Software Requirements Specification (SRS) for Online Chatbot-Based Ticketing System

1. Introduction

1.1 Purpose

The primary purpose of an **Online Chatbot-Based Ticketing System for Museums** is to create a modernized, automated ticketing system that addresses inefficiencies in traditional booking systems, enhances the customer experience, and supports museum management in making data-driven decisions. The system aims to simplify ticket booking, provide multilingual support, and enhance user experience with AI-powered features.

1.2 Scope

The project aims to develop an intelligent and user-friendly **museum ticketing chatbot system** that integrates advanced technologies to enhance the visitor experience and streamline operations. The system includes features such as user registration and authentication, multilingual support, museum search and recommendations, dynamic pricing, secure payment integration, booking management, and analytics. It is designed to provide seamless booking and ticket management services while leveraging AI and ML for personalization, efficiency, and data-driven insights.

2. Overall Description

2.1 System Overview

The chatbot will:

- Interact with users in multiple languages.
- Assist users in museum search and Recommend museums based on user preferences.
- Process bookings with payment integration.
- Handle booking management (modifications, cancellations).
- Send notifications and alerts for user convenience.
- Answer FAQs, providing information about museums, available exhibits, timings, and ticketing policies.

2.2 Product Features

1. User Registration and Authentication
2. Multilingual Interaction
3. Museum Search and Recommendations
4. Dynamic Ticket Pricing
5. Payment Integration
6. Booking Confirmation
7. Booking Management
8. Notifications and Alerts
9. Analytics and Reporting
10. Accessibility Features

11. Real-Time Crowd Prediction
12. Chat Context Awareness

2.3 Assumptions and Dependencies

- Users will have internet access.
- Museums will provide updated data (ticket prices, timings, etc.).
- Third-party APIs for payments, text-to-speech, and speech-to-text are reliable.

---

3. Functional Requirements

3.1 User Registration and Authentication

- **Description**: Allows users to register with their email and securely authenticate using OTP (One-Time Password). Ensures secure access and unique user identification for the system.
- **Functionalities**:
  - Register users with email
  - Authenticate using OTP
  - Secure storage of user data
- **Models/tools used for implementing**:
  - Flask Authentication (Python)
  - Rule-based OTP validation (Email based OTP Authentication)

3.2 Multilingual Support

- **Description**: Provides the ability for users to interact with the chatbot in their preferred language. Uses language detection and real-time translation to enhance user experience and accessibility for diverse audiences.
- **Functionalities**:
  - Detect user's preferred language.
  - Translate chatbot responses dynamically.
- **Models/tools used for implementing:**
  - Flask-Babel (for Python)
  - langdetect for language detection.
  - Google Translate API for translation tasks.

3.3 Museum Search and Recommendations

- **Description**: Offers a search interface to find museums based on user queries. Recommends museums tailored to user interests, location, or categories, enhancing discoverability and engagement.
- **Functionalities**:
  - Search museums by name, location, or category
  - Recommend museums based on user preferences
- **Models/Tools for implementing:**
  - Content-Based Filtering Models (TF-IDF or embeddings from Word2Vec).
  - K-Nearest Neighbor (KNN) for similarity-based recommendations.
  - Elasticsearch for search functionality

3.4 Dynamic Pricing

- **Description**: Implements a flexible pricing system that adjusts ticket rates based on demand, time of visit, and visitor patterns. Helps optimize revenue and manage crowd control effectively.
- **Functionalities**:
    - Adjust ticket prices dynamically based on demand and time
    - Analyse visitor trends
- **Models/Tools for implementing:**
    - ARIMA or Prophet for demand forecasting (time-series models).
    - LSTMs or Transformer-based models for advanced demand forecasting.
    - Regression Models (XGBoost, LightGBM) for pricing optimization.
    - Clustering Models (K-Means) for user segmentation.

3.5 Payment Integration

- **Description**: Provides secure and seamless payment options through integration with a dummy or third-party payment gateway. Ensures smooth transaction processing and generates receipts.
- **Functionalities**:
    - Process transactions.
    - Generate transaction receipts.
- **Methods/Tools for implementing:**
    - Handled by Stripe, Razorpay, or PayPal SDKs

3.6 Automated Ticket Booking and Confirmation

- **Description**: Enables users to book tickets effortlessly through the chatbot by specifying preferences such as date, time, ticket quantity, and event type. The system verifies real-time availability, processes payments securely, and sends instant booking confirmations via email or chatbot messages, including booking details for user reference. This streamlines the entire ticketing process for convenience and reliability.
- **Functionalities**:
    - Handle automated bookings based on user inputs.
    - Send confirmations via email or chatbot.
- **Methods/Tools for implementing:**
    - Rule-based logic suffices for sending confirmations via email
    - Integrate with messaging platforms by Third Party Tool API calls

3.7 Booking Management

- **Description**: Allows users to view, modify, or cancel bookings. Offers the flexibility to adjust ticket numbers, change time slots, or reschedule visit dates as per user convenience.
- **Functionalities**:
    - Allow modifications to existing bookings.
    - Handle cancellations and refunds.
- **Methods/Tools for implementing:**
    - Reinforcement Learning (Q-Learning) for optimizing rescheduling or cancellations based on system capacity.
    - KMeans Clustering to segment users based on booking behaviors and preferences.

3.8 Notifications and Alerts

- **Description**: Sends timely reminders and updates related to bookings, events, or visit schedules. Ensures users are informed about cancellations, reschedules, or other changes.
- **Functionalities**:
    - Send timely notifications.
- **Models/Tools for implementing:**
    - LSTM for predicting and scheduling user notifications based on historical data.
    - Random Forest Classifier for alert categorization.
    - Email based messaging

3.9 Analytics and Reporting

- **Description**: Generates insights for administrators by analysing visitor data, booking trends, and revenue patterns. Supports decision-making with comprehensive reports and predictive analytics.
- **Functionalities**:
    - Analyse visitor patterns.
    - Generate reports for museum administrators.
- **Models/Tools for implementing:**
    - KMeans Clustering for visitor segmentation.
    - ARIMA for forecasting visitor trends.
    - Logistic Regression for predictive analysis.

3.10 Accessibility Features

- **Description**: Ensures inclusivity by providing features like voice-to-text and text-to-voice interaction for users with disabilities. Enhances the usability of the system for diverse user groups.
- **Functionalities**:
    - Enable voice commands and responses.
    - Improve usability for differently-abled users.
- **Models/Tools for implementing:**
    - Wav2Vec 2.0 (Speech-to-Text).
    - Tacotron 2 or Google TTS (Text-to-Speech).

3.11 Real-Time Crowd Prediction

- **Description:** Predicts visitor crowd levels in real time using historical data, bookings, and live footfall data. Helps users plan visits during less crowded hours and supports crowd management.
- **Functionalities:**
    - Predict crowd levels in real-time.
    - Provide recommendations for less crowded visit times.
- **Models/Tools used for implementing**:
    - LSTM (Long Short-Term Memory) for time-series prediction.

3.12 Chat Context Awareness

- **Description:** Enables the chatbot to remember previous interactions and tailor responses based on user history and current context. Enhances user experience with personalized and coherent communication.
- **Functionalities:**

    - Maintain context across multiple interactions.
    - Personalize responses based on user history and preferences.
- **Models/Tools used for implementing**:
    - GPT-based Chat Models (e.g., OpenAI GPT-4).

---

4. Non-Functional Requirements

4.1 Performance

- The system must handle up to 100 concurrent users without degradation.
- API response time should not exceed **2 seconds** under normal load.
- Real-time booking updates should reflect within **5 seconds**.

4.2 Scalability

- Database architecture should accommodate up to 10 million records for user and booking data.
- The system should be able to scale horizontally to support **increased user traffic**

4.2 Reliability and Availability

- Ensure 99.9% uptime for critical functionalities like booking and payment.
- Booking services must ensure transaction integrity to avoid duplicate bookings or data corruption.

4.3 Usability

- The user interface should be intuitive and user-friendly, supporting **navigation.**
- Multilingual support should be seamless, with no visible delays during language switching.

4.4 Security

- Implement **OAuth 2.0** for secure authentication and authorization.
- Ensure protection against common vulnerabilities like **SQL injection, XSS, and CSRF attacks**.

4.5 Maintainability

- Codebase should adhere to clean coding practices to facilitate updates and bug fixes.
- System updates (e.g., adding a new museum or dynamic pricing rules) should take no more than 4 hours to implement.

4.6 Fault Tolerance

- In case of system failure, auto-recovery mechanisms should restore services within 1 minute.
- Payment transactions must be retried automatically in case of a failure to ensure no financial loss.

4.7 Data Backup and Recovery

- Perform automated daily backups of user, booking, and transaction data.

- Data restoration from backups should not take more than 30 minutes.

4.8 Response Time for Notifications

- Notifications and alerts (email, SMS, or chatbot) should be delivered within **10 seconds** of triggering an event.

4.9 Audit and Logging

- Maintain detailed logs for all user interactions and transactions for auditing purposes.
- Logs should be retained for at least 1 year and accessible via an admin interface.

4.10 Accessibility

- Voice-based navigation and text-to-speech features should function with **95% accuracy**.

---

5. Technical Requirements

5.1 Hardware

- Server with at least 8GB RAM and 100GB SSD.

5.2 Software

- Python (for backend logic)
- APIs: Google Translate, PyOTP, Dummy Payment Gateway
- Libraries: Scikit-learn, NLTK, Matplotlib

---

6. Appendices

A. Datasets

- Historical ticket sales data
- Museum metadata

B. Tools

- IDE: VS Code
- Frameworks: Flask/Django

C. References

- Existing ticketing systems
- Research on chatbot and AI applications