ChatBot for Diagnostic Center:

1. Introduction

1.1 Project Overview

The AI chatbot for diagnostic centers will provide the following features:

- Query resolution (test prices, doctor availability, etc.)
- Appointment booking
- Test report progress tracking
- Billing information
- Home diagnosis and ambulance booking

2. System Requirements

2.1 Functional Requirements

- **User Query Handling**:
    - Provide information on test prices.
    - Check doctor availability.
    - Give details about diagnostic tests and reports.
- **Booking Services**:
    - Book appointments, ambulance, and home diagnosis.
- **Billing**:
    - Fetch billing details and payment history.
- **Report Tracking**:
    - Update and share test report statuses.

2.2 Non-Functional Requirements

- **Performance**: Responses should be within seconds.
- **Scalability**: Handle multiple users simultaneously.
- **Security**: Encrypt sensitive data like patient details.
- **Usability**: Provide a user-friendly interface.

2.3 Technical Requirements

- **Programming Language**: Python for backend and ML.
- **Database**: MySQL/PostGreSQL
- **Hosting**: Cloud-based (AWS, GCP).
- **Libraries/Frameworks**:
    - **NLP**: NLTK, spaCy, Hugging Face Transformers.
    - **Web Framework**: Django
    - **Frontend**: React

3. System Design

3.1 Architectural Design:

1. **Presentation Layer**: Chatbot UI (Web/Mobile).
2. **Application Layer**: Backend with logic.
3. **Database Layer**: Stores all data.

3.2 Database Design

Designing the database schema:

- **Tables**:
    - Tests:
        - Fields: test_id, test_name, price
    - Doctors:
        - Fields: doctor_id, name, specialty, availability
    - Appointments:
        - Fields: appointment_id, user_id, doctor_id, date, status
    - Reports:
        - Fields: report_id, user_id, status, uploaded_date
    - Services:
        - Fields: service_id, type (e.g., ambulance, home diagnosis), availability

3.3 Data Flow Diagram (DFD)

User sends query → NLP processes query → Intent matched → Relevant data fetched from database → Output shown to user

3.4 Use Case Diagram

Use Cases:

- User queries test prices → Chatbot fetches prices.
- User books appointment → Chatbot confirms and updates the database.
- User checks report progress → Chatbot fetches report status.

4. Detailed Design

4.1 NLP Pipeline

- **Intent Recognition**:
  - Use supervised learning to classify user queries.
  - Example: *"What is the price of an MRI?" → Intent: test_price_query*.
- **Entity Extraction**:
  - Recognize keywords like test names, doctor names, and dates using NLTK, spaCy or custom-trained models.

4.2 Backend Logic

- **API Design**:
  - Endpoints:
    - /get_test_price: Fetches test prices.
    - /check_doctor_availability: Checks doctor schedules.
    - /book_appointment: Books a test or consultation.
    - /track_report: Retrieves report status.
    - /book_service: Books ambulance/home diagnosis.
- **Appointment Booking Logic**:
  - Validate user input (date, time).
  - Check doctor availability and confirm booking.

4.3 User Interface (UI)

- **Chatbot Interface**:
    - Simple input box for user queries.
    - Display dynamic responses in chat bubbles.
- **Appointment Booking Form**:
    - Embedded within the chatbot when booking is initiated.

5. Implementation

5.1 Development Stages

1. **NLP Module**:
    - Train intent classification and NER models.
    - Test on sample queries.
2. **Backend Development**:
    - Build APIs.
    - Integrate with NLP for processing queries.
3. **Database Integration**:
    - Set up Database and design tables.
4. **Frontend Development**:
    - Build UI with React/HTML+CSS.
5. **Integration and Testing**:
    - Connect frontend, backend, and database.
    - Test with real-world scenarios.

5.2 Tools and Technologies

- **IDE**: PyCharm, VS Code.
- **Version Control**: Git/GitHub.
- **Database**: MySQL Workbench.
- **Cloud Hosting**: AWS/Heroku.

6. Security Considerations

- **Data Encryption**:
    - Use HTTPS and encrypt sensitive data.
- **Authentication**:
    - Implement OAuth for secure logins.
- **Data Validation**:
    - Sanitize user inputs to prevent SQL injection.

7. Testing and Deployment

7.1 Testing

- **Unit Testing**: Test each module (NLP, API).
- **Integration Testing**: Ensure seamless communication between UI, backend, and database.
- **User Testing**: Gather feedback from real users.

7.2 Deployment

- **Backend**: Deploy on cloud platforms like AWS , Azure , GCP , .
- **Frontend**: Host on a web server (Netlify).
- **Database**: Use managed databases (e.g., AWS RDS).

8. Maintenance

- Regularly update models with new intents/entities.
- Monitor chatbot performance.
- Update database with new tests, services, and doctor schedules.