ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY (ASTU)

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

DEPARTEMENT OF COMPUTER SCIENCE AND ENGINEERING (CSE)

Fundamentals of Software Engineering

GROUP ASSIGNMENT(section 2)

Prepared by:-	
1.Begonet Debebe	**Ugr/30244/15**
2.Bekam Genene	**Ugr/30253/15**
3.Enderias Eshetu	**Ugr/30469/15**
4.Kenenisa Beyan	**Ugr/30772/15**
5.Yeabsira Goitom	**Ugr/31390/15**
6.Yobsan Girma	**Ugr/31417/15**

Submitted to:- Dr. Sintayew Hirpesa
Submission date:- December/19/2024

Table of Contents

CHAPTER ONE: DESCRIPTION OF MAJOR SYSTEM

INTRODUCTION

The rapid advancement of technology has revolutionized various industries, including tourism. As one of the most vital sectors globally, tourism significantly contributes to cultural exchange, economic growth, and social development. With the growing adoption of digital platforms, web-based applications have become essential tools for connecting travelers with destinations, services, and experiences.

Traditional tourism planning often relied heavily on guidebooks, travel agencies, and word-of-mouth recommendations. While effective in the past, these methods have limitations in terms of accessibility, real-time updates, and personalization. The rise of web-based tourism applications has transformed the way people plan and experience travel. By offering comprehensive information, interactive features, and seamless connectivity, such platforms empower users to explore destinations efficiently and make informed decisions. A web-based tourism application provides a centralized platform where users can access a wealth of information about destinations, accommodations, attractions, and local services. Accessible via any device with an internet connection, these applications offer features such as reviews, ratings, interactive maps, and travel itineraries. They not only enhance convenience but also foster an engaging and informed travel experience.

The benefits of web-based tourism applications are numerous and address the evolving needs of travelers:

- **Convenience:** Users can explore and plan trips anytime, anywhere, without the need for physical travel agencies.
- **Personalization:** Tailored recommendations and itineraries based on user preferences enhance the travel experience.
- **Real-Time Updates:** Instant access to current information, including weather, events, and availability, ensures accurate planning.
- **Cost-Effectiveness:** Travelers can compare prices and choose options that fit their budgets.
- **Enhanced Accessibility:** Destinations and services become discoverable to a broader audience, including underserved regions.

This document presents the vision for a web-based tourism application tailored to Ethiopia. The platform will highlight the country's rich cultural heritage, diverse landscapes, and vibrant traditions, creating opportunities for tourists to connect with authentic and memorable experiences. By leveraging modern web technologies, this application aspires to serve as a comprehensive gateway for both domestic and international travelers exploring Ethiopia

1.1 DESCRIPTION OF MAJOR SYSTEMS

1.1.1 DESCRIPTION OF THE EXISTING SYSTEM

The existing tourism system in Ethiopia predominantly relies on traditional, manual methods for trip planning and destination discovery. Travelers often depend on printed travel guides, word-of-mouth recommendations, or physical visits to travel agencies to gather information. These methods are time-consuming, lack personalization, and often provide outdated or limited details about destinations, accommodations, and services.

Reviews and feedback about attractions and services are typically shared informally, making it difficult for new travelers to access reliable insights. Booking accommodations, tours, and other services often requires direct communication with providers, which can be inefficient and prone to delays. Additionally, the absence of centralized information limits travelers' ability to compare options and make informed decisions.

This traditional approach also presents challenges for local tourism providers, who often struggle to reach a broader audience and showcase their offerings effectively. The lack of digital tools to promote destinations and manage bookings further compounds these issues.

The proposed system aims to digitize and centralize tourism information, providing an interactive and accessible platform for travelers to discover, plan, and book trips efficiently. It will connect tourists with local service providers, enabling a more seamless and engaging travel experience.

1.1.2 DRAWBACKS OF THE PROPOSED SYSTEM

- **Limited social interaction:** The platform reduces direct, face-to-face interactions between travelers and local tourism providers, which might impact the personal touch of traditional travel planning.
- **Dependence on technology:** Travelers without internet access or sufficient digital literacy may find it challenging to use the platform effectively.
- **Prone to misinformation:** User-generated reviews and ratings could be manipulated or biased, leading to inaccurate representations of destinations or services.
- **Over-reliance on screens:** The digital nature of the platform may result in excessive screen time for users during the planning process.
- **Technical challenges:** The system could encounter technical issues such as server downtime or bugs, which might disrupt the user experience.
- **Training requirements:** Tourism service providers and administrative staff may require additional training to effectively use the platform.

1.2 PURPOSE OF THE PROJECT

The purpose of this project is to develop a web-based tourism platform that leverages technology to provide an efficient, interactive, and accessible solution for travel planning. The platform aims to showcase Ethiopia's diverse attractions, connect travelers with local services, and enhance the overall travel experience. It will serve as a comprehensive and user-friendly tool, offering high-quality content, seamless navigation, and features that cater to the needs of both domestic and international travelers.

1.3 SCOPE OF THE PROJECT

The scope of this project encompasses several key areas to ensure the development of a comprehensive, user-friendly, and feature-rich tourism platform. This scope clarifies the deliverables, timelines, and boundaries of the project, aligning all stakeholders on expectations and outcomes, such as:

1. **Tourism Promotion Goals**

    - Highlight Ethiopia's rich cultural heritage, diverse landscapes, and unique attractions.
    - Provide a platform to connect travelers with local service providers, including hotels, tour operators, and transportation services.

2. **Content and Service Outline**

    - Detail the attractions, accommodations, and travel services covered in the platform.
    - Include categories for cultural, historical, natural, and adventure-based experiences.

3. **Modes of Delivery**

    - Utilize various digital technologies to present content and services, including:

        - Interactive maps and virtual tours
        - Reviews and rating systems
        - Booking tools for accommodations and tours
        - Travel itineraries and planning assistance
        -

4. **Intended User Demographics**

    - Identify the primary user groups, including:

- Domestic travelers
- International tourists
- Local service providers (e.g., hotels, guides, restaurants)

5. **Feedback and Review Mechanisms**

    - Provide tools for users to:

        - Leave reviews and ratings for attractions and services
        - Access recommendations based on traveler feedback
        - Report inaccurate or outdated information

6. **Project Timeline and Resources**

    - Define the timeline for each phase of development, including:

        - Initial design, content curation, and development
        - Platform testing and quality assurance
        - Launch and marketing efforts
    - Identify budgetary needs and resource allocation, considering factors like platform maintenance and marketing.

Key Features and Functions

- **Search and Discovery:**

    - A search system allowing users to find attractions, services, and experiences by location, category, or interest.

- **Booking System:**

    - Seamless options for booking accommodations, tours, and tickets directly through the platform.

- **Interactive Maps:**

    - Detailed maps highlighting tourist destinations and their locations.

- **User Profiles:**

    - Personalized accounts for users to save itineraries, leave reviews, and track bookings.

- **Travel Recommendations:**

    - AI-powered suggestions tailored to user preferences and past activities.

Limitations

- **Internet Requirement:**
  - A stable internet connection is necessary to access the platform's features.
- **Language Support:**
  - Limited support for multiple languages in the initial phases.
- **Device Compatibility:**
  - Optimized for smartphones and web browsers but may not be fully functional on all devices.

Significance of the Project

- **Enhanced Tourism Experience:**
  - Provides tourists with a centralized platform to plan and book trips seamlessly.
- **Promotion of Local Tourism:**
  - Supports local businesses by connecting them with a broader audience.
- **Increased Accessibility:**
  - Makes Ethiopia's attractions more accessible to both domestic and international travelers.
- **Cost and Time Efficiency:**
  - Simplifies trip planning, reducing the need for physical travel agencies.

1.4 OBJECTIVE OF THE PROJECT

The project aims to develop a comprehensive web-based tourism platform that enhances the travel experience for users by providing a user-friendly interface and a robust backend system. This platform will improve accessibility by offering centralized information about Ethiopian attractions, accommodations, and services. It will foster efficiency through seamless booking systems, interactive maps, and personalized recommendations that support itinerary planning and decision-making. Additionally, features like reviews, ratings, and travel progress tracking will assist users in planning and enjoying their trips effectively, while strong security measures will protect user data. The platform will serve as a valuable resource for both domestic and international travelers, helping them explore Ethiopia in an informed, engaging, and efficient way.

The objectives of the project are to design and develop:

- A database that stores detailed information on Ethiopian destinations, hotels, restaurants, tours, and local services
- A system to manage and regularly update content, including new travel destinations and services
- A review and rating system allowing users to share experiences and rate services, accommodations, and attractions
- A search engine for travelers to explore destinations, read reviews, and compare options
- A booking system for reservations and tickets for accommodations, tours, and other travel services
- A personalized recommendation engine based on user preferences, previous searches, and feedback
- An interactive map to help travelers discover destinations and nearby services
- A user account system where travelers can save itineraries, track bookings, and access travel history
- A system that tracks user activities, including booking history and reviews, for personalized suggestions
- A secure platform with encryption to ensure the protection of user data, including payment and personal information
- A messaging system for direct communication between travelers and service providers

A responsive, web-based platform accessible on various devices, ensuring a smooth user experience across desktops, tablets, and smartphones

1.5 SOFTWARE DEVELOPMENT APPROACH

Successful software projects are managed efficiently. To manage a project effectively, the development team must choose a software development methodology that aligns with the project's goals, requirements, and constraints. Different methodologies have distinct strengths and weaknesses, and selecting the right one is crucial for achieving success. Common approaches used in the software development process include Waterfall, V-Shaped, Structured Evolutionary Prototyping, Rapid Application Development (RAD), Incremental Development, Spiral Development, the (Rational) Unified Process, Agile, and others.

For our project, we have selected the **Agile methodology**, which is well-suited for web-based platforms like ours. Agile emphasizes flexibility, collaboration, and iterative progress, making it ideal for projects where user feedback and frequent updates are essential. In Agile, the development process is broken down into smaller, manageable iterations, allowing the development team to refine and enhance the platform incrementally.

Agile is particularly valuable for projects that require constant user input, adaptability to changes, and the ability to prioritize features based on evolving needs. This methodology enables faster releases, continuous improvements, and more user-centered design, all of which are critical for creating a tourism platform that meets the diverse needs of travelers and local service providers. The Agile approach fosters collaboration between stakeholders, including developers, designers, and users, and facilitates quicker adaptations to feedback or market demands.

There are various frameworks under the Agile umbrella, such as Scrum, Extreme Programming (XP), and Kanban, each offering distinct processes and techniques. For this project, we plan to implement **Scrum**, which focuses on short sprints, regular feedback loops, and continuous delivery of functional increments. This will ensure that the platform evolves with user needs, is regularly updated with new content, and remains adaptable throughout the development cycle.

Figure 1: Agile

1.6 SOFTWARE PROCESS MODEL

For the modeling and documentation of our web-based tourism platform, we have adopted a practice-based approach using the **Agile method**. This approach is well-suited for our system as it focuses on rapid development while minimizing the time spent on extensive analysis and design. The Agile methodology is particularly effective for projects that require flexibility and the ability to quickly adapt to changing user needs and feedback, which is critical for a dynamic platform like a tourism service.

The Agile methodology for this project is characterized by the following features:

- **Well-suited for web-based platforms** with evolving content and features.
- **Prioritizes rapid development and coding** over extensive upfront design, allowing for faster delivery of functional software.
- **Delivers incremental improvements** with frequent releases, ensuring that the platform continuously evolves and improves based on user feedback.
- **Adapts quickly to changing requirements**, ensuring the platform remains aligned with user needs and market demands.
- **Facilitates regular collaboration** between stakeholders (developers, designers, and users) to ensure that each feature meets the expectations of travelers and service providers.

This iterative and flexible development approach ensures that our tourism platform remains user-centered, with frequent updates and refinements to meet the needs of both local and international travelers

1.7 REQUIREMENT ELICITATION APPROACH

The primary source of data for our proposed tourism platform project comes from analyzing information gathered from various sources, including potential users, service providers, and industry research. Additionally, we employed the following methodologies to gather project requirements:

- **Document Analysis:** This involves reviewing existing documentation of similar tourism platforms, industry reports, and government travel information to extract relevant data and insights that can guide the development of our system. This helps us understand the types of features, services, and content most beneficial to travelers and service providers.
- **Practical Observation:** This method entails observing interactions between travelers and current tourism platforms, such as websites, mobile apps, or in-person service experiences, to identify which content and features engage users most. We also observe how service providers manage their offerings and respond to customer needs, helping us design a system that aligns with both user expectations and service provider capabilities.
- **Interviews and Surveys:** Engaging directly with users—both travelers and service providers—through structured interviews and surveys is crucial. This enables us to gather specific, actionable insights into their needs, preferences, and pain points. Feedback from these interviews helps to refine features like search functionality, booking systems, and user interface design.
- **Workshops and Focus Groups:** Facilitating workshops with potential users, including both tourists and local service providers, allows for collaborative brainstorming and idea validation. Focus groups help us uncover deeper insights into user behavior and priorities, especially concerning personalized travel recommendations, ratings, and review systems.
- **Prototyping and Feedback Loops:** Prototyping allows us to create early versions of key platform features, which can be tested by users. Their feedback during testing sessions helps us make iterative improvements and refine the platform to meet their needs effectively.
- **Competitive Analysis:** Analyzing similar tourism platforms (not mentioning specific ones) and identifying the strengths and weaknesses of their features allows us to develop a more competitive and user-friendly solution. This also provides insights into industry best practices, design trends, and common user expectations.

By combining these elicitation techniques, we ensure a comprehensive understanding of the project requirements, focusing on both the technical needs of the system and the practical requirements of users. This approach ensures that our platform will deliver valuable, relevant features that enhance the travel experience while also supporting local service providers.

1.8 DEVELOPMENT REQUIREMENTS

1.8.1 SOFTWARE REQUIREMENTS

The following software tools and technologies will be used in the development of the tourism platform:

General Activity	Tool Used
DBMS (Database Management System)	MySQL
Operating System	Microsoft Windows 10
Frontend Languages	HTML, CSS, JavaScript
Documentation Tools	MS Office, Google Docs
Browsers	Microsoft Edge, Chrome, Firefox
Web Server	XAMPP
Backend Language	PHP
Code Editor	Visual Studio Code, Notepad++

1.8.2 HARDWARE REQUIREMENTS

The following hardware specifications are required to develop and run the tourism platform:

Hardware Requirement	Specifications
Processor	Intel Core i3/i5/i7/i9-10xxx
RAM	4GB or higher
Storage	1GB (minimum for development)
Display	For all devices (responsive design)

1.9 REQUIRED RESOURCES WITH COST

The following resources are necessary for the development and operationalization of the tourism platform:

Cost Type	Quantity/Amount	Unit Price	Total Price
Paper	25 sheets	$10	$250
Hard Drive	1 unit	$500	$500

Cost Type	Quantity/Amount	Unit Price	Total Price
Transport	50 miles	$20	$1000
Time	100 hours	$50/hr	$5000

1.10 Feasibility Study

Feasibility analysis helps determine whether the TMS project is practical and worth pursuing by examining technical, operational, and economic aspects. The purpose of this study is to assess the technical, economic and operational aspects of the project in order to determine if it is feasible and worth pursuing.

1.10.1 Technical Feasibility

Technical feasibility test examines the technical aspects necessary for the successful implementation of a project. To assess the technology requirements for the project, it's essential to identify the necessary hardware, software, and network infrastructure, determining if existing technology can meet project needs or if new solutions are required.

- ✓ **Technology Availability:**
  - Suitable programming languages, frameworks, and hosting platforms are available and well-documented.
  - APIs like Google Maps and Stripe are easily integrable and well-documented.
- ✓ **Development Team Skills:**
  - The project requires a skilled team of frontend, backend, and database developers.
- ✓ **Infrastructure:**
  - Necessary hardware and software tools are affordable and widely available.
- ✓ **Scalability:**
  - The system can scale to accommodate increasing users and data in the future.

1.10.2 Operational Feasibility

Operational feasibility assesses how well a proposed project aligns with the existing operational capabilities and processes of an organization. It evaluates whether the organization can effectively implement and manage the project within its current operational framework.

- ✓ **User-Friendly Interface(**Target Users**):**
  - Tourists: Must be able to easily browse destinations, book tours, and make payments.
  - Travel Agents: Should have features for managing bookings, customers, and schedules.
  - A responsive, intuitive design ensures minimal training is required.
  - Mobile-first design ensures usability on smartphones, the most common device for travel planning.
- ✓ **Adoption by Stakeholders:**
  - Tourists benefit from convenience, flexibility, and real-time updates on bookings.
  - Travel Agencies can use the platform to advertise their services and manage operations more effectively.
- ✓ **Operational Challenges:**
  - **Data Accuracy:** The system must have accurate, real-time information about tour availability and pricing.
  - **Customer Support:** Providing 24/7 assistance through chatbots or live agents may be necessary, especially for tourists in different time zones.
- ✓ **Maintenance and Upgrades:**
  - The system must be regularly maintained to address bugs, improve performance, and incorporate new features (e.g., seasonal packages or regional customization).

1.10.3 Economic Feasibility

The project is economically feasible, with high revenue potential and manageable costs.

- ✓ **Development Costs:**
  - Initial development costs (~$2,000–$5,000) are reasonable for a web-based system.
  - Hosting and API integration fees (~$200–$500/month) ensure a scalable solution without heavy infrastructure investments.

- ✓ **Revenue Potential:**
  - ➢ Income can be generated through subscription fees, advertisements, or commission from travel bookings.
- ✓ **Return on Investment (ROI):**
  - ➢ The system has high revenue potential due to the growing tourism industry and increasing demand for digital travel management platforms.
  - ➢ Short-Term ROI: Revenue can cover operational costs within the first year, especially if targeting high-demand markets.
  - ➢ Long-Term ROI: Potential for scaling the platform internationally, attracting more users, and increasing revenue streams.

- ✓ **Cost Optimization**
  - ➢ Open-Source Tools: Use open-source frameworks and databases to reduce licensing costs.
  - ➢ Cloud Solutions: Avoid expensive server hardware by using cloud services that offer pay-as-you-go pricing.
- ✓ **Economic Risks**
  - ➢ Initial Marketing Costs: Heavy marketing may be needed to attract users initially, requiring a higher upfront investment.
  - ➢ Competition: Competing with established platforms may require pricing adjustments or unique features.

1.11 Test Plan

The test plan outlines the approach, scope, resources, and schedule for the testing activities required to validate the Tourism Platform against the requirements specified in the Software Requirements Specification (SRS) document. The goal of testing is to ensure that the platform meets functional, non-functional, and performance requirements, guaranteeing a seamless user experience.

Objectives

- Verify that the platform meets the functional requirements as specified in the SRS document.
- Ensure compliance with non-functional requirements such as performance, usability, security, and compatibility across multiple devices and browsers.
- Identify and resolve defects before deployment to ensure high-quality functionality and user experience.

Scope

The test plan will validate the following modules based on the SRS:

1. **User Management:**
    - User registration, login, and authentication.
    - Role-based access (Tourist/Service Provider/Admin).
2. **Tour Management:**
    - Tour search by location, price, rating, and categories.
    - Viewing tour details (itinerary, availability, pricing).
3. **Booking and Payments:**
    - Booking process (tour selection, confirmation, user details).
    - Integration with secure payment gateways.
4. **User Engagement:**
    - Reviews and ratings for tours and services.
    - Customer support inquiries and resolution.
5. **Admin Panel:**
    - Managing tour listings, user accounts, and bookings.
6. **Performance and Security Requirements:**
    - Ensure quick response times for browsing and booking activities.
    - Protect user data, and ensure secure payment processes.

Test Types and Strategy

A. Test Levels

1. **Unit Testing**
    - Test individual components such as user registration, search filters, and payment processing.

2. **Integration Testing**

    - Test interactions between modules, such as the integration of user registration with tour bookings and payment systems.

3. **System Testing**

    - Validate end-to-end workflows based on user scenarios, such as browsing tours, booking a tour, and making payments.

4. **User Acceptance Testing (UAT)**

    - Test the platform with actual users (travelers and service providers) to ensure the platform meets business requirements and user expectations.

B. Testing Types

- **Functional Testing:** Ensure that each feature (user registration, booking process, etc.) works as expected according to the SRS.
- **Non-Functional Testing:** Validate platform usability, scalability, security, and performance under different conditions.
- **Regression Testing:** Retest previously tested features after updates or bug fixes to ensure no new issues arise.
- **Compatibility Testing:** Test platform compatibility across devices (smartphones, tablets, desktops) and browsers (Chrome, Firefox, Safari, Edge).

Test Environment

- **Hardware:** Desktops, laptops, tablets, and mobile devices with varying specifications.
- **Software:**
    - **Platforms:** Mobile (iOS, Android), Desktop (Windows, macOS).
    - **Browsers:** Chrome, Firefox,Edge.
    - **Database:** MySQL/PostgreSQL with predefined test data.
- **Tools:**
    - Selenium (for Functional Testing).
    - JMeter (for Performance Testing).
    - Postman (for API Testing).

Risks and Mitigation

Risks

- **Incomplete Requirements:** Missing or unclear requirements may lead to inadequate testing or missed features.
- **Resource Availability Issues:** Limited availability of testing resources (such as devices or personnel).
- **Delays in Development:** Delays in the development process could impact the testing timeline and delivery.

Mitigation Strategies:

- Regular communication with stakeholders to ensure complete and accurate requirements.
- Prioritize testing critical features (booking process, payment gateway) early in the testing phase.
- Maintain a flexible testing schedule to accommodate changes in development and resource availability.

CHAPTER TWO: PROPOSED SYSTEM

2.1 Functional Requirements

Functional Requirements for Tourism Management System

Functional requirements define the specific operations or behaviors that the system must perform. These requirements outline the system's core features and functionalities based on user needs, ensuring a smooth and engaging experience for both travelers and service providers.

1. User Management

1.1 User Registration

- The system shall allow new users (tourists, tour operators, administrators) to register using their personal details such as name, email, contact number, and password.
- A confirmation email shall be sent to the user upon successful registration.

1.2 User Login

- The system shall authenticate users using their login credentials (email and password).

- A password recovery option shall be provided via email to recover forgotten passwords.

1.3 User Roles

- The system shall support different user roles, including:
    - **Tourist/Traveler**
    - **Tour Operator/Service Provider**
    - **Admin**

2. Tour Package Management

2.1 Package Creation (Tour Operators)

- Registered tour operators shall be able to create new tour packages.
- Each tour package shall include:
    - Tour name
    - Destination
    - Duration
    - Cost (with currency)
    - Itinerary details
    - Available dates
    - Maximum number of participants
    - Images (optional)

2.2 Package Viewing (Tourists)

- Tourists shall be able to view a list of available tour packages.
- Travelers can filter and search for packages based on:
    - Destination
    - Cost range
    - Duration
    - Ratings

2.3 Package Booking

- The system shall allow tourists to book a selected tour package.
- A confirmation email shall be sent to the traveler upon successful booking, with details of the tour.

2.4 Package Modification (Admin)

- The admin shall have the ability to update, modify, or remove tour packages.

3. Payment Management

3.1 Payment Integration

- The system shall integrate with popular payment gateways (e.g., PayPal, Stripe, credit/debit cards) for easy payment processing.

3.2 Payment Confirmation

- The system shall confirm payments and generate a transaction receipt for the traveler.
- All payment transactions shall be logged and stored in the system.

3.3 Refund Management

- The system shall allow travelers to request refunds for cancellations.
- Refunds will be processed based on the platform's cancellation policy.

4. Booking Management

4.1 View Booking History

- The system shall allow tourists to view their past and current bookings.

4.2 Manage Bookings

- Tourists shall be able to cancel or reschedule bookings (if permitted by the policy).

4.3 Notifications

- The system shall send notifications to users regarding:
  - Booking confirmations
  - Payment receipts
  - Tour reminders (e.g., 2 days before the tour)
  - Cancellations or changes to bookings

5. Search and Filter

- The system shall allow tourists to search and filter for:
  - Tour packages by destination, date, duration, and price.
  - Hotels, flights, or transportation services (if applicable).

6. Review and Ratings

6.1 Add Review

- Tourists shall be able to review and rate completed tours based on their experience.

6.2 View Reviews

- Tourists shall be able to view ratings and reviews for each tour package.

7. Reports and Analytics (Admin)

- The system shall generate various reports for admins, including:
  - Number of bookings per month/year
  - Most popular destinations
  - Performance reports for tour operators

8. Security Requirements

8.1 Authentication and Authorization

- The system shall ensure secure access using role-based permissions, restricting access based on user roles.

8.2 Data Encryption

- User data and payment information shall be encrypted both during storage and transmission to ensure privacy and security.

8.3 Secure Payment Handling

- Payment transactions shall comply with PCI-DSS standards to ensure secure handling of financial data.

9. Notifications

- The system shall send notifications to users regarding booking confirmations, payment receipts, and tour reminders.
- Users will have the option to opt-in or opt-out of promotional notifications

10. Help and Support

- The system shall provide a support section that includes:
  - Frequently Asked Questions (FAQs) to answer common user inquiries.

- Contact support through email or live chat for assistance with any issues or questions.

2.2 Non-Functional Requirements

Non-Functional Requirements (NFRs) define the overall quality and performance characteristics of the system. These ensure that the system functions efficiently, is secure, scalable, and provides an excellent user experience across different platforms and devices.

1. Performance Requirements

- The system shall handle up to 100 concurrent user bookings or searches within 1 minute without delays or system lags.
- Search and filtering operations (e.g., by destination, price, date) shall return results in less than 2 seconds.
- Payment processing and confirmation shall be completed within 5 seconds of submission.
- The system shall ensure fast page load times, with no page exceeding 3 seconds to load on average.

2. Usability Requirements

- The system shall have a user-friendly and intuitive interface that allows seamless navigation for travelers, tour operators, and administrators.
- The application shall be easy to use for individuals with minimal technical expertise and provide clear instructions where necessary.
- The system must be responsive and adapt seamlessly to desktops, tablets, and mobile devices.
- It shall include accessibility features for users with disabilities, such as screen readers and easy navigation controls.

3. Security Requirements

- The system shall encrypt sensitive user data (e.g., passwords, payment details, personal information) using AES 256-bit encryption.
- All payment transactions shall comply with the PCI-DSS standards to ensure secure processing of payment details.
- User authentication shall include multi-factor authentication (MFA) to improve security, especially for critical actions like booking and payment.
- Access control shall be role-based (Admin, Tour Operator, Traveler) to restrict unauthorized access to sensitive features.
- User data shall be stored securely with strong encryption and comply with privacy regulations.

4. Scalability Requirements

- The system must scale to handle high traffic volumes during peak seasons, such as holidays or special promotions, with the ability to manage up to 1,000 concurrent users.
- The system shall support the addition of new features or modules, such as hotel bookings, flight reservations, or review management, with minimal disruption to the existing services.

5. Reliability Requirements

- The system shall have an uptime of 99.9% throughout the year, ensuring minimal downtime and continuous service availability.
- In case of failure, the system shall incorporate failover mechanisms to restore service within 10 minutes.
- The system must be designed with fault-tolerant mechanisms to ensure reliability even during unexpected outages.

6. Availability Requirements

- The system shall be available 24/7 to allow global users to browse and book tours at any time.
- Maintenance and system updates shall be scheduled during off-peak hours to minimize disruption to users.
- The system shall maintain redundant servers and backup data to ensure continuous availability.

7. Maintainability Requirements

- The system shall be modular in design to allow easy updates, bug fixes, and the integration of new features.
- The codebase must follow industry-standard practices to ensure maintainability, code clarity, and ease of debugging.
- Documentation shall be maintained to provide clear guidance for future development and troubleshooting.

8. Portability Requirements

- The system must be compatible with all major browsers, including Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, as well as on mobile devices (iOS and Android).
- The application shall provide a consistent user experience across different screen sizes, from desktops to mobile phones, by using responsive design principles.

- The system shall not require specific hardware to function but shall work optimally on widely used devices.

9. Compliance Requirements

- The system shall adhere to GDPR (General Data Protection Regulation) standards to protect user data and privacy for European users.
- The system shall comply with local data protection laws and regulations in the regions where it operates, including user consent for collecting personal information.
- The application shall follow any industry-specific regulations regarding user data, payment processing, and service offerings.

10. Disaster Recovery Requirements

- The system shall implement automated data backup mechanisms, ensuring that user data is regularly backed up and can be restored in case of system failure.
- A disaster recovery plan shall be in place to restore the system and user data with minimal disruption in the event of unexpected failures.
- Regular backups shall be tested to ensure that recovery is possible in different failure scenarios.

11. Logging and Monitoring

- The system shall log all critical activities, including user logins, bookings, payments, and system errors.
- Admins shall have access to real-time monitoring dashboards that display important metrics, such as server performance, active users, and transaction logs.
- Alerts shall be triggered for abnormal system activity, such as unauthorized login attempts or system errors, to allow for timely intervention.

2.3 Identification of Actors

Actor	Type	Description	Responsibilities
Traveler (End-User)	**Primary Actor**	A customer who uses the system to search, book, review, and manage travel experiences.	• Search for destinations, tours, and activities. • Book tours, accommodations, and activities. • Manage bookings (modify, cancel). • View and write reviews for tours, restaurants, and hotels. • Share travel experiences through photos and feedback.

Actor	Type	Description	Responsibilities
Tour Operator	**Primary Actor**	A service provider responsible for managing and offering tour packages.	• Create and manage tour listings, including destinations, itineraries, and prices. • Respond to customer inquiries. • Manage bookings and cancellations. • Offer promotions and discounts.
Admin	**Primary Actor**	A user with system-wide control, managing platform functionality and user roles.	• Manage user accounts (travelers, agents, etc.). • Monitor and manage system performance and content. • Oversee content approval (reviews, listings). • Generate reports (e.g., booking activity, user statistics). • Implement system updates and security protocols.
Payment Gateway System	**Secondary Actor**	An external system that processes secure online payments for bookings.	• Handle transactions made by travelers for tour packages and services. • Provide payment confirmation and receipts. • Securely transmit payment data between users and the system.
Review & Rating System	**Secondary Actor**	An external service that collects and manages user-generated content, such as reviews and ratings.	• Collect and display reviews for tours, hotels, and restaurants. • Aggregate ratings and facilitate filtering based on user feedback. • Moderate inappropriate content or spam.
Hotel & Activity Providers	**Secondary Actor**	External partners offering accommodations, restaurants, or activities included in tour packages.	• Provide availability and pricing for their services. • Confirm bookings made by travelers through the platform. • Update information about rooms, meals, or activities.
Email/SMS Notification System	**Secondary Actor**	External service that sends notifications to users regarding their bookings and updates.	• Send booking confirmations, cancellations, reminders, and special offers. • Notify users about updates or changes to their tours, activities, and accommodations.

Actor	Type	Description	Responsibilities
Customer Support Team	**Secondary Actor**	A team that assists users with any issues or inquiries during their use of the platform.	• Provide help via chat, email, or phone. • Assist with troubleshooting bookings, refunds, and cancellations. • Resolve user complaints or escalate to higher support levels when necessary.

2.4 Use Case Identification

1. Use Cases for Traveler (Primary Actor)

1. **User Registration**
    - Description: A new traveler registers by providing necessary details (name, email, contact, etc.).
2. **User Login**
    - Description: A registered traveler logs in with credentials (email and password).
3. **Search Tours**
    - Description: The traveler searches for available tours by destination, price range, and other filters.
4. **View Tour Details**
    - Description: The traveler views detailed information about a selected tour (itinerary, cost, availability).
5. **Book Tour**
    - Description: The traveler selects and books a tour package.
6. **Cancel Booking**
    - Description: The traveler cancels a previously booked tour, if applicable.

7. **View Booking History**

    - Description: The traveler views their past and upcoming tour bookings.

8. **Make Payments**

    - Description: The traveler processes payment for tour bookings through integrated payment systems.

9. **Provide Feedback**

    - Description: After completing a tour, the traveler provides a review or feedback on the experience.

2. Use Cases for Tour Agent (Primary Actor)

10. **Add Tour Packages**

    - Description: The agent adds new tours to the platform with relevant details (destination, itinerary, etc.).

11. **Update Tour Packages**

    - Description: The agent updates existing tour packages with new information (dates, prices, etc.).

12. **Remove Tour Packages**

    - Description: The agent removes a tour package from the platform when it's no longer available.

13. **View Bookings**

    - Description: The agent views and manages bookings made for their tours.

14. **Approve/Reject Bookings**

    - Description: The agent approves or rejects bookings based on tour availability or other criteria.

3. Use Cases for Administrator (Primary Actor)

15. **Manage User Accounts**

- Description: The admin manages user accounts, including approval, suspension, or deletion of accounts.

16. **Monitor System Logs**

    - Description: The admin monitors system activities, including user actions and errors, for system health.

17. **Generate Reports**

    - Description: The admin generates various reports related to bookings, user activities, and revenue.

18. **Manage System Settings**

    - Description: The admin configures system settings, such as user roles, payment configurations, etc.

4. Use Cases for Payment Gateway (Secondary Actor)

19. **Process Payment**

    - Description: The payment gateway processes payments made by travelers for bookings.

20. **Verify Transaction**

    - Description: The payment gateway verifies and confirms the payment transaction.

5. Use Cases for Email/SMS Notification System (Secondary Actor)

21. **Send Confirmation Emails**

    - Description: The system sends confirmation emails to users for bookings, cancellations, etc.

22. **Send Reminders**

    - Description: The system sends reminders to users about upcoming bookings and tours.

23. **Notify Cancellations**

- Description: The system notifies users about tour cancellations or changes in itinerary

6. Use Cases for Hotel/Transport Providers (Secondary Actor)

24. **Provide Availability Details**

- Description: Hotel and transport providers update the platform with available rooms, transportation, and pricing.

25. **Confirm Bookings**

- Description: Providers confirm bookings made by travelers for accommodations or transportation.

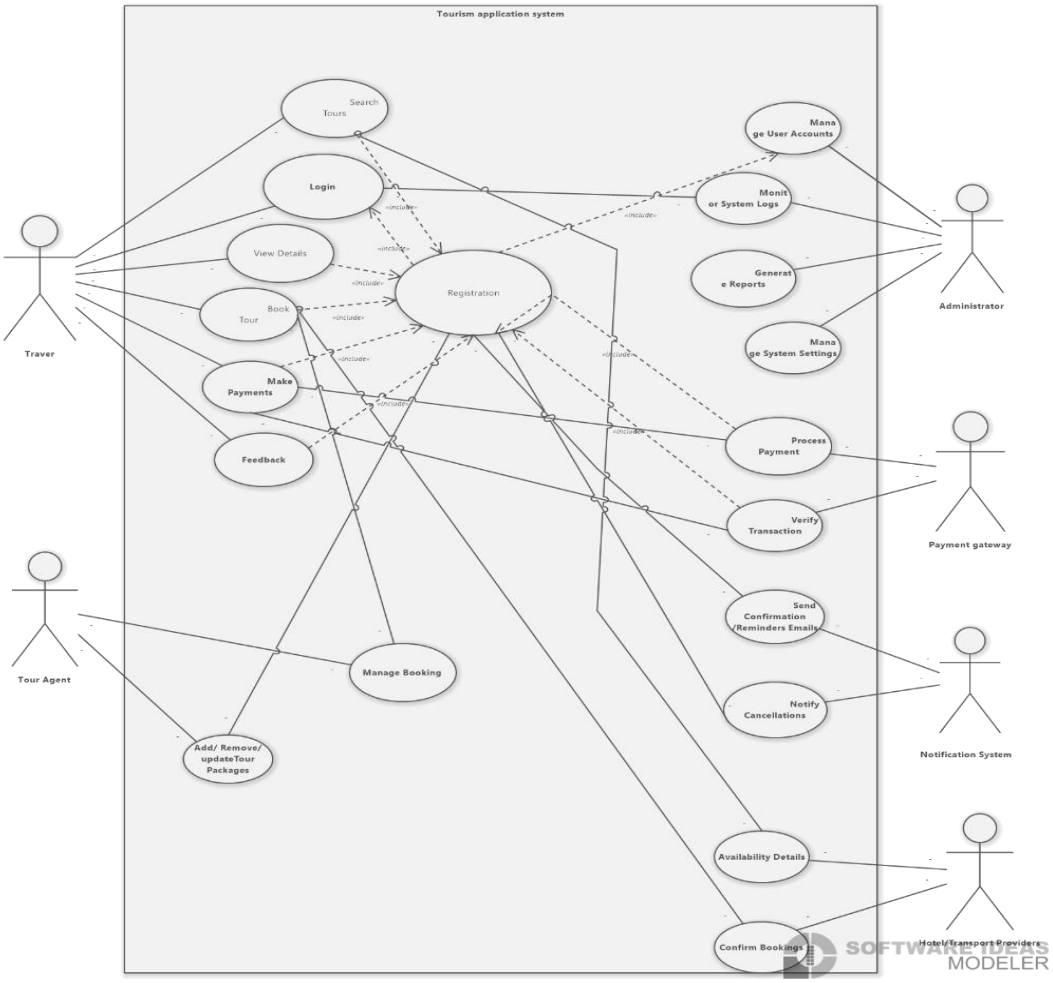


Figure 2: use case diagram

2.6 OBJECT MODEL

An object model represents a conceptual framework that captures the core components of a platform using object-oriented methods. It provides a structural blueprint for the system prior to implementation. This model visually illustrates essential entities such as users, destinations, tours, accommodations, and reviews, along with their attributes and relationships. When integrated with a design system, the object model ensures consistency in user experience and seamless functionality across the platform's various features.

2.6.1 Data Dictionary

The data dictionary is used to define each class within the system, including the attributes, operations, and a description of each class.

Class: User

- **Attributes**: User ID, User Name, Password, Email, Contact Information (Phone, Address).
- **Operations**:
  - register(): To create a new user account.
  - login(): To authenticate the user.
  - logout(): To log out of the platform.
  - updateProfile(): To edit user details.
- **Description**: Represents the traveler using the platform to explore and book tours.

Class: Traveler

- **Attributes**: Traveler ID, Name, Booking History, Feedback.
- **Operations**:
  - searchTours(): To browse tours based on filters.
  - viewTourDetails(): To view information about specific tours.
  - bookTour(): To reserve a tour package.
  - cancelBooking(): To cancel a previously booked tour.
  - provideFeedback(): To submit reviews and ratings.
- **Description**: Represents a traveler who uses the platform to search, book, and provide feedback for tours.

Class: Tour Agent

- **Attributes**: Agent ID, Name, Tour Listings, Contact Info.
- **Operations**:
    - addTourPackage(): To create a new tour listing.
    - updateTourPackage(): To modify an existing tour package.
    - removeTourPackage(): To delete a tour package.
    - viewBookings(): To see all bookings for tours they manage.
    - approveBooking(): To confirm a traveler's booking.
    - rejectBooking(): To deny a traveler's booking.
- **Description**: Manages tour packages and bookings on behalf of travel service providers.

Class: Administrator

- **Attributes**: Admin ID, Name, System Logs, User Accounts.
- **Operations**:
    - manageUserAccounts(): To add, suspend, or delete user accounts.
    - monitorSystemLogs(): To review system activities and errors.
    - generateReports(): To create reports on revenue, bookings, and user activities.
    - configureSettings(): To adjust system settings and permissions.
- **Description**: Oversees the platform’s operations, ensuring proper functionality and user management.

Class: Payment Gateway

- **Attributes**: Transaction ID, Payment Amount, Payment Status.
- **Operations**:
    - processPayment(): To handle payment transactions.
    - verifyTransaction(): To confirm the validity of payments.
- **Description**: Processes and verifies payments made for bookings.

Class: Notification System

- **Attributes**: Notification ID, Message Content, Recipient Details.
- **Operations**:
    - sendConfirmation(): To send booking confirmations.
    - sendReminder(): To alert users of upcoming tours.
    - notifyCancellation(): To inform users about tour changes or cancellations.
- **Description**: Handles all communication between the platform and its users.

Class: Tour

- **Attributes**: Tour ID, Destination, Price, Itinerary, Availability, Tour Dates.
- **Operations**:
  - getDetails(): To retrieve information about the tour.
  - updateDetails(): To modify tour information.
  - checkAvailability(): To verify if the tour has open slots.
- **Description**: Represents a travel package available for booking.

Class: Booking

- **Attributes**: Booking ID, Traveler ID, Tour ID, Booking Date, Status, Payment Info.
- **Operations**:
  - createBooking(): To reserve a tour.
  - cancelBooking(): To remove a booking.
  - updateStatus(): To change the status of a booking (e.g., confirmed, canceled).
- **Description**: Tracks reservations made by travelers.

Class: Accommodation/Transport Provider

- **Attributes**: Provider ID, Name, Availability, Pricing.
- **Operations**:
  - updateAvailability(): To modify room or transport availability.
  - confirmBooking(): To validate reservations for accommodations or transport.
- **Description**: Offers room and transport options for tours.

Class: Review/Feedback

- **Attributes**: Feedback ID, Traveler ID, Tour ID, Rating, Comment.
- **Operations**:
  - submitReview(): To provide feedback for a tour.
  - viewFeedback(): To retrieve reviews and ratings for a tour.
- **Description**: Represents user-generated feedback for tours.

2.6.2 Class diagram

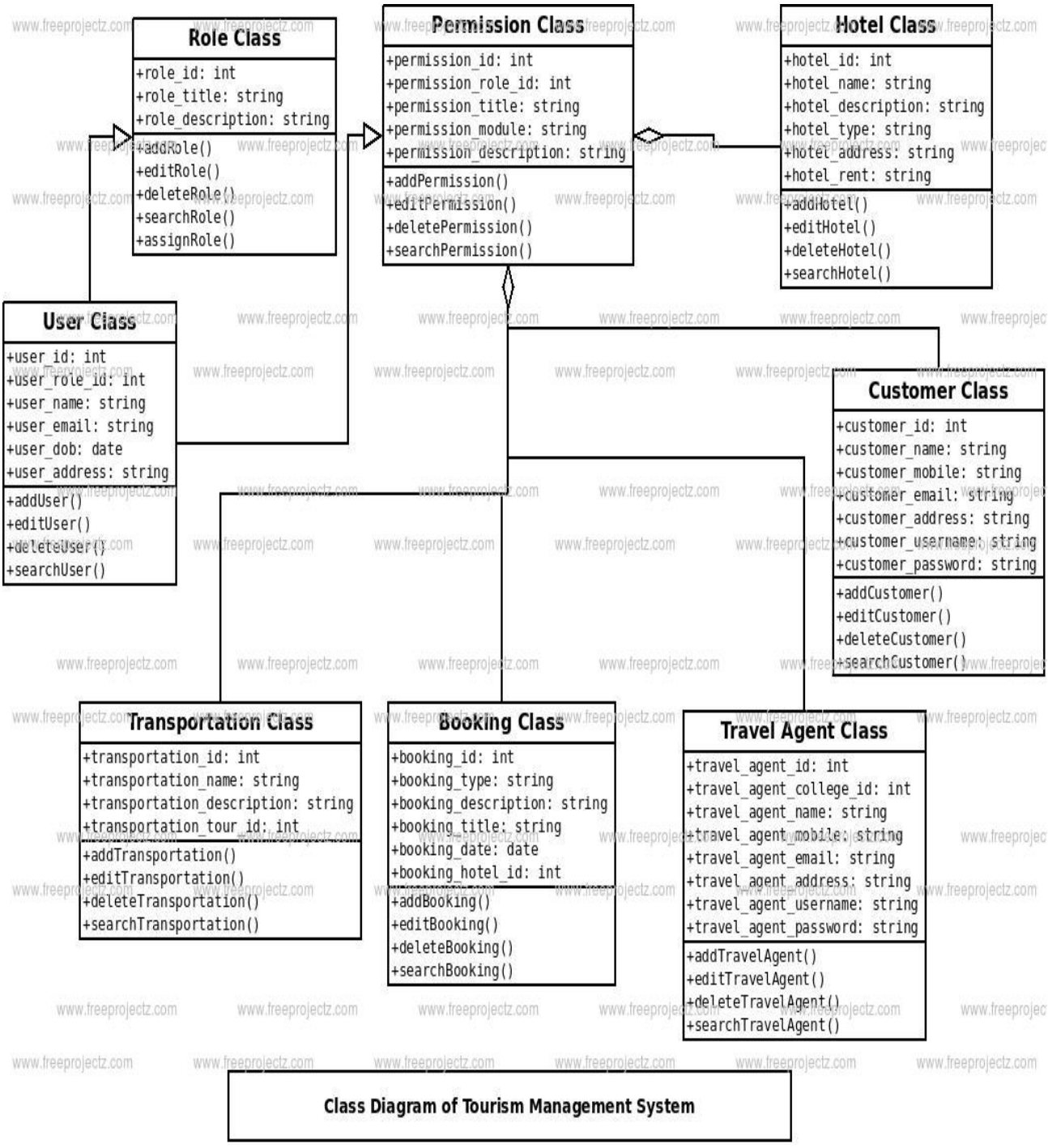


Figure 3 Class Diagram

2.6.3 Dynamic Modelling

There is some sort of interaction in every system. This can include interactions between the components of a software system, between the program being developed and other systems in its environment, or between the user and the inputs and outputs they provide. Modelling user interaction is crucial since it facilitates the identification of user requirements. System-to-system interaction modelling brings to light potential communication issues. We can determine whether a suggested system structure is likely to provide the necessary system performance and dependability by modelling the interaction between the components. This project's section on interaction modelling goes into great length about the dynamic model in particular. The way that the objects involved in each use case interact is represented by the dynamic model. It also explains the real interactions that the system has with its surroundings. The use cases and objects chosen during the object structuring of the intended system development serve as the foundation for creating the dynamic model. State charts and sequence diagrams are used to illustrate it.

2.6.4 Sequence Diagram

An example of an interaction diagram in a UML is a sequence diagram, which is used to represent the order and sequence of actions in a process or set of related processes. It is a Message Sequence Chart construct. An object's interactions are arranged chronologically in a sequence diagram. It shows the classes and objects that are a part of the scenario as well as the messages that are passed between the objects in order for the scenario to function. In this picture, the players and items that are involved are enumerated along the top, and a vertical dotted line is drawn from these. Annotated arrows show how items interact with one another. The dotted lines' rectangle denotes the lifeline of the object in question (that is, the amount of time the object instance spends in the computation). The exchanges are read from top to bottom in order. The calls to the objects, their parameters, and the return values are indicated by the annotations on the arrows. Sequence diagrams for our system situations are shown below

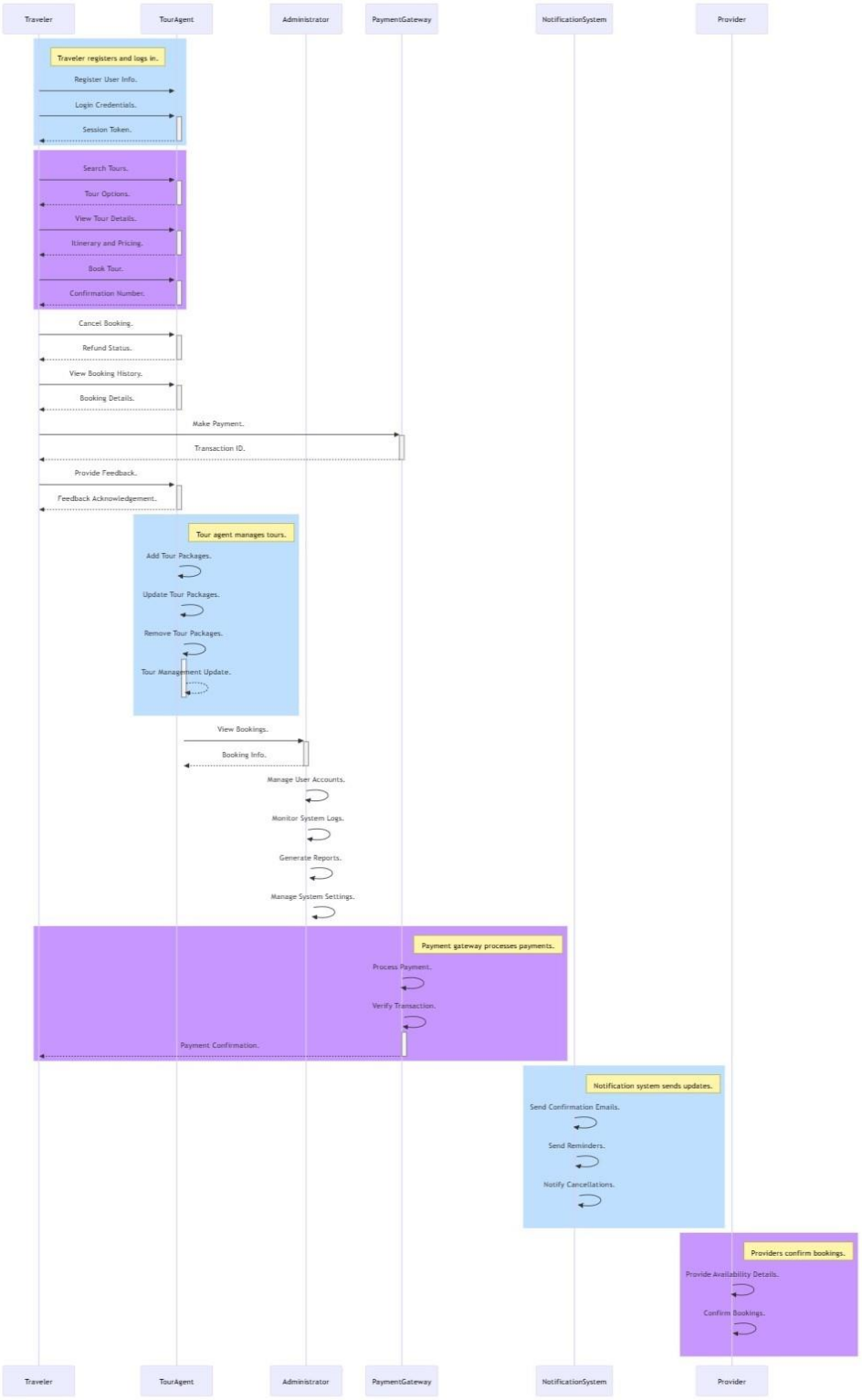


Figure 4: - sequence diagram in all case

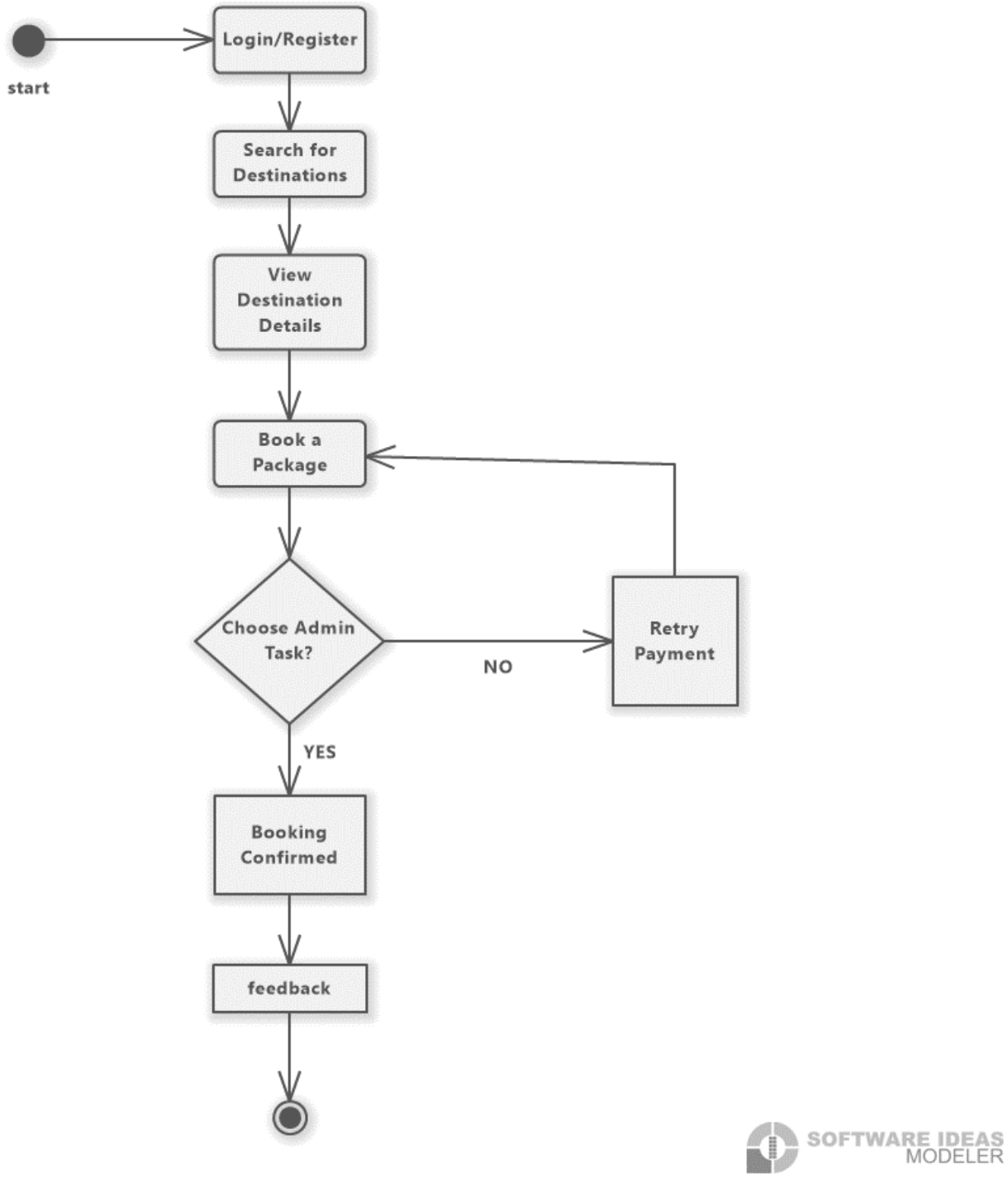
2.6.5 Activity Diagram

The graphical depiction of sequential activities and actions with provision for choice, iteration, and concurrency is called an activity diagram. Another crucial behavioral diagram in the UML diagram that describes the system's dynamic elements is this one. An activity diagram represents the flow from one activity to another and is simply a more complex form of a flowchart. This graphic helps describe behavior that involves a lot of parallel processing and illustrates the links between workflows. It is possible to select the sequence in which tasks are completed when utilizing an activity diagram. It lays down the fundamental guidelines for sequencing. It differs from a flow chart in that it displays processes in parallel as well as sequential order.

The following table illustrates Notation and Definition used in Activity diagram

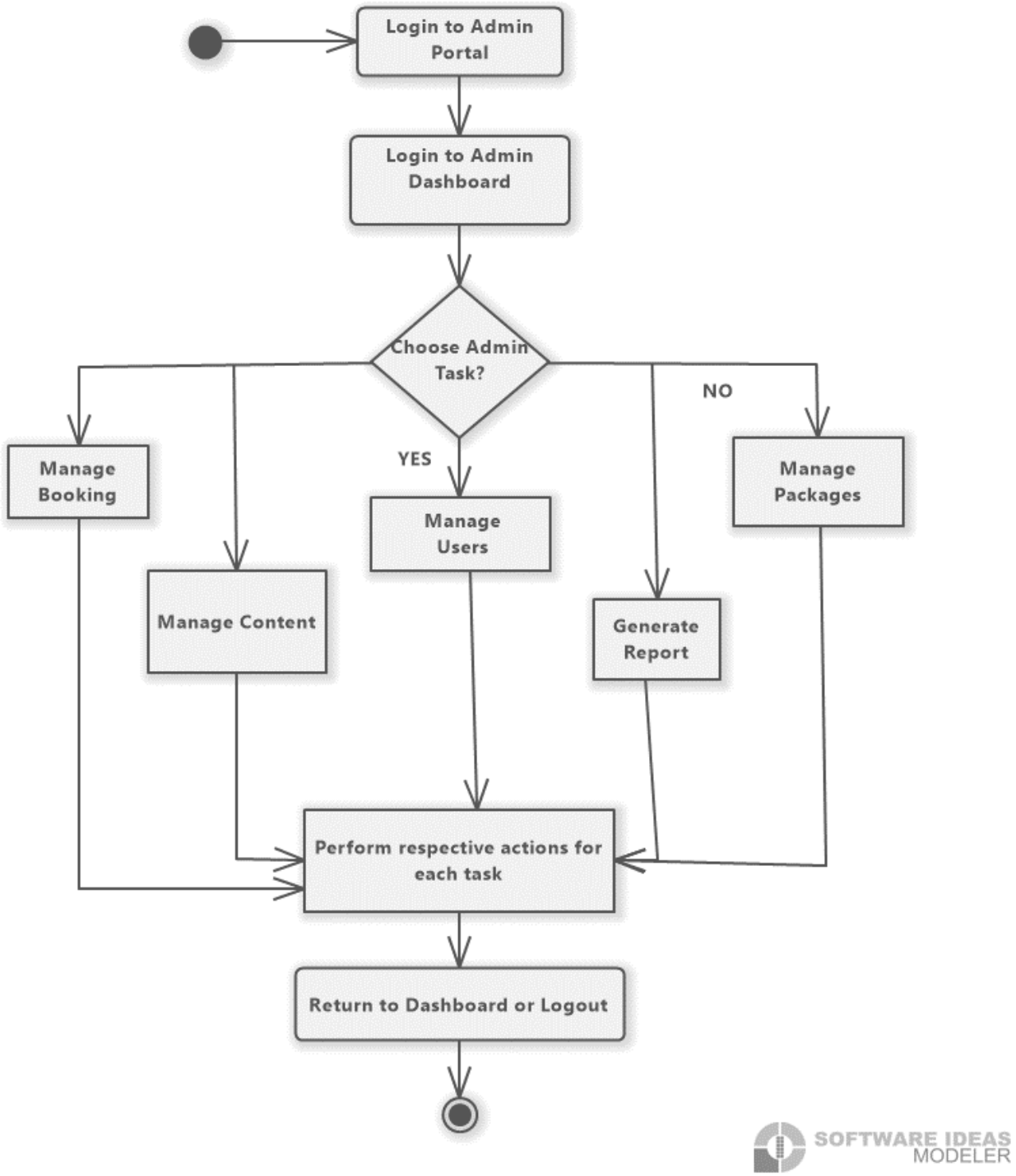
	Notation	Definition
1	State Transition	In an activity diagram the states are activities representing the performance of operations. The transitions are triggered by the completion of the operations.
2	Initial State	A pseudo state to establish the start of the event into an actual state.
3	Final State	Signifies when a state transition ends
4	Decisions	As in a flowchart, a conditional flow of control is modeled using the decision element. A decision is shown by labeling each output transition of a decision with a different guard condition

Activity Diagram for the following uses:
Describes activities and flows of data or decisions between activities Provides a very broad view of business processes
Can be used to break out the activities that occur within a use case
Commonly shows many different activities that will be handled by lots of different symbols
Good for showing parallel threads



activity diagram for our system

Fig Flow of Activities: for a Tourist:



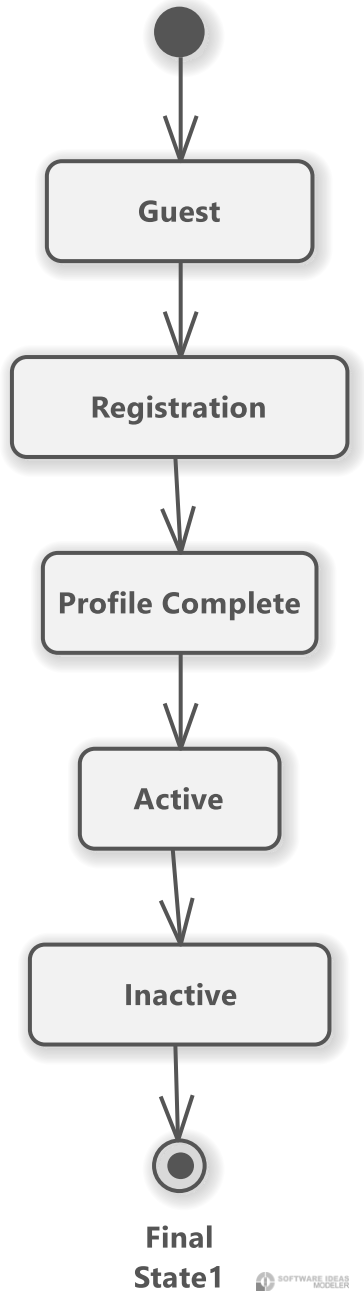
Activity diagram for Admin

2.6.6 State Chart Diagram

One type of UML diagram used to represent a system's dynamic nature is the state chart diagram. Throughout an object's existence, they specify several states, and events alter these states. They come very use when simulating reactive systems. A system that reacts to internal or external events is known as a reactive system.
The transfer of power between states is depicted in a state chart graphic. A state is an object's existing condition that modifies in response to an external event. The State Chart diagram's primary goal is to simulate an object's lifecycle from creation to termination.

System states are represented by rounded rectangles in UML state diagrams. They might also give a succinct explanation of the steps done in that stage (that comes after "do"). Stimuli that force a change in state are shown by the labelled arrows. As in activity diagrams, we can use filled circles to represent start and finish states. The following describes a few of our system's status chart diagrams.

State Chart Diagram for Booking



State Chart Diagram for User Profile

State Chart Diagram for Review and Feedback Process

CHAPTER THREE: The proposed system

3.1 SYSTEM DESIGN

Overview

The process of converting the analytical model into a system design model includes the system design component. A few years prior, we were in an area of concern. In software development, system design is the first component to enter the solution domain. The main objective of this chapter is to convert the analysis model into the design model, which considers the limitations and non-functional requirements mentioned in the previous sections on requirement analysis and issue definition. The system's overall design challenges, including the design aim, subsystem decomposition, hardware/software mapping, and persistent data management, are covered in this document. It offers a thorough architectural synopsis of the suggested system. It aims to convey and capture the important architectural choices that have been made for the system.

3.2 Purpose of the System

The system's overall design issues are covered in this document. It offers a thorough architectural synopsis of the suggested system. It aims to convey and capture the important architectural choices that have been made for the system.

Design Goal

The goal of design is to create a high-quality representation of the system. The non-functional requirements, which describe the features, traits, and other aspects of the system along with any limitations that can set boundaries for the suggested solution, are the source of the design goals.

Design goals describe the qualities of the system that the developers should consider.

- Performance
- Dependability
- Maintenance

Performance:

In order for the tourism management application to give the user a good service it should meet the following performance criteria.

Response time: - depending on the strength of available network the system should be responded in a short period of time.

Memory: - to do work efficiently the processor has more than 1GB RAM

Dependability

Our system includes the following dependability criteria:-

- Reliability: the system should be reliable. The system shouldn’t crash while in use

- Security: the platform will be secure, i.e., not allow other users or unauthorized users to access data that has no the right to access it.
- Availability: as long as there is an internet connection the system will be available 24 hours a day.

Maintenance

To be maintainable the system should meet the following maintenance criteria: -

Modifiablity: the system will be modifiable for further modification and enhancement of the system. Example to incorporate more complicated content management systems.

Portability: The system is developed to be viewed and retrieved from any web browser regardless of their version and platform it resides in.

Extensibility: If it is needed to add new functionality to the system, this must be achieved by only making a separate page and integrating this page with the existing system

End user

Forms and buttons with descriptive titles should have an easy-to-understand graphical user interface (GUI) in the system. In order to make the interfaces, forms, and buttons easily readable by users, they are always written or created in common or simple languages. The system ought to fulfil the following end user requirements, as seen from the user's perspective.

Utility: In order to help the user to easily understand and interact with the system, the system language must be easily understandable to express the function that activity is giving.

Usability: For the system to be more interactive in order to promote its usability. The following usability principle has to be considered.

Learnability: The ease with which new users can begin effective interaction and achieve Clio maximal performance. Any new user can use the system without any effect. The system has no ambiguity and has clear and easy interfaces.

Flexibility: The multiplicity of ways the user and system exchange information. Supports the user to get the services that are stored in the dataset by using search.

3.3 Purpose of the System

3.3.1 Proposed System Architecture for Tourism Management System

The **Proposed System Architecture** of a **Tourism Management System** can be represented in a multi-tiered architecture model. A standard architecture consists of **three main layers**: **Presentation Layer, Application (Business Logic) Layer and Data Layer**

1. Presentation Layer

The **Presentation Layer** is the interface through which users interact with the system. It includes:

Key Features:

- Provides the **front-end** of the system to different types of users (e.g., tourists, admins, travel agents).
- Ensures responsive and user-friendly interfaces for various devices (desktop, tablet, mobile).
- Facilitates **input validation** and **user interaction**.

2. Application (Business Logic) Layer

The **Application Layer** processes the core business logic of the Tourism Management System. It acts as an intermediary between the Presentation Layer and the Data Layer.

Key Features:

- Processes user requests (e.g., booking tours, creating packages).
- Implements business rules and logic, such as:
    - **Booking Management**
    - **Tour Package Creation and Updates**
    - **Payment Processing**
    - **User Authentication and Authorization**
- Handles communication with the Data Layer through **APIs**.
- Supports **error handling**, **logging**, and **performance optimization**.

3. Data Layer

The **Data Layer** manages the storage, retrieval, and maintenance of data in the system. It ensures data persistence, integrity, and security.

Key Features:

- Stores data related to users, destinations, tour packages, bookings, and payments.
- Handles database queries, transactions, and optimizations.
- Ensures **data consistency** and **security**.
- Supports backup, recovery, and scalability.

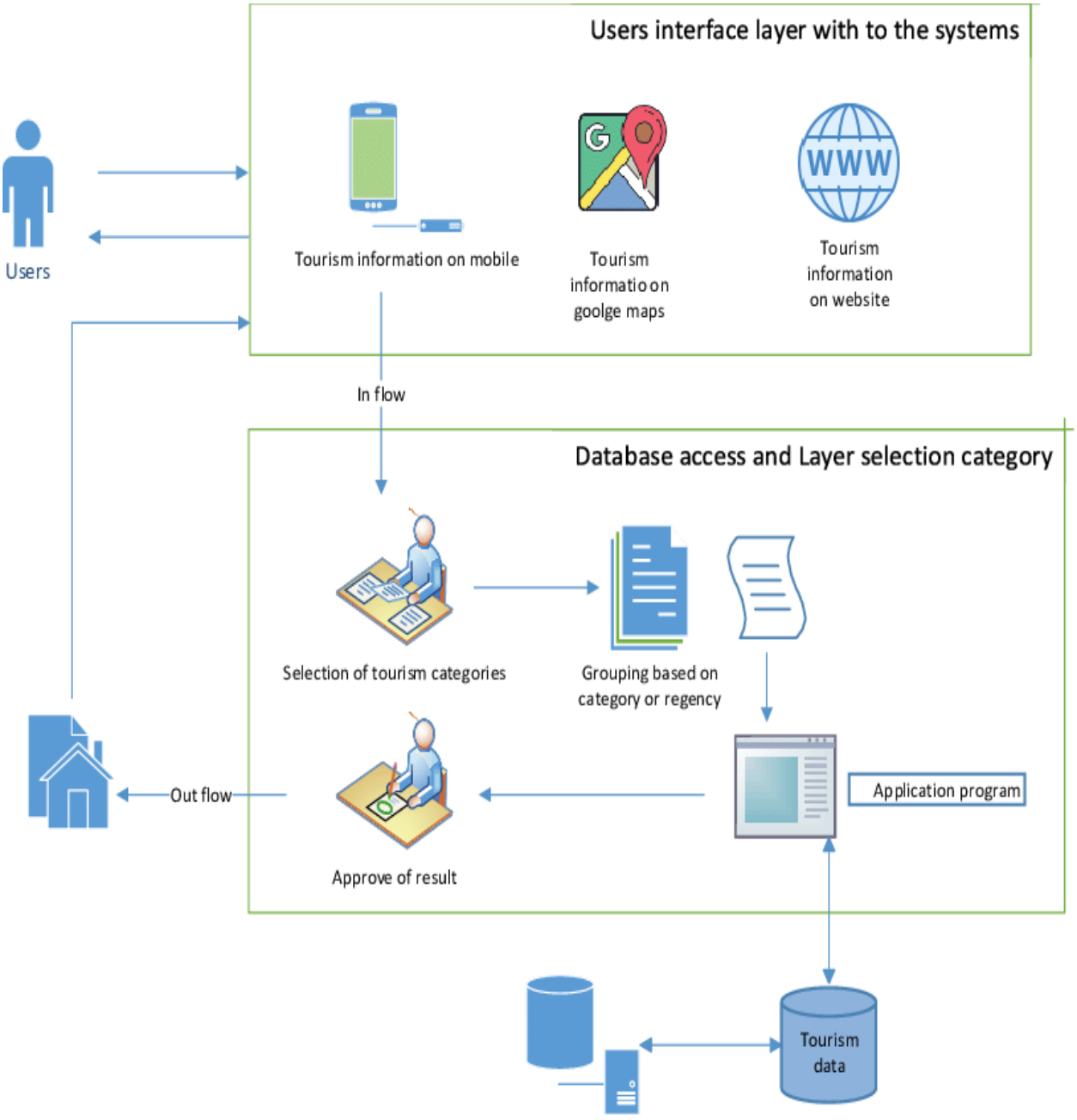


Figure: Client and Server Relationship Diagram

The diagram represents the Proposed System Architecture for a Tourism Management System, consisting of two main layers: the **User Interface Layer** and the **Database Access and Layer Selection Category**. In the **User Interface Layer**, users interact with the system through multiple platforms such as mobile devices for tourism information, Google Maps for location-based services, and websites for accessing tourism data, where user inputs (e.g., queries or category selections) flow into the system. In the **Database Access**

and Layer Selection Category, the system processes user requests by selecting and categorizing tourism data based on preferences like grouping by categories or regions. The **Application Program** interacts with the **Tourism Data** stored in the database to retrieve and process the results, which are then approved and sent back to the user as output. The flow of data can be summarized as **Users → Input → Processed through the database → Output back to users**, ensuring smooth access to categorized tourism data for an enhanced user experience.

Collaboration Diagram

A **collaboration diagram** for the **Tourism Management System** is essential for understanding how the system's components interact to fulfill user requests. It helps in visualizing the dynamic flow of data and processes between different system modules. The diagram shows the relationships between various entities such as the **User**, **System Interface**, **Destination Module**, **Package Module**, **Booking Module**, **Payment Module**, and the **Database**.

When a **User** (tourist) interacts with the system via the **System Interface** (mobile, website, or Google Maps), they can search for destinations, explore packages, and make bookings. The **System Interface** serves as the intermediary that processes user input and communicates with relevant modules such as the **Destination Module**, which provides detailed information about tourist spots, and the **Package Module**, which offers details on available tours for specific destinations.

Once the user selects a tour, the **Booking Module** records the booking details such as the number of people, booking dates, and the total cost. The **Payment Module** then processes the payment through an external gateway (e.g., PayPal, Stripe). Throughout this process, all user, destination, package, booking, and payment data are stored and retrieved from the **Database**. This ensures the system maintains a consistent state and that all necessary data is easily accessible when required.

The collaboration diagram is crucial in identifying the communication flow between these modules and ensuring that each component works together smoothly to provide a seamless experience for the user. It highlights how different objects or modules cooperate to complete tasks, providing a clear, structured representation of the system's dynamic behavior.

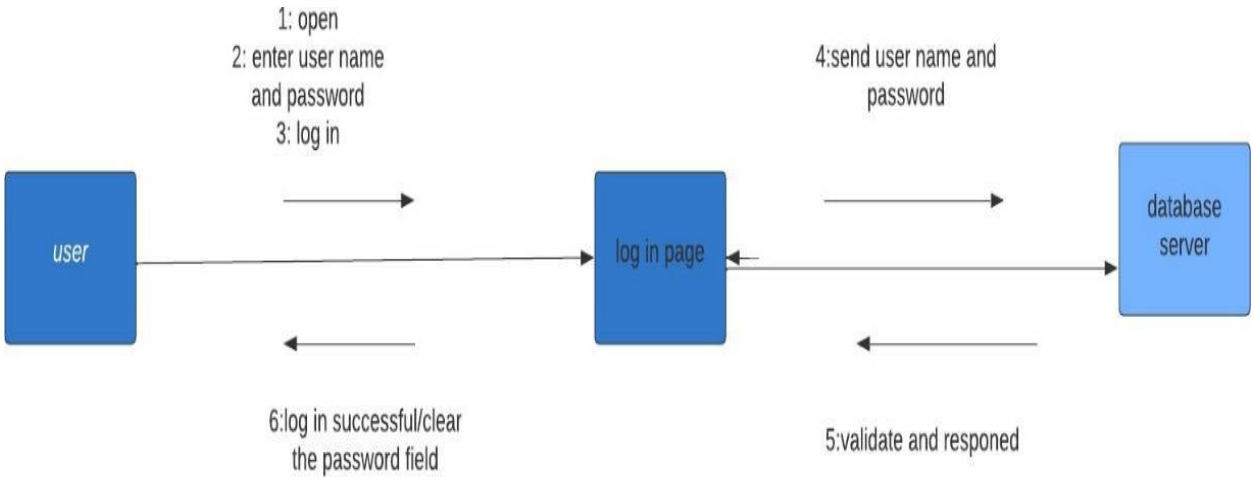


Figure: Collaboration diagram for log in

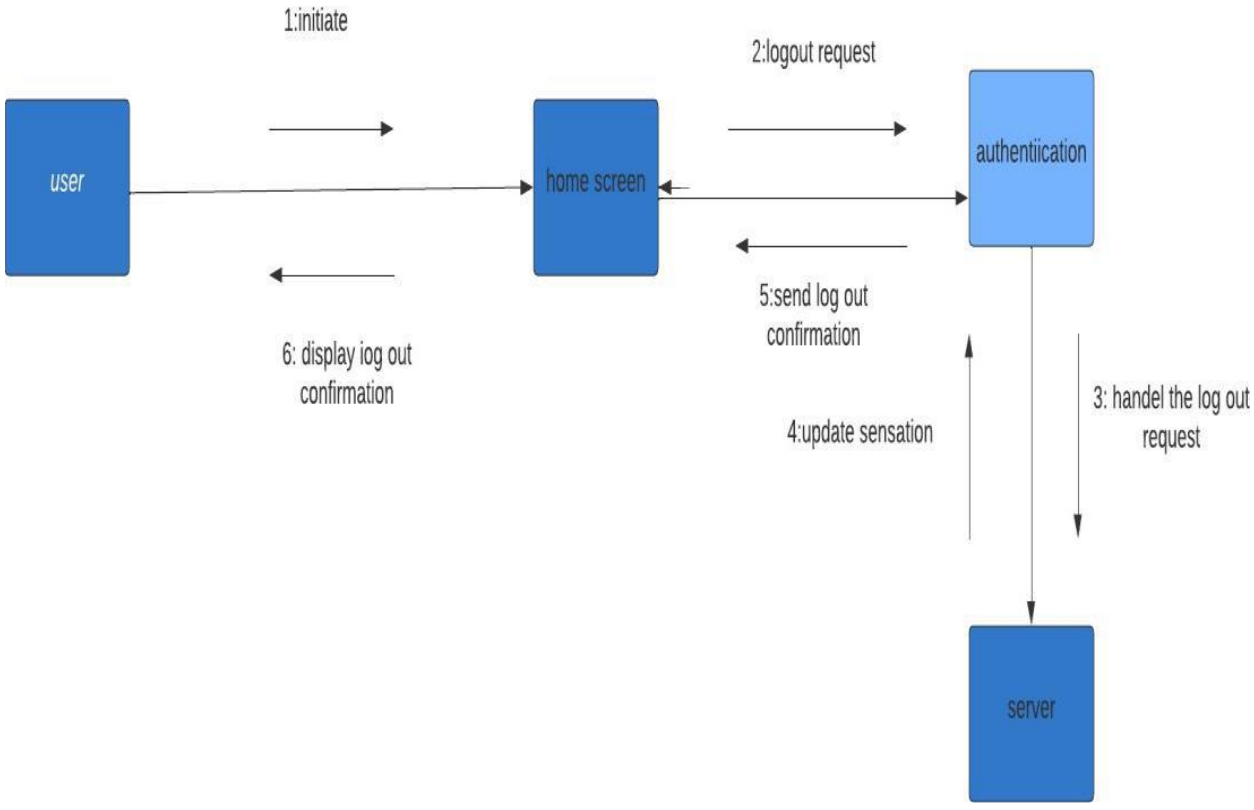


Figure: Collaboration diagram for log out

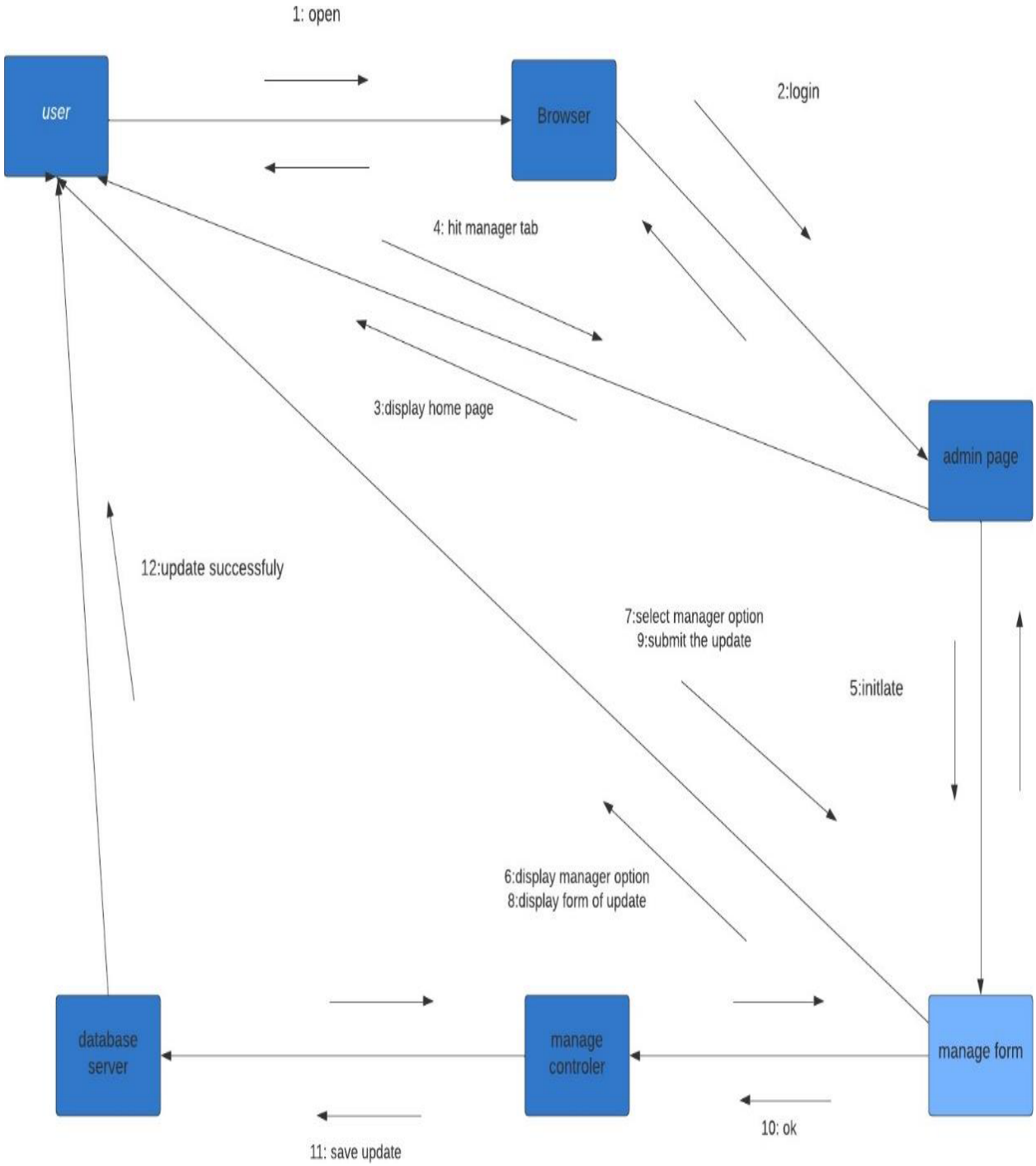


Figure: Collaboration diagram for update profile

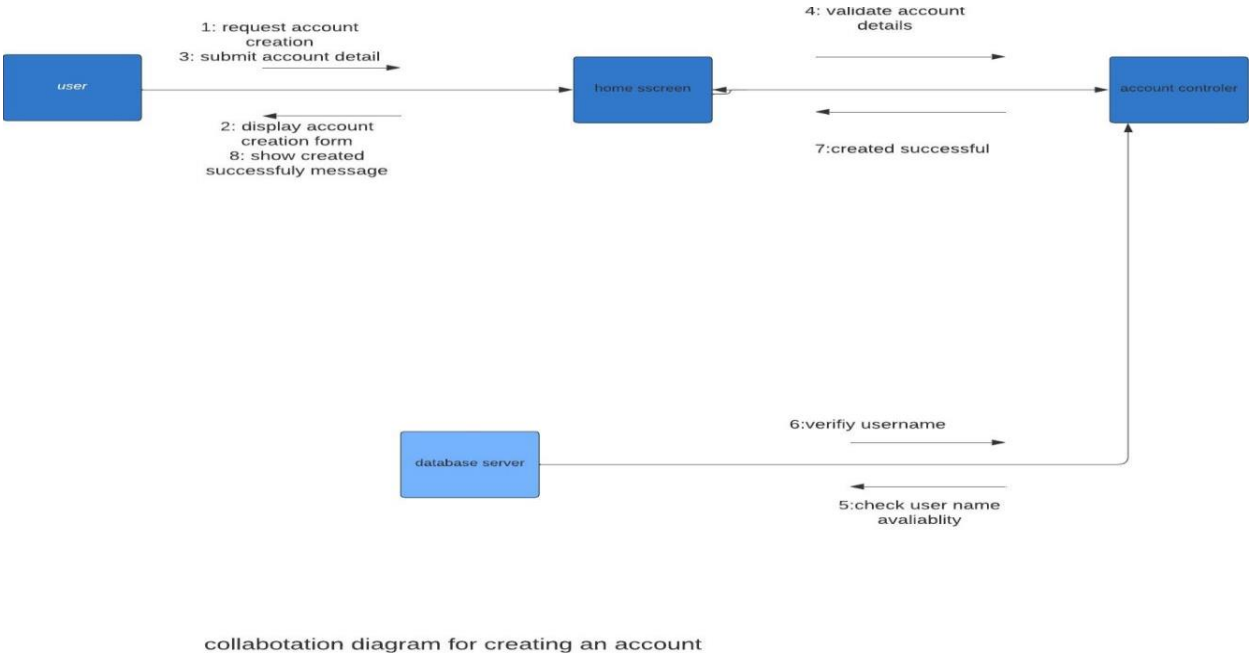


Figure 18: Collaboration diagram for creating account

3.3.2 Request process

This section explores how the user and request processor interact using architecture design diagram.

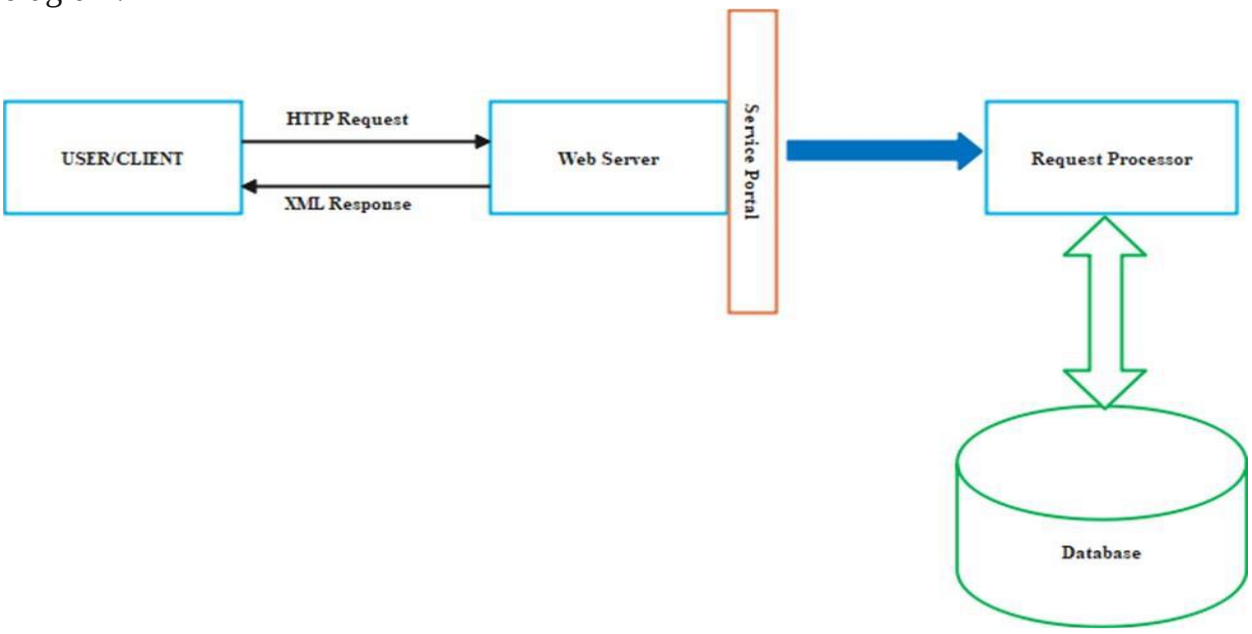


Figure 23: Request Processor

3.3.3 Subsystem Decomposition

To reduce the complexity of the solution domain, we decompose a system into simpler parts, called subsystems.

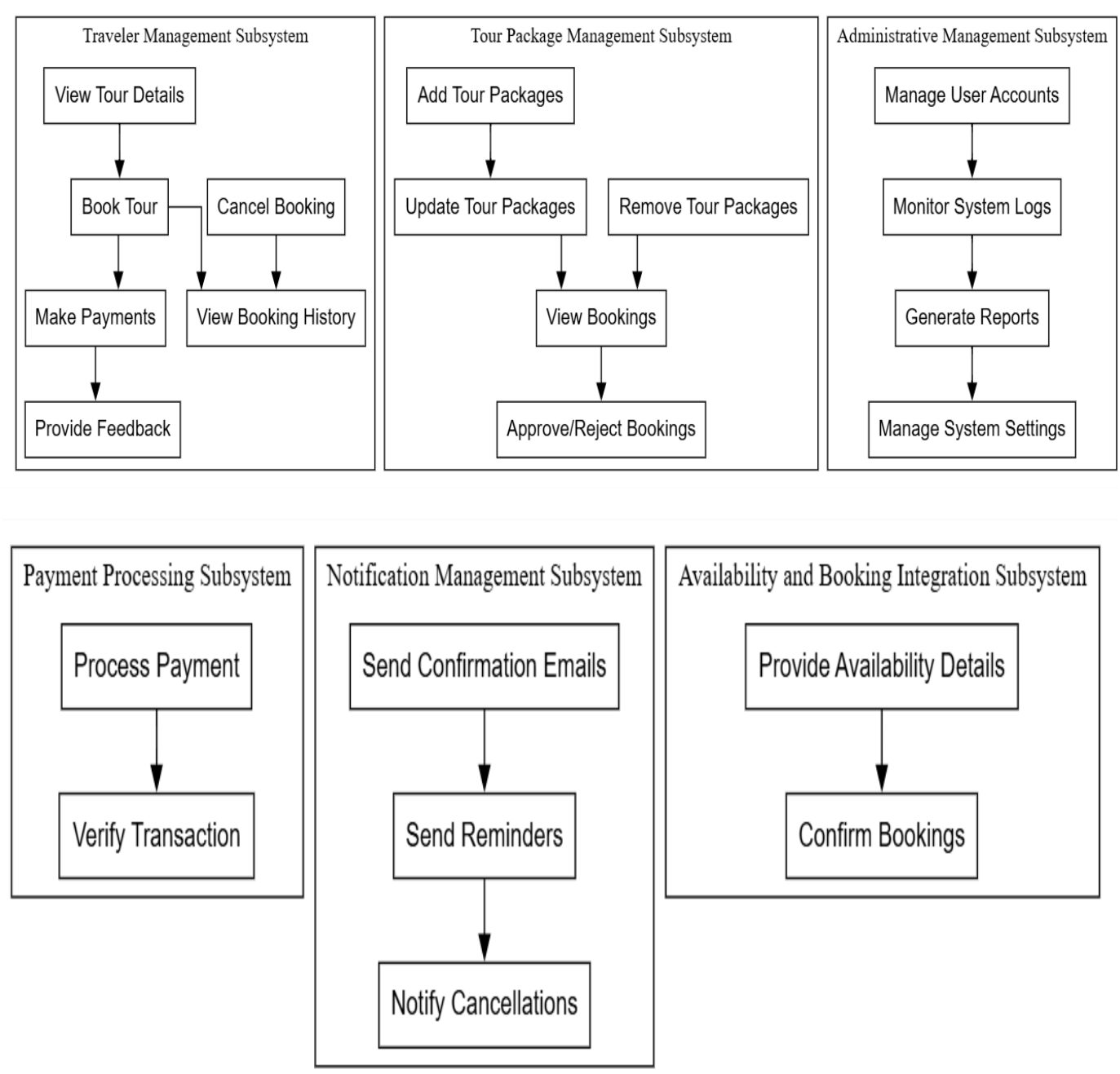


Figure 24: Subsystem diagram

3.3.4 Hardware/Software Mapping

The Hardware/Software Mapping outlines the interaction between the system's hardware components and its software modules. The system comprises three main hardware components: the client devices, the server infrastructure, and the database systems. These components are interconnected through a network to ensure seamless functionality.

- **Client Side**: User devices (e.g., desktops, laptops, smartphones) are equipped with web browsers or mobile applications to interact with the system. Software components such as the front-end user interface and APIs are deployed on these devices.
- **Server Side**: The server hosts the application logic, web server, and business logic processing. It manages requests from the client and performs data processing and operations based on user input. Technologies such as web servers (e.g., Apache, Nginx) and application servers handle this layer.
- **Database**: The database management system (DBMS) stores persistent data, including user profiles, tour details, bookings, feedback, and transactional records. It interacts with the server for data retrieval and updates. Relational databases (e.g., MySQL, PostgreSQL) or NoSQL alternatives (e.g., MongoDB) may be utilized.

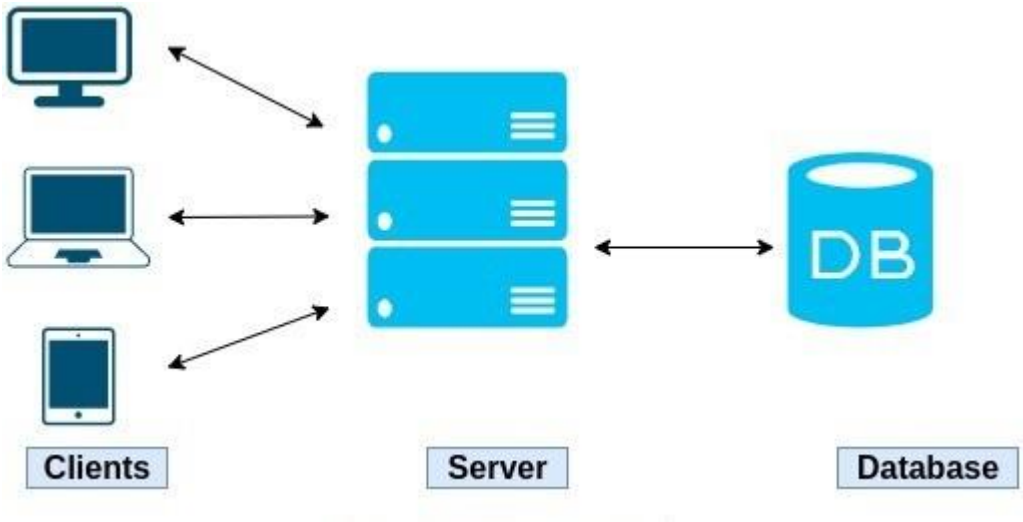
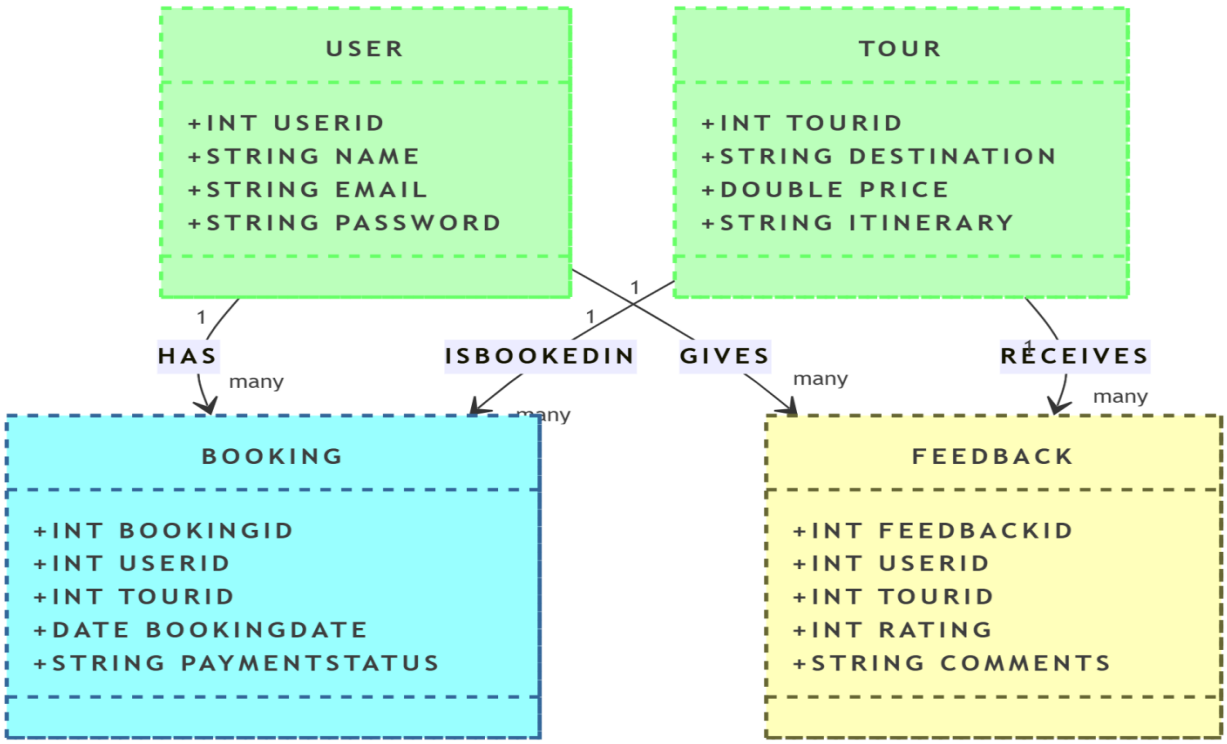


Figure 4: client, server diagram

Persistent Data Management

This section maps the objects and classes identified during system analysis to their respective relational database entities. For example:

- **User Class**: Mapped to a "Users" table with fields such as UserID, Name, Email, and Password.
- **Tour Class**: Mapped to a "Tours" table with fields such as TourID, Destination, Price, and Itinerary.
- **Booking Class**: Mapped to a "Bookings" table with fields such as BookingID, UserID, TourID, BookingDate, and PaymentStatus.
- **Feedback Class**: Mapped to a "Feedback" table with fields such as FeedbackID, UserID, TourID, Rating, and Comments.



The hardware is configured to support software deployment as follows:

- **Clients**: Access the system via a web interface or mobile app.
- **Servers**: Host the middleware and perform logic processing.
- **Database Servers**: Store and manage system data, ensuring availability and scalability.

Network connections between these components ensure efficient communication, enabling real-time data processing and user interaction across the system.

3.3.5 Deployment Diagram

The deployment diagram visually represents how the software components of the system are deployed on hardware components. This type of structural diagram illustrates the relationships between the system's hardware and software, highlighting physical deployment.

Deployment Diagram Description for the System

- **Nodes**: The deployment diagram consists of nodes (depicted as three-dimensional boxes), which represent the main hardware or software components of the system. These nodes include client devices, servers, and databases.
- **Relationships**: Connections between nodes are represented by lines, showing how the components communicate and transfer data.
- **Software Artifacts**: Smaller shapes within the nodes represent the deployed software components, such as front-end applications, APIs, or database systems.

System Nodes

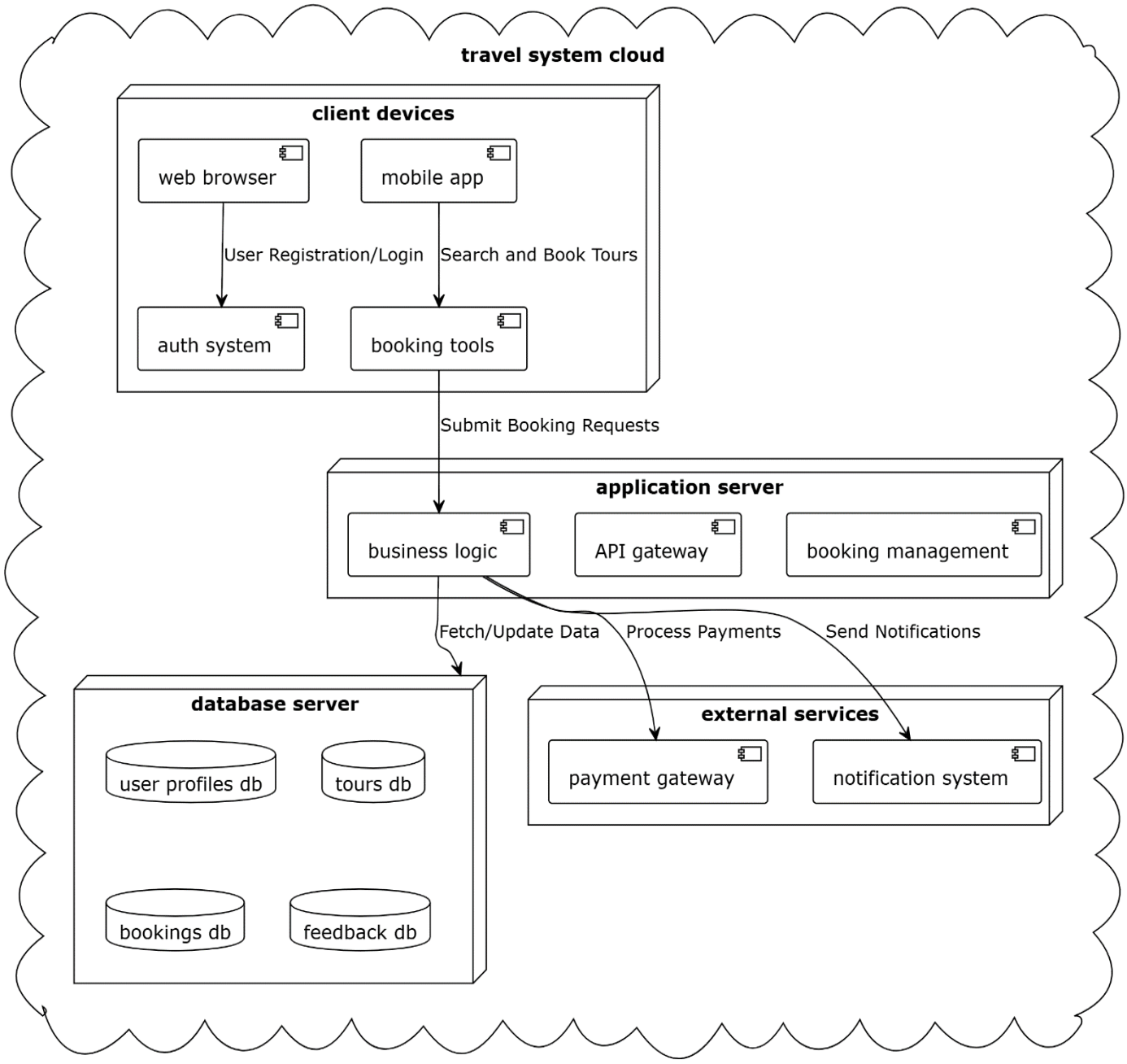
1. **Client Devices** (Traveler/Tour Agent/Administrator):
    - **Software Deployed**:
        - Web browser/mobile app (UI and APIs).
        - Authentication system for user registration and login.
        - Tools for search, bookings, payments, and feedback submission.
    - **Functionality**:
        - Travelers interact with the system to search and book tours.
        - Tour agents manage tour packages.
        - Administrators oversee user accounts and system activities.
2. **Application Server**:
    - **Software Deployed**:
        - Business logic handling for booking, payments, and user feedback.
        - APIs for client-server interaction.
        - Booking and approval logic for agents and administrators.

- **Functionality**:
  - Processes user requests and connects the client to the database.
  - Manages tour information and booking workflows.

3. **Database Server**:
   - **Software Deployed**:
     - Relational or NoSQL database systems (e.g., MySQL).
     - Persistent storage for user profiles, tours, bookings, and feedback.
   - **Functionality**:
     - Stores and retrieves system data such as tour details, user information, and booking records.
4. **Payment Gateway** (External Node):
   - **Software Deployed**:
     - Secure transaction system for payment processing.
   - **Functionality**:
     - Processes and verifies payments for tour bookings.
5. **Notification System** (External Node):
   - **Software Deployed**:
     - Email/SMS service for confirmations and reminders.
   - **Functionality**:
     - Sends notifications about bookings, cancellations, and reminders

Key Interactions

- **Client ↔ Server**: Secure communication for submitting user requests, booking tours, and providing feedback.
- **Server ↔ Database**: Server fetches or updates persistent data from the database.
- **Server ↔ Payment Gateway**: Processes payment transactions and verifies them.
- **Server ↔ Notification System**: Triggers and sends notifications for confirmations, reminders, or cancellations.



This deployment architecture ensures efficient data processing, secure transactions, and reliable system communication, making it scalable and user-friendly for travelers, tour agents, and administrators.

3.4 Access Control Access control

which establishes the conditions under which specific data, applications, and resources can be accessed, is a crucial component of security. Stated differently, they admit the right people while excluding the incorrect ones. Techniques like authorization and authentication are critical to the operation of access control rules. Various players in multiuser systems have varying levels of access to data and functionality. We modelled these differences by assigning various players to specific use cases. The access table that follows provides details about this.

Access Control for Traveler

Object	**Actors**	**Use Case**
User Registration	Traveler	Register as a new user
User Login	Traveler	Login with credentials (email, password)
Search Tours	Traveler	Search for tours by destination, price, etc.
View Tour Details	Traveler	View selected tour details
Book Tour	Traveler	Book a selected tour
Cancel Booking	Traveler	Cancel a previous booking
View Booking History	Traveler	View past and upcoming bookings
Make Payments	Traveler	Process payment for a tour booking
Provide Feedback	Traveler	Submit feedback or review after a tour

Access Control for Tour Agent

Object	**Actors**	**Use Case**
Add Tour Packages	Tour Agent	Add a new tour package
Update Tour Packages	Tour Agent	Update an existing tour package
Remove Tour Packages	Tour Agent	Remove an unavailable tour package
View Bookings	Tour Agent	View and manage bookings for their tours
Approve/Reject Bookings	Tour Agent	Approve or reject tour bookings

Access Control for Administrator

Object	Actors	Use Case
Manage User Accounts	Administrator	Manage user accounts (approval, suspension, deletion)
Monitor System Logs	Administrator	Monitor system activities
Generate Reports	Administrator	Generate reports for bookings, activities, revenue
Manage System Settings	Administrator	Configure system settings (roles, payments, etc.)

Access Control for Payment Gateway

Object	Actors	Use Case
Process Payment	Payment Gateway	Process payment for booking
Verify Transaction	Payment Gateway	Verify and confirm the payment

Access Control for Email/SMS Notification System

Object	Actors	Use Case
Send Confirmation Emails	Notification System	Send confirmation emails for bookings, cancellations
Send Reminders	Notification System	Send reminders for upcoming bookings and tours
Notify Cancellations	Notification System	Notify about tour cancellations or changes

Access Control for Hotel/Transport Providers

Object	Actors	Use Case
Provide Availability Details	Providers	Update availability for rooms/transportation
Confirm Bookings	Providers	Confirm bookings for accommodations or transportation

This table reflects the various user roles and their corresponding permissions to access specific actions within the system

Acknowledgement

We sincerely appreciate Dr. Sintayewu Hirphasa for his outstanding contributions to this project. We are deeply grateful for his valuable knowledge and insightful guidance, which played a significant role in helping us achieve our objectives. His expertise and experience, particularly in the field of Computer Science and Engineering, were truly beneficial throughout this process. His support and encouragement were instrumental in the successful completion of this project.