**System Requirements Document (SRD) for Chatbot Application**

**1. Introduction**

This document specifies the system requirements for the development and deployment of a chatbot application designed to improve customer engagement and support for [Company Name]. The chatbot will be capable of handling customer queries, providing information, assisting with transactions, and escalating issues to human agents when necessary.

**2. System Overview**

The chatbot application will be a cloud-based solution integrated with the company's existing infrastructure. It will leverage machine learning for natural language processing (NLP) and will be accessible via web and mobile interfaces.

**3. Functional Requirements**

**3.1 User Interaction**

* **Greeting and Introduction**:
  + The chatbot should initiate conversations with a friendly greeting and introduction.
  + Example: "Hello! I'm [Chatbot Name], your virtual assistant. How can I help you today?"
* **Natural Language Processing (NLP)**:
  + The chatbot should understand and process user inputs in natural language using machine learning models such as BERT or GPT.
  + Must support multiple languages including English, Spanish, French, and German.
* **Multi-turn Conversations**:
  + The chatbot should maintain context and continuity in multi-turn conversations.
  + Example: If a user asks about order status and then asks for delivery details, the chatbot should understand the context of the order.
* **Error Handling**:
  + Provide appropriate responses when unable to understand the user or process a request.
  + Example: "I'm sorry, I didn't understand that. Can you please rephrase?"

**3.2 Customer Support**

* **FAQ Handling**:
  + The chatbot should respond to frequently asked questions using a pre-defined knowledge base stored in a database table named faq.
* **Ticket Generation**:
  + Create support tickets for unresolved queries and store them in a table named support\_tickets.
  + Example fields: ticket\_id, user\_id, issue\_description, status, created\_at.
* **Escalation to Human Agents**:
  + Seamlessly transfer conversations to human agents via an internal ticketing system integrated with the company's CRM.
* **Status Updates**:
  + Provide users with updates on the status of their queries or tickets.
  + Example: "Your ticket #12345 is currently being reviewed by our support team."

**3.3 Transactional Capabilities**

* **Order Tracking**:
  + Allow users to track their orders by retrieving order details from the orders database table.
  + Example fields: order\_id, user\_id, order\_status, delivery\_date.
* **Product Information**:
  + Provide detailed information about products or services from the products database table.
  + Example fields: product\_id, product\_name, description, price, stock\_level.
* **Booking and Reservations**:
  + Facilitate bookings or reservations and store details in the reservations table.
  + Example fields: reservation\_id, user\_id, service\_id, booking\_date, status.
* **Payment Processing**:
  + Integrate with payment gateways (e.g., Stripe, PayPal) for secure payment processing.
  + Store transaction details in the transactions table.
  + Example fields: transaction\_id, user\_id, amount, payment\_method, status.

**3.4 Personalization**

* **User Authentication**:
  + Authenticate users using OAuth2 or JWT tokens.
  + Store user credentials and details in the users table.
  + Example fields: user\_id, username, password\_hash, email, phone\_number.
* **User Preferences**:
  + Remember user preferences and past interactions stored in the user\_preferences table.
  + Example fields: user\_id, preference\_type, preference\_value.
* **Tailored Responses**:
  + Provide responses tailored to individual user needs and preferences.
  + Example: "Welcome back, [User Name]! How can I assist you today?"

**3.5 Multilingual Support**

* **Language Detection**:
  + Automatically detect the user's language and respond accordingly.
* **Language Options**:
  + Support for multiple languages configured in the languages table.
  + Example fields: language\_code, language\_name.
* **Translation Services**:
  + Integrate with translation APIs (e.g., Google Translate) for real-time translation.

**3.6 Analytics and Reporting**

* **User Interaction Analytics**:
  + Track and analyze user interactions with the chatbot stored in the interaction\_logs table.
  + Example fields: interaction\_id, user\_id, timestamp, query, response\_time.
* **Performance Metrics**:
  + Monitor chatbot performance metrics such as response time, resolution rate, etc.
  + Example metrics: Average response time, resolution rate, user satisfaction score.
* **Reporting Tools**:
  + Provide tools for generating detailed reports on chatbot usage and performance.
  + Store report configurations in the reports table.
  + Example fields: report\_id, report\_name, created\_at, created\_by.

**4. Non-Functional Requirements**

**4.1 Performance**

* **Response Time**:
  + The chatbot should respond to user queries within 2 seconds.
* **Scalability**:
  + The system should handle up to 10,000 concurrent users.

**4.2 Reliability**

* **Uptime**:
  + The chatbot should have an uptime of 99.9%.
* **Error Rate**:
  + The system should have an error rate of less than 0.1%.

**4.3 Usability**

* **User Interface**:
  + The UI should be intuitive and easy to navigate.
  + Should be responsive and work across various devices and screen sizes.
* **Accessibility**:
  + Ensure the chatbot is accessible to users with disabilities, adhering to WCAG 2.1 guidelines.

**4.4 Maintainability**

* **Code Quality**:
  + Follow best practices for code quality and documentation.
  + Use a version control system (e.g., Git) for source code management.
* **Updates and Upgrades**:
  + The system should support easy updates and upgrades.
  + Ensure backward compatibility where necessary.

**4.5 Security**

* **Authentication**:
  + Implement secure authentication mechanisms (e.g., OAuth2, JWT).
  + Use HTTPS for all communications.
* **Data Protection**:
  + Ensure robust data protection measures are in place, including encryption of sensitive data.
  + Comply with relevant data protection regulations (e.g., GDPR, CCPA).

**5. System Architecture**

**5.1 Overview**

* **Frontend**:
  + Technologies: HTML, CSS, JavaScript, React.js
  + Description: Provides the user interface for web and mobile interactions.
* **Backend**:
  + Technologies: Python, Flask/Django, Node.js
  + Description: Handles business logic, API requests, and integration with external systems.
* **Database**:
  + Technologies: MySQL, MongoDB
  + Description: Stores user data, chatbot interactions, and other application data.
* **NLP Engine**:
  + Technologies: TensorFlow, Hugging Face Transformers
  + Description: Processes natural language queries and generates responses.
* **Cloud Infrastructure**:
  + Providers: AWS, Azure
  + Description: Hosts the application and ensures scalability and reliability.

**5.2 Database Schema**

1. **users**
   * user\_id (Primary Key)
   * username
   * password\_hash
   * email
   * phone\_number
2. **faq**
   * faq\_id (Primary Key)
   * question
   * answer
3. **support\_tickets**
   * ticket\_id (Primary Key)
   * user\_id (Foreign Key)
   * issue\_description
   * status
   * created\_at
4. **orders**
   * order\_id (Primary Key)
   * user\_id (Foreign Key)
   * order\_status
   * delivery\_date
5. **products**
   * product\_id (Primary Key)
   * product\_name
   * description
   * price
   * stock\_level
6. **reservations**
   * reservation\_id (Primary Key)
   * user\_id (Foreign Key)
   * service\_id
   * booking\_date
   * status
7. **transactions**
   * transaction\_id (Primary Key)
   * user\_id (Foreign Key)
   * amount
   * payment\_method
   * status
8. **user\_preferences**
   * user\_id (Primary Key, Foreign Key)
   * preference\_type
   * preference\_value
9. **languages**
   * language\_code (Primary Key)
   * language\_name
10. **interaction\_logs**
    * interaction\_id (Primary Key)
    * user\_id (Foreign Key)
    * timestamp
    * query
    * response\_time
11. **reports**
    * report\_id (Primary Key)
    * report\_name
    * created\_at
    * created\_by

**6. Integration Requirements**

* **CRM Integration**:
  + Integrate with the company's existing CRM to access customer data and support ticketing.
* **Payment Gateway Integration**:
  + Integrate with Stripe and PayPal for secure payment processing.
* **Translation API Integration**:
  + Use Google Translate API for real-time language translation.

**7. Testing Requirements**

**7.1 Unit Testing**

* **Scope**:
  + Test individual components and functions for correctness.
  + Technologies: PyTest, Mocha, Chai

**7.2 Integration Testing**

* **Scope**:
  + Test the interaction between different components and systems.
  + Technologies: Postman, Selenium

**7.3 Performance Testing**

* **Scope**:
  + Test the system's performance under various load conditions.
  + Technologies: JMeter, LoadRunner

**7.4 Security Testing**

* **Scope**:
  + Test for vulnerabilities and ensure data protection.
  + Technologies: OWASP ZAP, Burp Suite

**8. Deployment Requirements**

* **Environment**:
  + Development: Local machines, Docker
  + Staging: AWS EC2 instances
  + Production: AWS ECS, RDS for databases
* **CI/CD Pipeline**:
  + Tools: Jenkins, GitHub Actions
  + Description: Automate build, test, and deployment processes.

**9. Maintenance and Support**

* **Monitoring**:
  + Tools: Prometheus, Grafana
  + Description: Monitor system performance and alert on issues.
* **Logging**:
  + Tools: ELK Stack (Elasticsearch, Logstash, Kibana)
  + Description: Collect and analyze logs for troubleshooting.
* **Backup and Recovery**:
  + Tools: AWS Backup, Azure Backup
  + Description: Regular backups of databases and critical data.

**10. Assumptions and Dependencies**

* **Assumptions**:
  + Users have access to the internet.
  + Users have basic digital literacy.
* **Dependencies**:
  + Availability of CRM and other backend systems for integration.
  + Third-party services for language translation and payment processing.

**11. Risks and Mitigations**

* **Risk**: Inaccurate responses due to NLP limitations.
  + **Mitigation**: Regularly update and train the NLP model.
* **Risk**: Data breaches and security threats.
  + **Mitigation**: Implement robust security measures and conduct regular audits.
* **Risk**: High volume of queries overloading the system.
  + **Mitigation**: Ensure scalability and load balancing mechanisms.

**12. Timeline**

1. **Phase 1**: Requirements Gathering and Planning (2 weeks)
2. **Phase 2**: Design and Prototyping (4 weeks)
3. **Phase 3**: Development (8 weeks)
4. **Phase 4**: Testing (4 weeks)
5. **Phase 5**: Deployment and Go-Live (2 weeks)
6. **Phase 6**: Post-Launch Support and Maintenance (Ongoing)

**13. Appendices**

**13.1 Glossary**

* **NLP**: Natural Language Processing
* **CRM**: Customer Relationship Management
* **API**: Application Programming Interface
* **GDPR**: General Data Protection Regulation
* **CCPA**: California Consumer Privacy Act

**13.2 References**

* [Include references to any relevant documents, research papers, or standards followed]