Problem Defination

Scenario:

You are tasked with creating an AI Receptionist for a small business. The AI should assist customers by answering common inquiries, scheduling appointments, and customizing responses based on the business's preferences.

Tasks:

1. Business Information Integration

- **Goal**: Use information from a simulated database (e.g., MongoDB) and a mock business website to answer customer inquiries about services and availability.
- **Requirements**:
    1. Create a MongoDB collection business_data to store information:
        - Business Name
        - Services Offered (Service Name, Description, Price)
        - Operating Hours
        - Contact Information
    2. Simulate a mock business website by providing an HTML page with the same details.
    3. Build an API to fetch business information based on customer queries (e.g., "What services do you offer?").

---

2. Appointment Scheduling

- **Goal**: Implement an appointment scheduling system based on available slots.
- **Requirements**:
    1. Create a calendar collection in MongoDB to store:
        - Available Slots (Date, Start Time, End Time)
        - Booked Appointments
    2. Design an endpoint /schedule that:

        - Accepts customer name, preferred service, and requested date/time.
        - Checks for availability and books an appointment if the slot is free.
        - Returns a confirmation message with the appointment details.
    3. Handle edge cases like double-booking and invalid time slots.

---

3. AI-Powered Customer Interaction

- **Goal**: Use a simple NLP model to process customer inquiries and respond appropriately.
- **Requirements**:
    1. Train or use a pre-trained NLP model (e.g., Hugging Face Transformers) to understand:
        - Service-related questions ("What services do you offer?")
        - Scheduling requests ("Can I book an appointment for tomorrow at 3 PM?")
        - Miscellaneous queries ("What are your operating hours?")
    2. Use the AI model to provide intelligent responses, fetching data from business_data or calendar collections as needed.
    3. Log all interactions in a customer_queries collection for future analytics.

---

4. Personalization and Customization

- **Goal**: Allow customization of responses to align with business preferences.
- **Requirements**:
    1. Add a custom_responses collection in MongoDB:
        - Fields: Query Type, Custom Response Template (e.g., "We offer {Service_Name} for ${Price}.")
    2. Build an admin interface to update these custom response templates.

3. Implement dynamic response generation using templates stored in custom_responses.

---

5. Advanced Features (Bonus)

- **Optional Enhancements**:
  - Integrate Google Calendar API to sync available slots and bookings in real-time.
  - Enable multi-language support using a translation API.
  - Add sentiment analysis to respond empathetically (e.g., cheerful tone for happy queries, apologetic tone for complaints).
  - Use a small LLM (e.g., GPT-3.5) for more nuanced query handling.

Deliverables

1. **Code Repository**:
   - Include a well-structured Node.js or Python application (use Flask, Express, or similar frameworks).
   - Use MongoDB for database operations.
2. **API Documentation**:
   - Include endpoints, request/response examples, and error handling details.
3. **Demo**:
   - Provide a Postman collection or a simple frontend (e.g., React or plain HTML) to test the functionality.
4. **Deployment**:
   - Host the application locally or on a service like Heroku, Vercel, or AWS, and provide access details.

Evaluation Criteria

1. **Functionality**:
   - Correctness of business information fetching and appointment scheduling.

2. **AI Implementation**:
    - Quality of NLP integration and query understanding.
3. **Customization**:
    - Ease of updating custom response templates and adaptability to business preferences.
4. **Scalability**:
    - Robust handling of edge cases, clean architecture, and extensibility.
5. **Documentation**:
    - Clear instructions for setting up, running, and testing the application.