Software Requirements Specifications

Backend System for Travel Agency Application

2024/2025

#	Name	ID	Group
1	Mahmoud Hisham Fawzy	20226098	S1
2	Khaled Ahmed Kamal	20226037	S1
3	Abdelrahman Maged	20226058	S1
4	Salman Waleed Farouk	20226046	S1
5	Eyad Mazen Ibrahim	20226022	S1

TA: Hassan Mourad

Glossary for Abbreviations

1. **CRUD - Create, Read, Update, delete**
   The four basic functions for managing data in a database or application.
2. **JWT - JSON Web Token**
   A compact, URL-safe means of securely transmitting information between parties, often used for user authentication.
3. **REST - Representational State Transfer**
   An architectural style for designing networked applications, often used in web APIs to handle client-server interactions.
4. **SSL/TLS - Secure Sockets Layer / Transport Layer Security**
   Cryptographic protocols for securing communications over a computer network, ensuring safe data transmission.
5. **PCI-DSS - Payment Card Industry Data Security Standard**
   A set of security standards designed to ensure that all companies that process, store, or transmit credit card information maintain a secure environment.
6. **SLA - Service Level Agreement**
   A formal agreement between service providers and customers specifying the expected level of service.
7. **DNS - Domain Name System**
   A system that translates domain names into IP addresses, enabling communication between devices on the internet.
8. **CI/CD - Continuous Integration / Continuous Deployment**
   Practices that automate the integration and deployment of code, ensuring frequent updates to the software without disruptions.
9. **OAuth2 - Open Authorization 2.0**
   A protocol for secure, delegated access, allowing third-party services to access user data without sharing passwords.
10. **KPI - Key Performance Indicator**
    A measurable value that indicates how effectively a company is achieving key business objectives.
11. **RPA - Robotic Process Automation**
    The use of software robots to automate repetitive tasks, often used for system integration or data entry.
12. **TFA - Two-Factor Authentication**
    A security process that requires users to provide two forms of identification before gaining access to an account.
13. **GDPR - General Data Protection Regulation**
    A regulation in EU law on data protection and privacy in the European Union and the European Economic Area. It gives individuals control over their personal data and imposes strict rules on data collection, processing, and storage. It includes rights like access, correction, deletion (right to be forgotten), and strict penalties for non-compliance.

1. Introduction

1.1 Purpose

The purpose of developing the **Travel Agency Backend System** is to create a robust, scalable, and secure platform that enables the backend operations of a travel agency, supporting key business functions like hotel bookings, event ticketing, payment processing, and user notifications. This system will act as the core engine for the travel agency, processing user requests and enabling seamless interactions with external APIs, payment gateways, and hotel providers.

The backend system is designed using a **Layered Architecture** and follows a **Microservices Architecture** to ensure modularity, scalability, and fault tolerance. The primary purpose of the system is to ensure a seamless user experience while booking hotel rooms, purchasing event tickets, and receiving real-time notifications via email/SMS. The **Notification Service** will be key in ensuring users are promptly informed about their bookings and upcoming events, improving engagement and customer satisfaction.

The system will also be designed with **cloud deployment** in mind, ensuring scalability and high availability during periods of high traffic, such as during holidays or special events. By implementing well-defined **APIs** and leveraging third-party external APIs (e.g., hotel availability, event ticketing, payment gateways), the system will integrate with external services to provide real-time data and transactions. This will enable the travel agency to offer accurate, up-to-date information to users, ensuring that they can make informed decisions when booking hotels or tickets.

1.2 Target Audience

The **Travel Agency Backend System** is aimed at a variety of stakeholders who are directly or indirectly involved with the system. The primary audience includes:

- **End Users** (Customers of the travel agency): These users will interact with the system to book hotel rooms, purchase event tickets, manage their profiles, and receive notifications. Their key concerns are ease of use, responsiveness, security, and the availability of personalized recommendations based on their preferences and bookings.

User challenges:

  - Difficulty in finding real-time availability and booking hotels or tickets quickly.
  - Frustrating and slow notification systems (e.g., confirmation emails, reminders).
  - Lack of personalized event recommendations based on hotel bookings.
  - Concerns about the security of personal and payment information.

- **System Operators**: Responsible for managing and maintaining the backend system, ensuring it operates smoothly, and monitoring its performance. They require a reliable and fault-tolerant system that can scale to meet varying levels of demand, particularly during peak times.

Operator challenges:

  - Handling system downtime or performance issues under load.
  - Ensuring security of user data (e.g., payment details, personal information).
  - Managing system integrations and handling failures in external APIs (e.g., hotel or payment gateways).

- **Developers**: The development team is responsible for building and maintaining the system. Their concerns are related to ensuring the system is modular, maintainable, and scalable while adhering to best practices in software engineering, such as **SOLID principles** and **object-oriented programming**.

Developer challenges:

  - Designing a system that is easy to scale and maintain over time.
  - Integrating with external third-party APIs (e.g., hotel availability APIs, event APIs, payment gateways) and managing dependencies effectively.
  - Ensuring security standards and compliance (e.g., PCI-DSS, GDPR).

- **Project Managers**: Oversee the development of the system, ensuring it is completed on time, within budget, and aligned with business objectives.

Project Manager challenges:

  - Managing resources and coordinating between different teams (e.g., development, testing, operations).
  - Ensuring that the system meets business requirements and delivers value to the company.
  - Tracking progress and ensuring the project remains on schedule.

- **Business Owners**: Concerned with the overall value the system brings to the travel agency, including **return on investment (ROI)**, user engagement, and business growth. They are focused on achieving business goals and ensuring that the backend system aligns with the agency's broader objectives.

Business Owner challenges:

  - Achieving scalability to handle increasing user demand.
  - Aligning technical decisions with business goals and ensuring system flexibility for future expansions.
  - Maximizing user satisfaction and engagement through features like personalized recommendations and timely notifications.

1.3 Product Value

The **Travel Agency Backend System** provides significant value to both the end-users and the business by offering a highly functional and secure platform for booking hotels, purchasing tickets, and handling payments. The product's value proposition is defined by the following key benefits:

1. **Seamless Booking Experience**:
   Users can easily search for and book hotel rooms, view available options, and make secure payments—all in one place. Integration with external APIs ensures real-time hotel availability, accurate pricing, and up-to-date event information. This increases user trust and satisfaction.

2. **Personalized Recommendations**:
   By leveraging user data (e.g., past bookings, preferences), the system can suggest personalized events or hotels, enhancing the user experience and encouraging more bookings. This helps the travel agency increase customer retention and loyalty.

3. **Real-Time Notifications**:
   The **Notification Service** provides timely and relevant notifications to users, such as booking confirmations, reminders, and recommendations for nearby events. This service increases engagement and ensures that users remain informed at every step of their journey, leading to higher satisfaction rates.

4. **Scalability and High Availability**:
   The system is designed to handle varying traffic loads, ensuring that performance remains optimal even during high-demand periods, such as holidays or special events. The use of **Microservices Architecture** and cloud deployment (e.g., AWS) allows for independent scaling of services like booking and payment processing to meet fluctuating demand.

5. **Security and Compliance**:
   The backend system ensures secure handling of sensitive user data (e.g., personal details, payment information) and complies with industry standards such as **PCI-DSS** for payment processing and **GDPR** for data privacy. This builds user trust and ensures that the travel agency meets regulatory requirements.

6. **Operational Efficiency**:
   By automating key operations, such as payment processing and booking confirmations, and integrating with external APIs for hotel and event data, the system reduces the manual workload on operators and developers, allowing them to focus on higher-value tasks. This improves overall operational efficiency and reduces the risk of errors.

7. **Flexibility for Future Growth**:
   The system's modular design, supported by **Microservices Architecture**, allows for easy expansion. New features or external integrations can be added with minimal disruption to the existing services, ensuring that the system can grow alongside the business's needs.

1.4 Scope

The **scope of the Travel Agency Backend System** is to provide a reliable, secure, and scalable platform for managing hotel bookings, event ticket purchases, payment processing, and user notifications. This system will serve as the backend infrastructure for a travel agency located in Cairo, interacting with external APIs for hotel availability, event listings, and payment gateways.

Key deliverables:

1. **Hotel Booking Service**:
   This service will manage the hotel booking process, allowing users to search for hotels, view availability, make reservations, and manage bookings through a user-friendly interface. The service will integrate with external hotel APIs to provide real-time availability and pricing.

2. **Event Ticketing Service**:
   Users will be able to browse local events, purchase tickets, and receive personalized recommendations based on their hotel bookings. This service will consume external event APIs to provide up-to-date event information.

3. **Payment Service**:
   The payment service will securely process payments via external gateways (e.g., Stripe, PayPal) and store transaction data in a **MySQL** database. The system will comply with **PCI-DSS** standards for secure payment processing.

4. **Notification Service**:
   This service will handle notifications sent to users, including booking confirmations, reminders, and event recommendations. It will support multiple channels (e.g., email, SMS) and allow for dynamic content through **notification templates**.

5. **User Management Service**:
   The system will allow users to create accounts, authenticate securely, and manage their personal data and bookings. This service will integrate with **OAuth2** for authentication and provide a dashboard for managing bookings.

Constraints:

- The system must be designed for cloud deployment, supporting platforms like **AWS**.
- The system should comply with **GDPR** for user data protection and **PCI-DSS** for secure payment handling.
- The backend should be able to handle high traffic during peak seasons (e.g., holidays, events), ensuring optimal performance and uptime.
- The system should be modular, allowing for easy integration with third-party APIs (hotel booking, event ticketing, payment processing).
- The system must be secure, protecting user data through **SSL encryption** and **OAuth2 authentication**.

2. Functional Requirements

2.1 Hotel Booking Service

1. **Hotel Search**

   - **Input**: User provides destination, check-in/check-out dates, room type (single, double, family), and number of guests.
   - **Process**: The system queries the external hotel API to retrieve real-time hotel availability that matches the user's input criteria (destination, dates, and room type). The system also checks availability from the **MySQL** database (if cached data is available).
   - **Output**: A list of available hotels is returned to the user, sorted by relevance (e.g., price, rating, availability). The list includes room types, prices, and other relevant information.
   - **Error Handling**: If no hotels match, the system displays: "No hotels available for the selected dates." If there is a failure in fetching data from the external hotel API, the system shows: "Unable to retrieve hotel data. Please try again later."

2. **Hotel Booking**
    - **Input**: User selects a hotel, room type, and provides personal details (name, email, phone number) and payment details (credit card, etc.).
    - **Process**: The system verifies room availability by checking the **MySQL** database for booking conflicts and the payment information via the integrated payment gateway. It then processes the payment using an external payment gateway (e.g., Stripe or PayPal).
    - **Output**: Upon successful booking, the system generates a unique booking ID and stores the booking details in the database. The user receives a booking confirmation via email and SMS.
    - **Error Handling**: If the payment fails, the system notifies the user: "Payment failed. Please check your payment details and try again." If the room is no longer available, the system displays: "The selected room is no longer available."
3. **Booking Management**
    - **Input**: User logs in and accesses their dashboard to manage bookings.
    - **Process**: The system retrieves and displays all the active and past bookings associated with the user's account from the **MySQL** database.
    - **Output**: The dashboard shows booking details, including hotel name, room type, dates, payment status, and booking ID.
    - **Error Handling**: If no bookings exist, the system displays: "You have no active bookings." If there is an issue retrieving data from the database, the system shows: "Unable to retrieve booking details. Please try again."

2.2 Event Ticketing Service

1. **Event Search**
    - **Input**: User provides event search criteria, including event type (e.g., concert, theater), location, and dates.
    - **Process**: The system queries an external event API for available events that match the user's input. The system may also use cached data from previous searches stored in the **MySQL** database to speed up search results.
    - **Output**: The system displays a list of events matching the criteria, including event name, date, time, venue, ticket availability, and prices.
    - **Error Handling**: If no events match, the system displays: "No events found for the selected criteria." If the external API call fails, the system will display: "Unable to retrieve event data. Please try again later."

2. **Event Ticket Purchase**
    - **Input**: User selects an event, specifies the number of tickets, and enters payment details.
    - **Process**: The system verifies ticket availability and proceeds to process the payment via a third-party payment gateway (e.g., Stripe). The system updates the ticket inventory upon successful payment and stores the transaction details in the database.
    - **Output**: The user receives a ticket confirmation email and SMS, including the event details, ticket quantity, and booking ID.
    - **Error Handling**: If the payment fails, the system informs the user: "Payment unsuccessful. Please try another payment method.
3. **Event Recommendations (Based on Hotel Booking)**
    - **Input**: User has an active hotel booking and accesses the recommendations section for nearby events.
    - **Process**: The system retrieves the user's hotel booking details (location, check-in/check-out dates) from the database and queries the event API for events that happen in the same location during the user's stay.
    - **Output**: The system displays recommended events based on location and dates.
    - **Error Handling**: If no relevant events are found, the system displays: "No events found during your stay."

2.3 Payment Service

1. **Payment Processing**
    - **Input**: User provides payment details (credit/debit card information, expiration date, CVV).
    - **Process**: The system verifies the payment details and sends the payment request to an external payment gateway (e.g., Stripe, PayPal). The system ensures secure handling of payment data using **SSL/TLS** encryption and complies with **PCI-DSS** standards.
    - **Output**: Upon successful payment, the system updates the user's booking or event ticket status to "Confirmed" and stores the payment transaction in the database.
    - **Error Handling**: If payment fails, the system shows: "Payment failed. Please check your payment details and try again." If the payment gateway is unavailable, the system displays: "Unable to process payment. Please try again later."
2. **Refund Handling**
    - **Input**: User requests a refund for a canceled booking or event ticket.
    - **Process**: The system verifies the cancellation eligibility (e.g., within the refund period) and initiates the refund request through the payment gateway.
    - **Output**: The system updates the booking or ticket status to "Refunded" and processes the refund transaction.
    - **Error Handling**: If the refund fails, the system displays: "Refund failed. Please try again later."

2.4 Notification Service

1. **Booking Confirmation Email**
    - **Input**: A successful hotel booking or event ticket purchase.
    - **Process**: The system generates a booking confirmation email, including the booking details (hotel/event name, dates, booking ID) and sends it to the user.
    - **Output**: The user receives an email containing booking confirmation information.
    - **Error Handling**: If the email cannot be sent, the system logs the error and retries sending the email.
2. **Booking Confirmation SMS**
    - **Input**: A successful hotel booking or event ticket purchase.
    - **Process**: The system generates and sends an SMS containing the booking details to the user.
    - **Output**: The user receives an SMS with booking confirmation information.
    - **Error Handling**: If the SMS cannot be sent (e.g., incorrect phone number), the system logs the error and retries sending the message.
3. **Event Reminder Notification**
    - **Input**: The event date is approaching (e.g., 1-2 days before the event).
    - **Process**: The system sends a reminder email and SMS to the user with event details (e.g., date, time, location).
    - **Output**: The user receives an email and SMS reminder.
    - **Error Handling**: If the reminder cannot be sent, the system retries sending the notification.

2.5 User Management Service

1. **User Registration**
    - **Input**: User provides their personal information (name, email, password).
    - **Process**: The system validates that the email is not already in use and stores the user's details securely in the database.
    - **Output**: A confirmation email is sent to the user with an account activation link.
    - **Error Handling**: If the email is already registered, the system displays: "Email already in use."

2. **User Login**
    - **Input**: User provides their email and password.
    - **Process**: The system validates the credentials and grants access to the user dashboard if successful.
    - **Output**: The user gains access to their account and booking details.
    - **Error Handling**: If authentication fails, the system shows: "Invalid email or password. Please try again."
3. **Password Reset**
    - **Input**: User requests a password reset by providing their registered email.
    - **Process**: The system sends a password reset link to the user's email.
    - **Output**: The user receives an email with the reset link to change their password.
    - **Error Handling**: If no account exists for the provided email, the system shows: "No account found with this email address."

3. Non-Functional Requirements

3.1 Performance Requirements

1. **Response Time**
    - The system must respond to a user request within **2 seconds** for most actions, including hotel search, event search, and booking confirmation, under normal operational conditions.
    - For actions requiring external API calls (e.g., querying external hotel or event APIs), the response time should not exceed **5 seconds** for 95% of requests.
    - Any process involving significant backend computation (e.g., payment processing) should complete within **10 seconds** in **99% of cases**.
2. **Throughput**
    - The system must be capable of handling **at least 500 concurrent user sessions** during peak hours (e.g., holidays, promotions).
    - The backend should be able to handle **100 bookings per minute** and **50 event ticket purchases per minute** without degradation of performance.
    - For batch operations, such as processing notifications or handling bulk promotions, the system should process at least **5,000 notifications per hour**.

3.2 Reliability Requirements

1. **System Availability**
    - The system should maintain an uptime of **99.9%** per month. This translates to a maximum allowable downtime of **43.2 minutes per month**.
    - Critical services, such as booking and payment processing, must have an availability rate of **99.95%** per month, ensuring high availability during peak periods.
2. **Error Rate**
    - The system should not experience more than **0.5% error rate** for any of the core services (booking, payment processing, event ticketing) under normal operational conditions.
    - The system should automatically handle failures (e.g., service outages, database errors) and retry operations up to **3 times** before reporting a failure to the user.
3. **Fault Tolerance**
    - The system should be able to recover from failures within **5 minutes** for 99% of cases. This includes failures in critical components such as the payment gateway or the event API.
    - The system should be designed for **graceful degradation**, where, if an external service is unavailable (e.g., hotel API), the system should provide cached or fallback data for the user, such as an error message or an outdated availability notice, without causing a full system failure.

3.3 Scalability Requirements

1. **Horizontal Scalability**
    - The system must be able to horizontally scale to handle increased traffic. Specifically, it should support up to **1,000 concurrent users** for each service (hotel booking, event ticketing, user management) by adding additional instances of the service as needed.
    - The **payment processing service** should be able to handle a sudden traffic surge by scaling out to **3 additional instances** during peak times without significant performance degradation.
2. **Data Storage Scalability**
    - The database should be able to scale to handle up to **10 million user records** and **5 million booking transactions** over a period of 5 years.
    - The system must ensure **data consistency** and provide efficient querying for large datasets (e.g., user booking history, event details), ensuring that the system does not experience significant slowdowns when accessing historical data for analysis or reporting.

3.4 Security Requirements

1. **Data Encryption**
    - All sensitive data, including payment information and personal details (e.g., passwords, credit card information), must be encrypted using **AES-256** encryption for data at rest and **TLS 1.2 or higher** for data in transit.
    - User passwords should be hashed using a strong algorithm, such as **bcrypt** or **PBKDF2**, with a minimum of **12 rounds** of hashing.
2. **Authentication**
    - The system must use **OAuth 2.0** for secure authentication and authorization, and should support **multi-factor authentication (MFA)** for administrative access and user login if requested.
    - The user session timeout for inactive users should be set to **15 minutes**. After this time, users should be logged out automatically and required to reauthenticate.
3. **Payment Security**
    - All payment transactions must comply with **PCI-DSS** standards to ensure the protection of payment card information.
    - The system must use a third-party payment gateway (e.g., Stripe, PayPal) that is **PCI-DSS certified** to process all payments securely. The system should never store sensitive payment information, such as credit card numbers.

3.5 Usability Requirements

1. **User Interface**
    - The system should provide a user interface that is intuitive, with **no more than 3 clicks** required to perform any core function (e.g., book a hotel, purchase event tickets, view booking history).
    - The system should support accessibility features that comply with **WCAG 2.1 Level AA** standards to ensure accessibility for users with disabilities.
    - The system should be mobile-friendly, providing a responsive design that ensures full functionality on devices with screen sizes ranging from **320px to 1920px**.
2. **Error Handling and User Feedback**
    - The system must provide clear, actionable error messages that guide users to correct the issue. These messages should be displayed in the user's preferred language (e.g., English, Arabic).
    - All critical errors (e.g., payment failure, booking issue) must be handled with appropriate notifications, and the system should provide the user with a means to contact support if needed (e.g., chat or email).

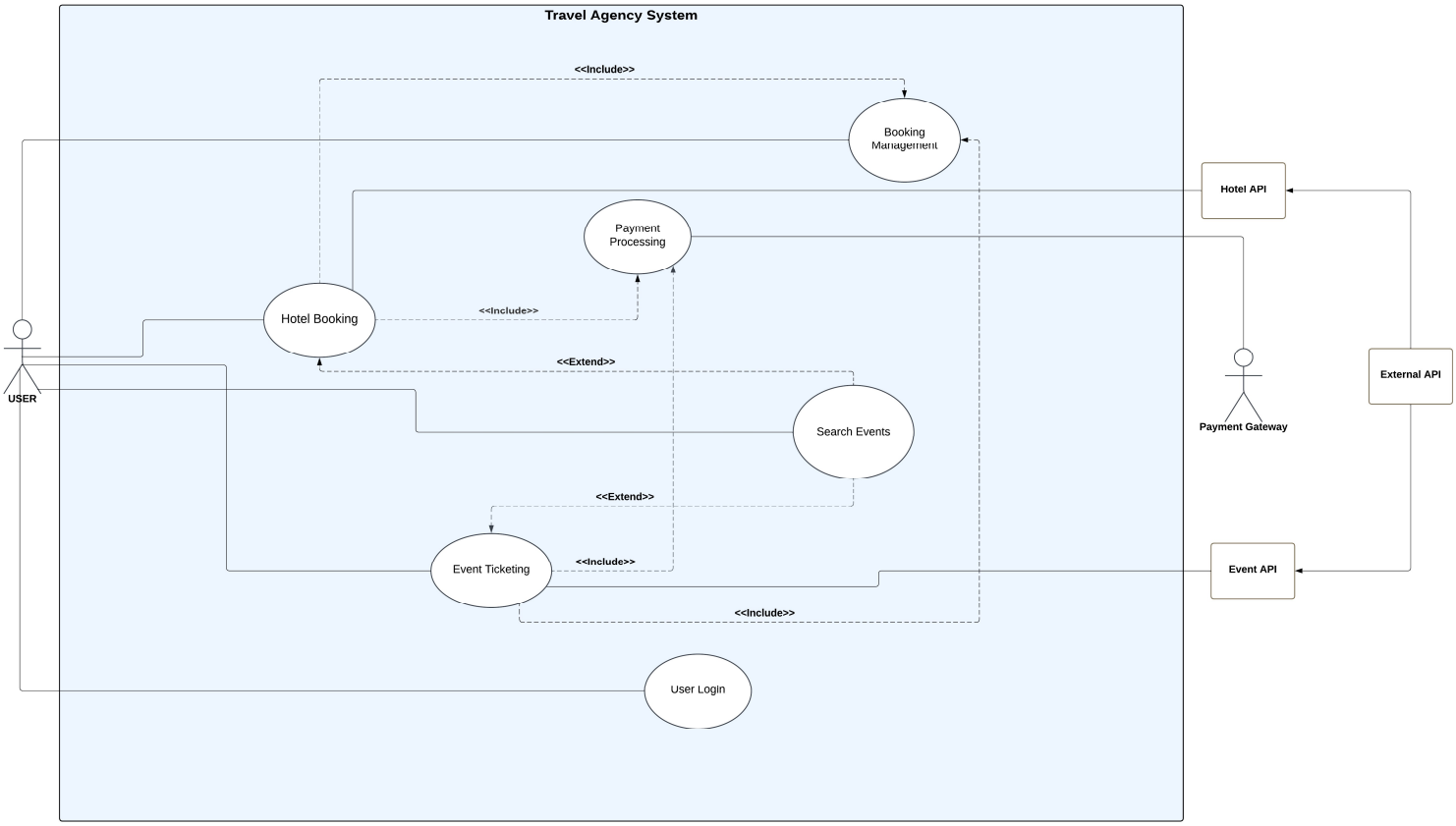
3.6 Compliance Requirements

1. **Data Privacy**
    - The system must comply with **GDPR (General Data Protection Regulation)** for users within the European Union, ensuring that personal data is collected, processed, and stored according to data protection standards.
    - Users must have the ability to request data deletion or modification through a user-friendly interface, and all data deletion requests should be completed within **30 days**.
2. **Regulatory Compliance**
    - The system must adhere to all relevant legal and regulatory requirements specific to the region in which the service operates, including compliance with **PCI-DSS** for payment security, **GDPR** for data privacy, and any local regulations regarding online booking or travel-related services.
    - The system must maintain logs of all user actions related to financial transactions and user account modifications for a minimum of **7 years** for auditing purposes.

3.7 Maintainability Requirements

1. **Code Maintainability**
    - The system must follow **SOLID principles** and best practices in object-oriented programming to ensure that the code is modular, easy to maintain, and scalable.
    - The system should have **unit tests** that cover at least **80% of the codebase** to ensure high-quality software development and facilitate easier future updates and bug fixes.
2. **Documentation**
    - Comprehensive technical documentation must be provided for developers, including an API reference, system architecture diagrams, and guidelines for integrating third-party services.
    - End-user documentation should be available, detailing how users can interact with the system, make bookings, purchase tickets, and manage their accounts.

4. Use Case Model



5. Use Case Tables

Use Case Table 1: Hotel Search

Use Case ID	**UC-001**	
Use Case Name	Hotel Search	
Actors	User	
Pre-conditions	User is on the hotel search page.	
Post-conditions	The system displays a list of available hotels matching the user's search criteria.	
Flow of Events	**User Action**	**System Action**
1.	User enters destination, check-in/check-out dates, room type, and number of guests.	System queries the external hotel API for available rooms that match the input criteria.
2.	User clicks the "Search" button.	System processes the response from the API and displays available hotel options.
3.	User views the list of available hotels.	System sorts the list based on relevance (e.g., price, rating).
4.	User selects a hotel to view more details.	System displays detailed information about the selected hotel.
Exception Flow	**User Action**	**System Action**
1.	User selects a hotel to book.	1. If the hotel is no longer available, the system displays: "Room no longer available."
2.	User clicks "Search" button.	2. If no hotels are available, the system displays: "No hotels available for the selected dates."
3.	User submits search criteria.	3. If the external API fails, the system displays: "Unable to retrieve hotel data. Please try again later."
Includes	**None**	
Notes and Issues	- Ensure that external hotel APIs are reliable. - Use caching to improve search response time.	

Use Case Table 2: Event Ticketing

Use Case ID	**UC-002**	
Use Case Name	Event Ticketing	
Actors	User	
Pre-conditions	User is on the event search page.	
Post-conditions	The system processes the ticket purchase, confirms the transaction, and sends a confirmation.	
Flow of Events	**User Action**	**System Action**
1.	User enters event search criteria (location, type, date).	System queries the external event API for matching events.
2.	User selects the event and specifies the number of tickets.	System checks ticket availability and prompts the user for payment details.
3.	User provides payment details and clicks "Confirm Purchase."	System processes the payment request, and updates ticket availability.
4.	User waits for confirmation message.	System processes the payment and confirms transaction success.
5.	User receives ticket confirmation message.	System generates a unique ticket confirmation number and stores the transaction in the database.
6.	User receives ticket confirmation via email and SMS.	System sends the ticket details via email and SMS to the user.
Exception	**User Action**	**System Action**
1.	User selects an event to purchase tickets for.	If the event is sold out, the system displays: "Tickets no longer available for this event."
2.	User enters payment details.	If payment fails (e.g., insufficient funds or invalid card), the system displays: "Payment failed. Please try another payment method."
Includes	**None**	
Notes and Issues	- Ensure the payment gateway is secure and supports various payment methods.	

Use Case Table 3: Payment Processing

Use Case ID	**UC-003**	
Use Case Name	Payment Processing	
Actors	User	
Pre-conditions	User has selected a hotel or event ticket and is ready to make a payment.	
Post-conditions	The system processes the payment and updates the booking or ticket status to "Confirmed."	
Flow of Events	**User Action**	**System Action**
1.	User provides payment details (credit card, expiration date, CVV).	1. System verifies payment details and sends a request to the payment gateway (e.g., Stripe, PayPal).
2.	User clicks "Confirm Payment" button.	2. System processes the payment request and checks for payment approval from the gateway.
3.	User waits for payment confirmation.	3. System receives payment approval and updates the payment status to "Successful."
4.	User receives confirmation message.	4. System stores payment information in the **MySQL** database and updates the booking or ticket status to "Confirmed."
5.	User receives payment confirmation via email and SMS.	5. System sends payment confirmation via email and SMS, including transaction details and booking ID.
Exception	**User Action**	**System Action**
1.	User submits payment details.	1. If payment fails (e.g., invalid card or insufficient funds), the system displays: "Payment failed. Please check your payment details and try again."
2.	User submits payment details.	2. If the payment gateway is unavailable, the system retries the transaction up to **3 times** before notifying the user: "Payment gateway is unavailable. Please try again later."
Includes	**None**	
Notes and Issues	- Ensure that all payment transactions comply with **PCI-DSS** security standards.	

Use Case Table 4: User Login

Use Case ID	**UC-004**	
Use Case Name	User Login	
Actors	User	
Pre-conditions	User has an existing account with the system.	
Post-conditions	The user is logged in and granted access to their dashboard.	
Flow of Events	**User Action**	**System Action**
1.	User enters email and password.	1. System checks if the credentials match the stored user data in the database.
2.	User clicks the "Login" button.	2. System verifies the credentials. If correct, grants access to the user's dashboard.
3.	User waits for login confirmation.	3. System provides access to the user's account and displays the dashboard with relevant details (e.g., booking history).
4.	User views the dashboard.	4. System loads user-specific data such as booking history, upcoming events, and notifications.
Exception	**User Action**	**System Action**
1.	User enters incorrect credentials.	1. System displays: "Invalid email or password. Please try again."
2.	User forgets their password.	2. System provides a password reset option, sending an email with a reset link.
Includes	**None**	
Notes and Issues	- Implement **OAuth2** or **JWT** for secure user authentication. - Ensure the login process is fast and user-friendly.	

5. Class Diagram