for the accommodation.
- **AvailableRooms:** Number of available rooms in the accommodation.

5. Activity: Manages information about activities or tours available

- **ActivityID (Primary Key):** Unique identifier for each activity.
- **Name:** Name or title of the activity.
- **Location:** Location or address of the activity.
- **Date:** Date of the activity.
- **Time:** Time of the activity.
- **Price:** Price of the activity.
- **Capacity:** Maximum capacity or number of participants for the activity.

Relationship Between These Entities

1. User - Accommodation Relationship (Many-to-Many):

- Users can book multiple hotels, indicating a user can make bookings for different accommodations.
- Every accommodation can be booked by multiple users, meaning a hotel can have bookings from different users.

2. User - Booking Relationship (Many-to-One):

- Many bookings can be associated with one user, showing that a user can make multiple bookings over time.

3. User - Activity Relationship (Many-to-Many):

- Users can book multiple activities, allowing a user to participate in various activities.
- Every activity can be booked by multiple users, indicating that an activity can have participants from different users.

4. Booking - Activity Relationship (Many-to-One):

- Many activities can be associated with one booking, meaning that a booking can include multiple activities.

5. Booking - Accommodation Relationship (Many-to-One):

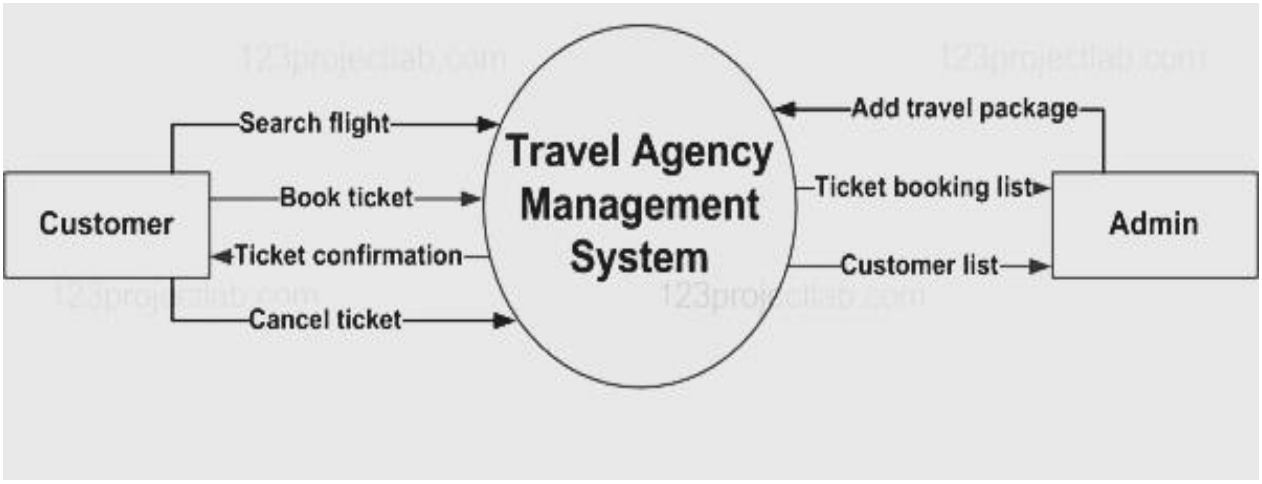
- Many hotels can be associated with one booking, showing that a booking can include stays at multiple hotels.

6. Booking - Flight Relationship (Many-to-One):

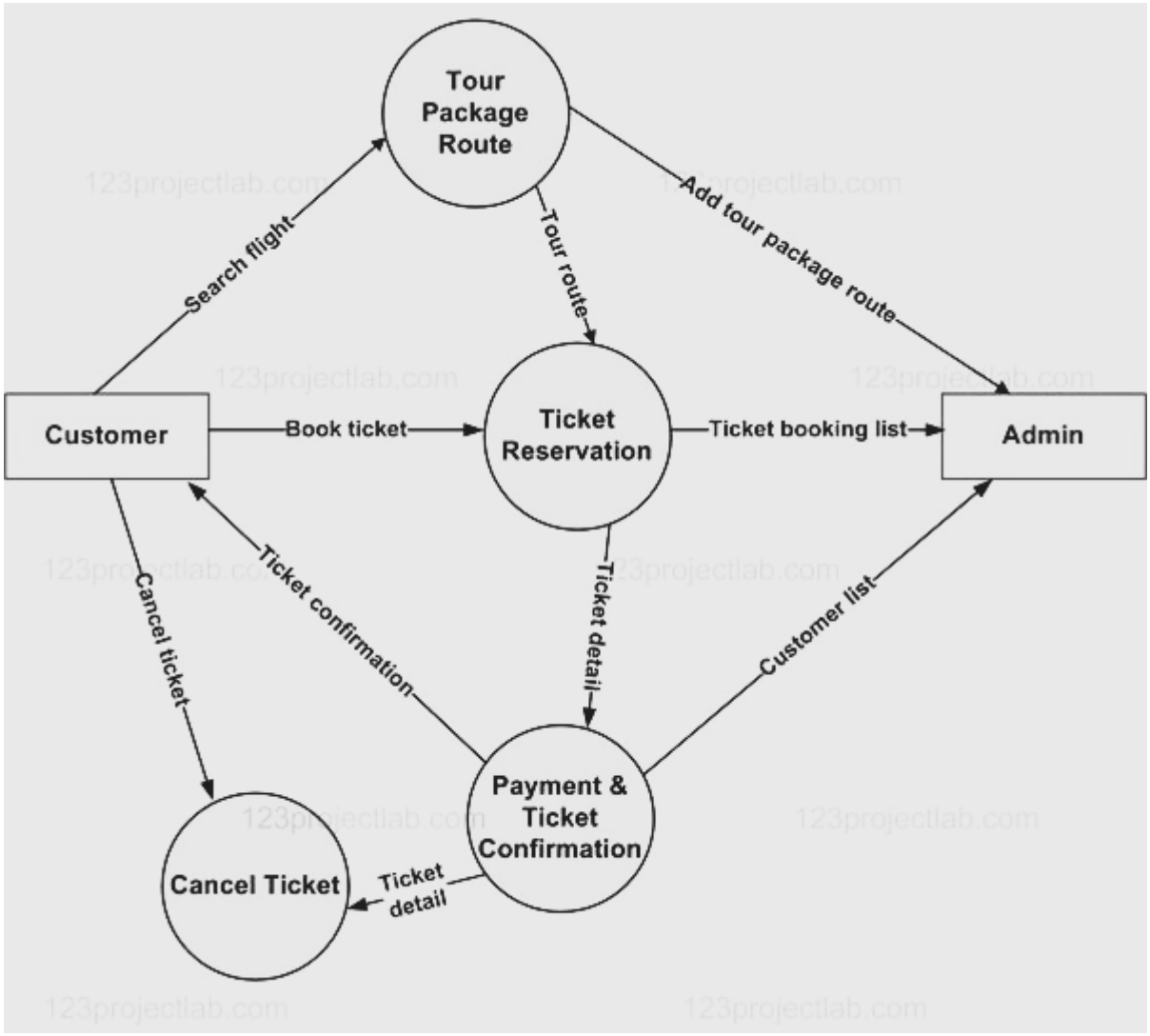
- Many bookings can be associated with one flight, indicating that a booking can include a flight reservation.

3.Draw the data flow diagram at level 0 and level 1.

Level 0



Level 1

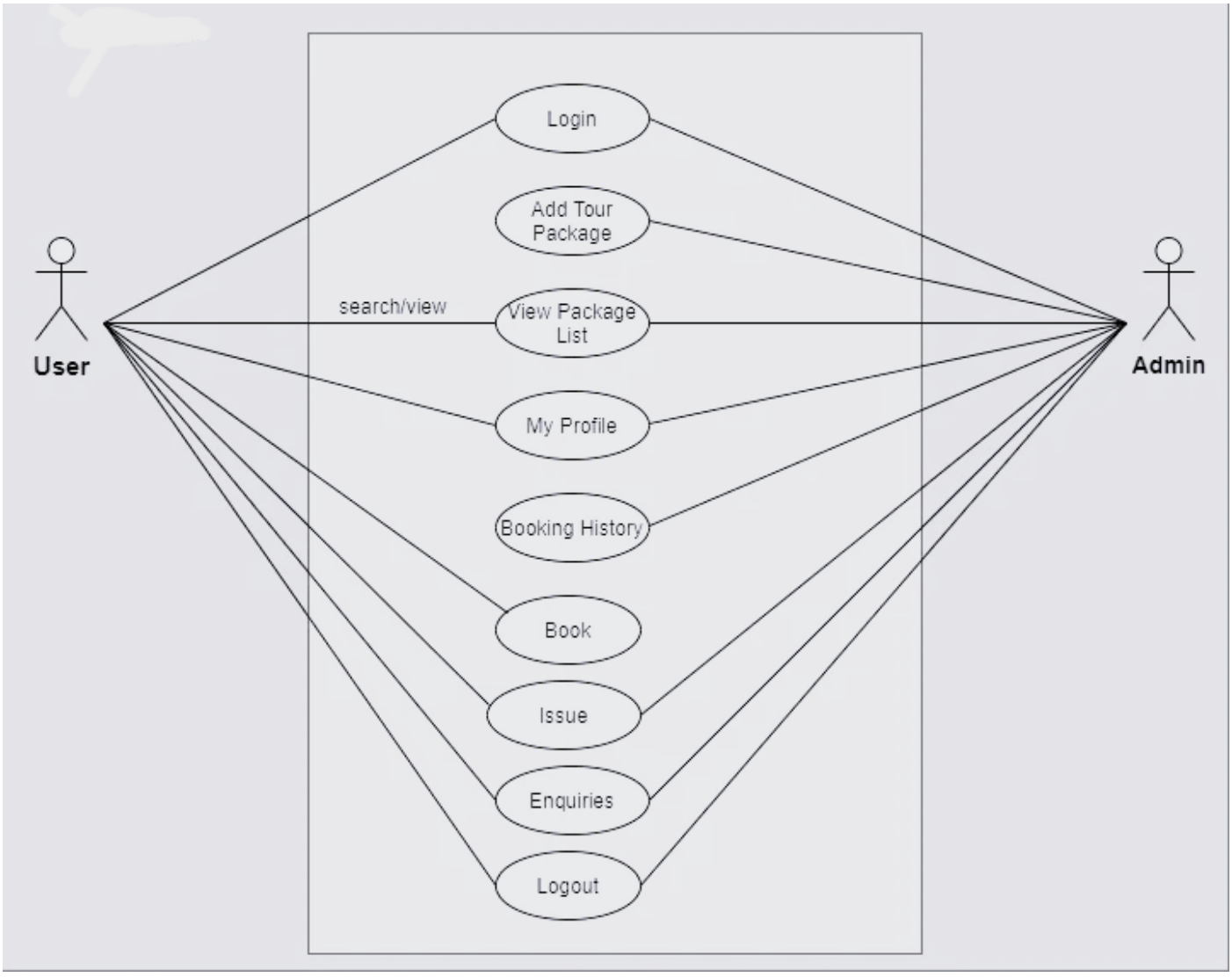


A DFD maps out the flow of information for any process or system. It gives a basic overview of the whole system or process being analyzed. It shows the system with its relationship to external entities.

4.Draw use case diagram in argo UML.

A UML use case diagram can create a broad, high-level view of the relationship between use cases, actors involved, and systems being performed.

use cases are represented by oval shapes, and the lines then show at which point an actor/user participates and interacts with their corresponding use case. You can see where each actor is involved within the entire process (and where they're excluded).

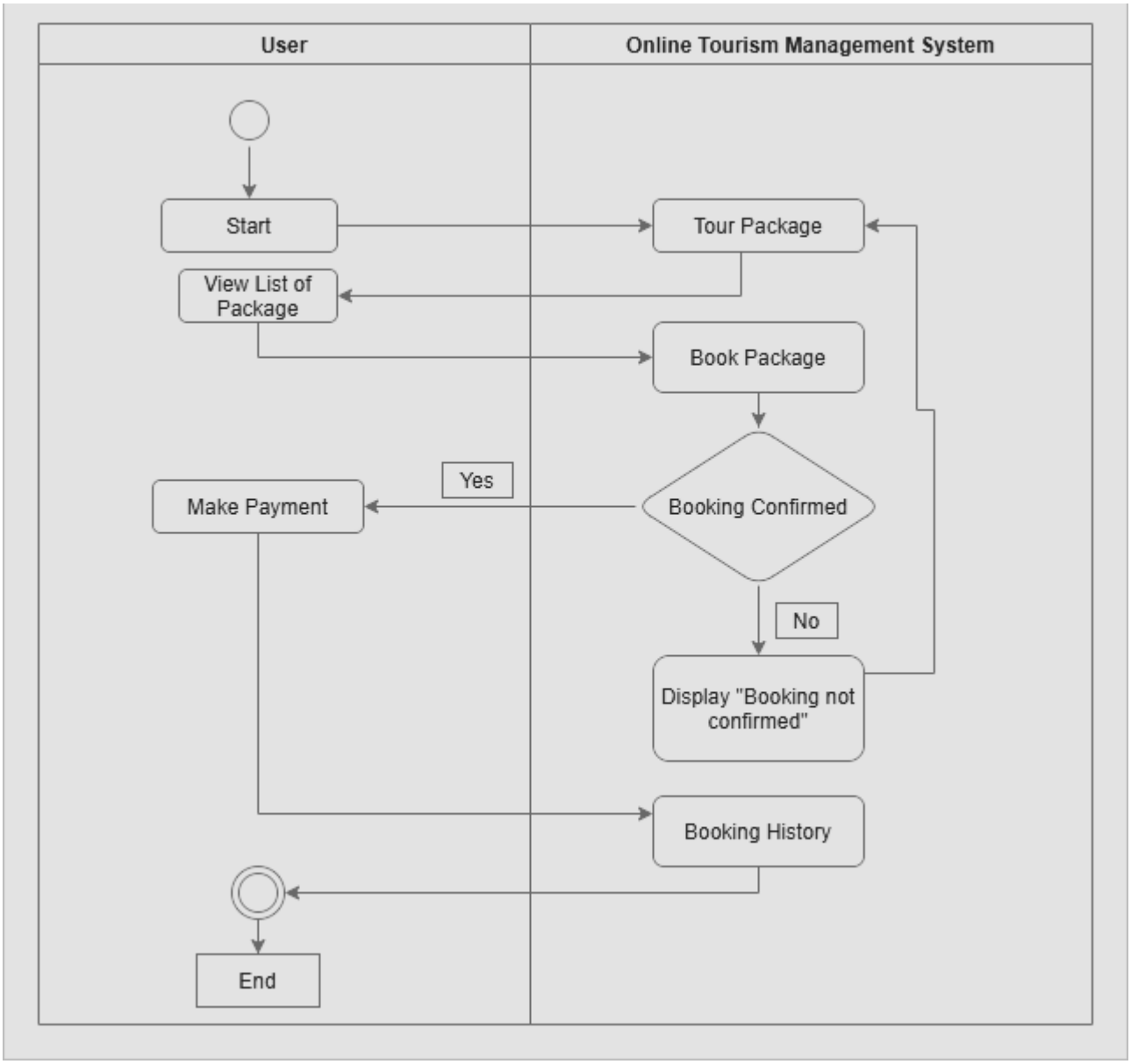


Here, Admin will have the access to Add/Remove packages, answer inquiries, etc. Whereas, Users can view package lists, make inquiries, make payments, book packages, etc.

5.Draw Activity diagram in argo uml.

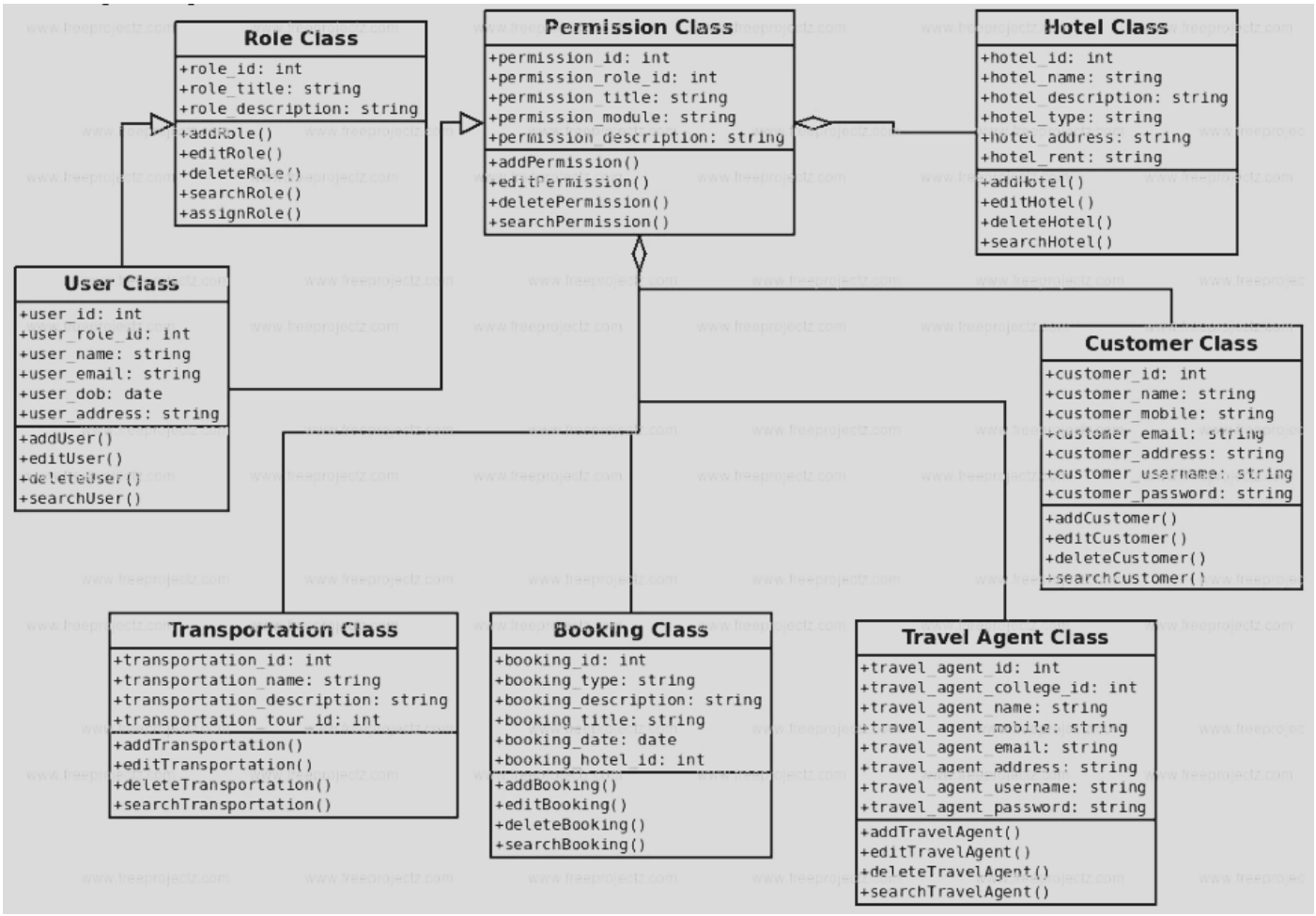
Activity diagrams in UML display the functionalities of various activities and flow in management processes and software systems. The flow in the activity diagram can be sequential, branched, or concurrent.

Admin can view the list of users. Admin can manage the category of packages and can update all package detail. Admin can view booking history and detail. Admin can manage inquiries, issues, payments, and transactions.



Online customers can browse or search packages, view specific packages, view them, book, and checkout. Users can view booking history at any time. Users can make payments for the booking and view the payment history.

6.Draw class diagram in argo UML.



Classes

.Travel agency class: manage all the operations of travel website.

.Customer class: manage all the operations of customer.

.Booking class: manage all the operatins of bookings.

.Hotels class: manage all the operations of hotels.

.Payment class:manage all the operations of payment.

Classes and their methods of Travel Agency Management System Class Diagram

Travel Agency Methods: addTravel Agency(), edit Travel Agency(), delete Travel Agency(), update Travel Agency), saveTlaver Agency)searchTravel Agency)

Customer Methods: addCustomer(), ed Customer(), deleteCustomers, updateCustomer), savelCustomer() searchCustomer)

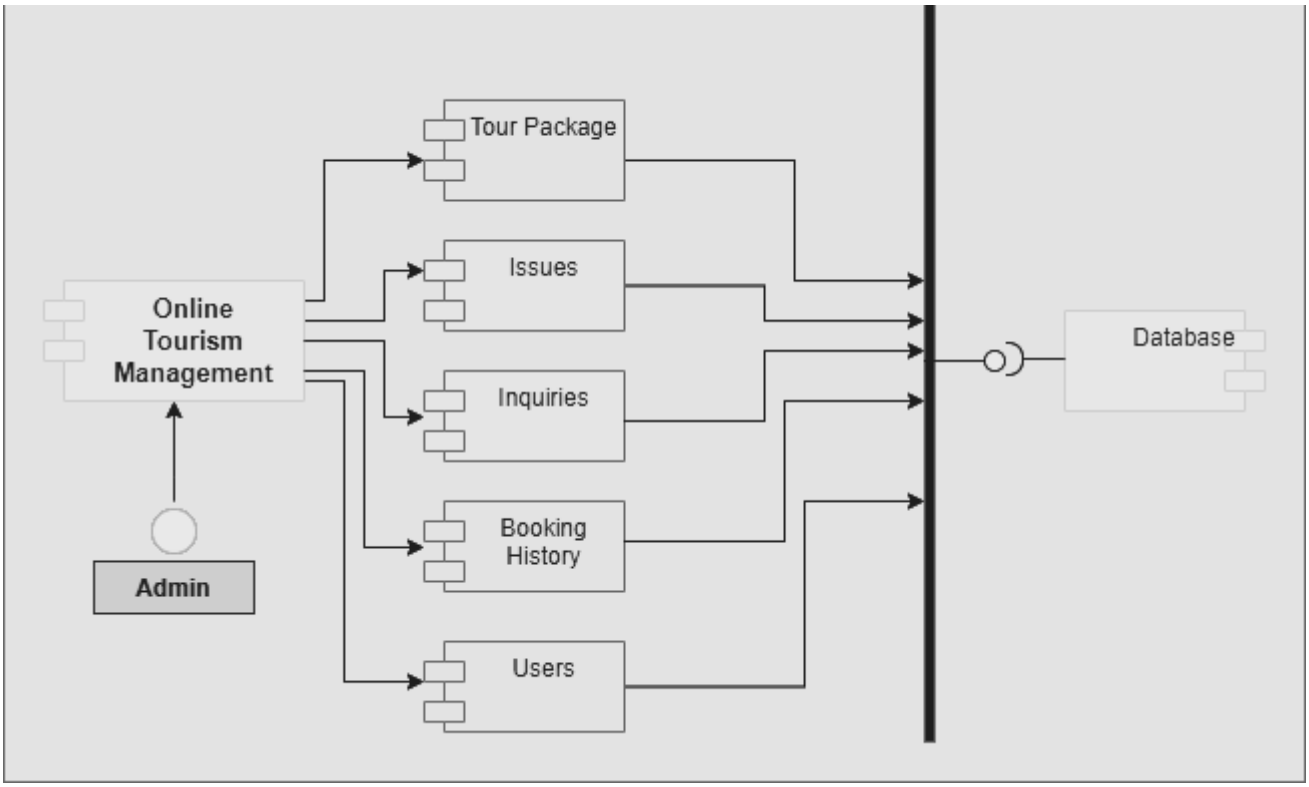
Bookings Methods: addBookings), edtBoolongs(), deleteBookings(), updatefilookings(), savefiookmgs(), searchBookings

Hotels Methods: add-Hotels) edotes), detefeHoters() update Hotels(), savel totes(), searchHotels)

Payments Methods: addPayments(), edtPayments(), deletePayments), updatePayments(), savePayments(); searchPayments

7 . Draw the component diagram in argo uml.

In the diagram, it can be seen that there are components namely product, order, customer, and account.

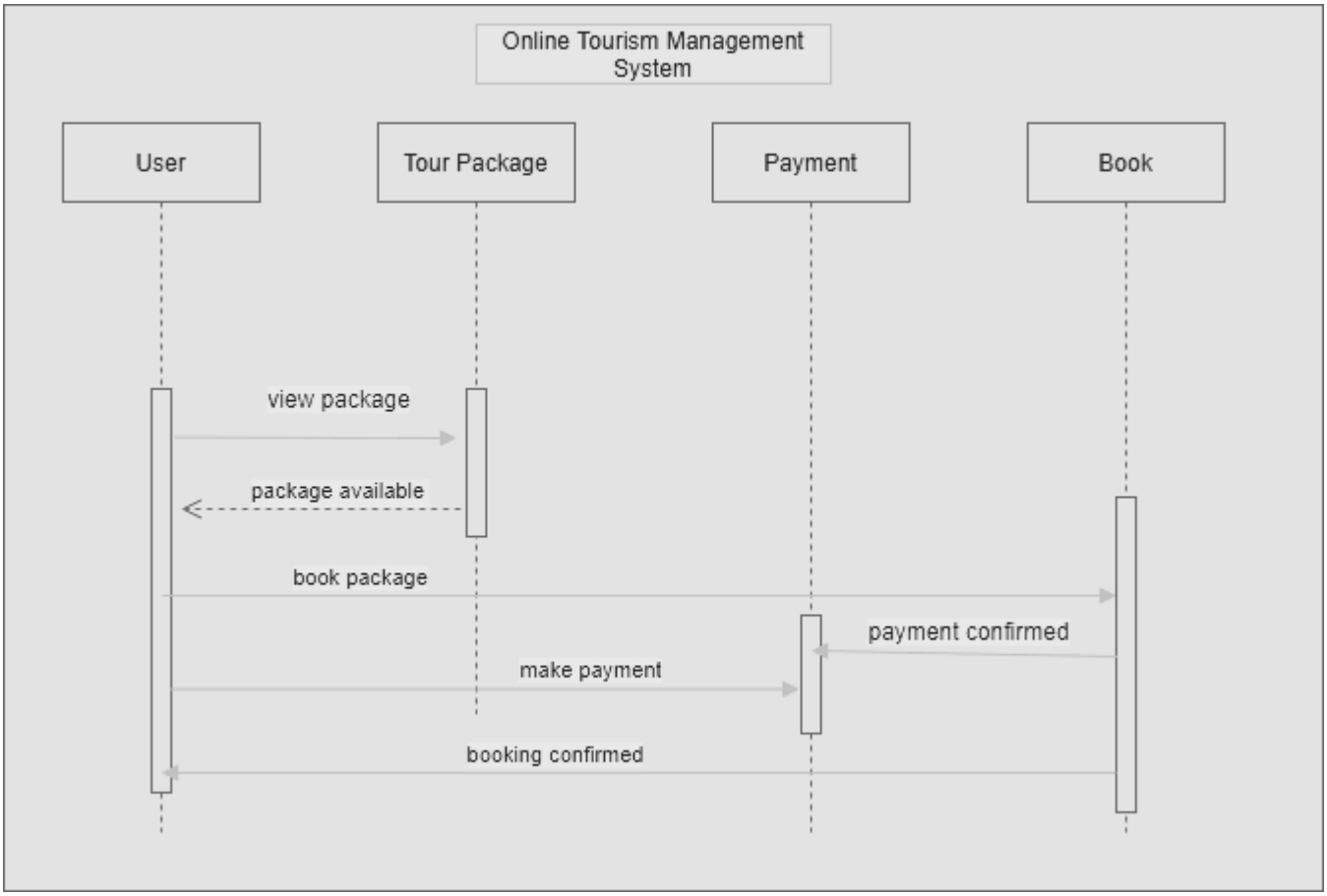


it shows how the customer component connects to the other components while using the system. Everything from the account details to product booking to payment flow can be seen in the component diagram.

Users can view all the packages according to the location or types of holidays and adventures. Users can check the price or select any holiday package according to the requirement. The online tourism management system will take responsibility for all the needful things about the customer requirement.

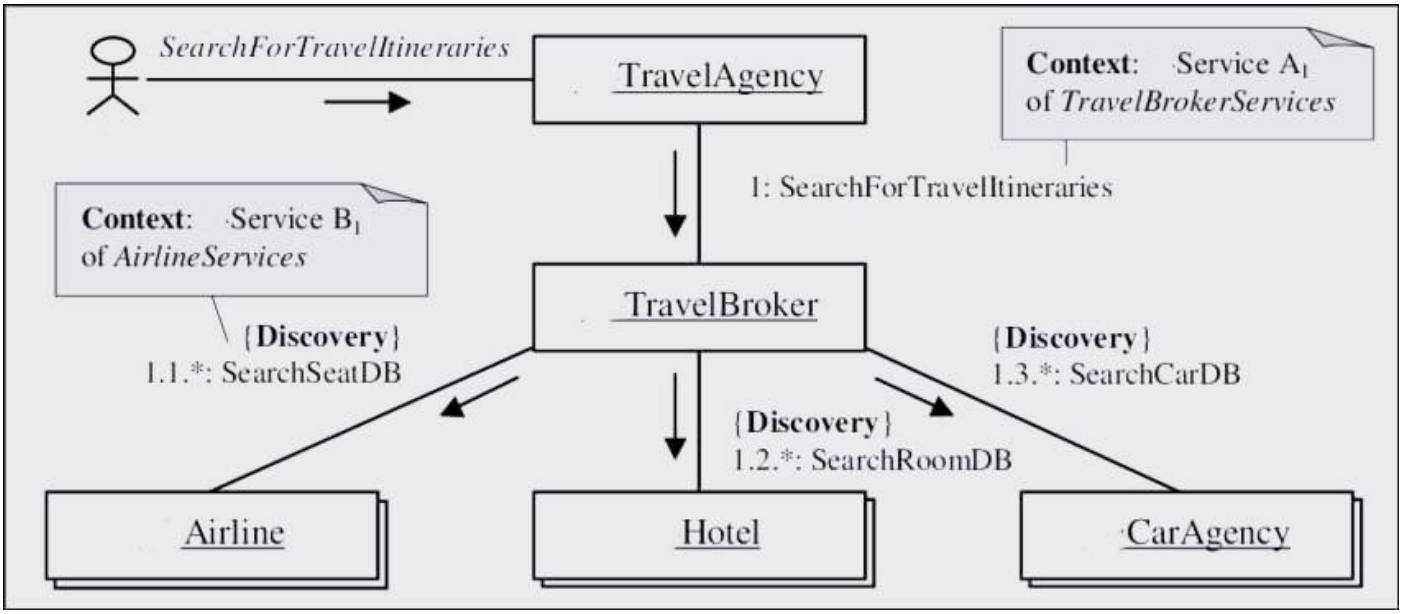
8.Draw sequence diagram in argo uml.

Sequence diagrams in UML are used to better understand how tasks within a project will function, overlap, and move between objects or components. Sequence diagrams display step-by-step interactions between objects and the order in which those interactions occur.



Users can log in and register in the application at any time from anywhere and check their booking history also can make a new booking. Users can search for packages, view the price of a selected package, book the package, and make payments for the booking. As soon as the payment is confirmed, the user's booking will be confirmed.

9.Draw collaboration diagram in argo uml.



- Customer signs up or login
- The customer search for destination,budged ,airline,hotel etc.
- The customer slects a suitable package.
- System collects traveler details in database.
- The customer pays through secure gateway.
- After successful payment booking is saved in thr system.

INDEX