Python Developer Task Sheet

⬔ Objective:

This assignment is designed to assess your backend development skills in **Python**, particularly in **API integration**, **real-time communication using WebSockets (via python-socketio)**, and **database handling with PostgreSQL**.

You are required to submit **any or three of the tasks within 5 days** from the date of assignment.

Task 1: Homebuyer Enquiry AI Prediction System

⬔ Goal:

Build a Django application that analyzes property enquiry data and uses **AI** to predict whether a buyer is **likely to purchase a flat** based on their enquiry details.

☑ Requirements:

• Create a Django project with models or a CSV file to store enquiry data
• Each enquiry record should include fields such as:
    - Name, Age, Income, City, Property Type, Budget, Follow-ups, Site Visited, Booked (1/0)
• Implement AI logic using Python libraries to:
    - Train a prediction model from the available enquiry data
    - Predict the buyer's intent (Interested / Not Interested)
• Create REST API endpoints:
    - /api/train/ → Train or retrain the AI model
    - /api/predict/ → Predict outcome for a new enquiry
    - /api/insights/ → Return summary or AI insights (optional)
• Display clear JSON responses for training, prediction, and insights

⬔ Tech & Tools:

• **Backend:** Python (Django + Django REST Framework)
• **AI Libraries:** scikit-learn, pandas, numpy, joblib
• **Database:** SQLite or PostgreSQL (as preferred)
• **Version Control:** GitHub for code submission
• (Optional) React or simple HTML template for showing results visually

Duration:

6-9 hours (Take-home assignment)

Bonus Features (Optional):

- Dashboard showing enquiry trends or lead funnel
- Insight on which features most influence buying decisions
- Auto-generated AI summary report
- Simple Chatbot to query insights like “Which city has the most interested buyers?”

OR ....

⬔ Task 2: Real-Time Group Chat with python-socketio

⬔ Goal:

Build a real-time **chat system** using python-socketio, supporting:

- Group chats (chat rooms)
- Text and image sharing
- Real-time message delivery
- Basic notification system

☑ Requirements:

- Use **python-socketio** for WebSocket server
- PostgreSQL to store chat messages and media file references
- Frontend can be anything (React JS preferred)
- Enable:
    - Joining and leaving chat rooms
    - Sending and receiving messages (text + image)
    - Real-time notifications for new messages
- Store all messages in DB with timestamp, sender, content type
- Allow creation of rooms and assigning members

⬔ Features to include:

- Text messages
- Image sharing (upload + preview)
- Real-time notification on new message
- Group chat support
- Message history via API (pagination)

⬔ Tech & Tools:

- Backend: python-socketio, FastAPI or Flask (or Django ASGI with separate socket service)
- DB: PostgreSQL
- File storage: Local or S3 (optional)
- Frontend: React.js (preferred), or Postman testing with socket.io-client

Duration:

9 - 12 hours (Take-home assignment)

OR ....

⬔ Task 3: AI-Powered Telephony Agent (Voice Assistant) - Advantage

⬔ Goal:

Build a Django application that integrates with a telephony service (like Twilio) to create a conversational AI agent capable of handling an outbound or simulated inbound voice call.

☑ Requirements:

- **Core Agent Logic:** Use a powerful LLM (like Google's Gemini API, OpenAI GPT, or a mock-AI service) to generate intelligent, context-aware conversational responses.
- **Telephony Integration:** Integrate with a service like **Twilio** to handle the actual voice call:
    - Setup a **webhook endpoint** in Django that Twilio can call upon an inbound/outbound connection.
    - Use **TwiML** (Twilio Markup Language, generated via the Python SDK) to instruct Twilio to:
        - Convert the user's voice input to text (**Speech-to-Text**).
        - Send the text transcript to your AI logic.
        - Convert the AI's text response back to speech (**Text-to-Speech**) and play it to the caller.
- **Database:** Store a **transcript** of the conversation (user utterance, AI response, and timestamp) in PostgreSQL/SQLite.
- **REST API Endpoints:**
    - /api/call/initiate/: A POST endpoint to trigger a **simulated outbound call** (takes a phone number and initiates the Twilio process).
    - /api/call/webhook/: The primary endpoint for Twilio to send and receive TwiML instructions.
    - /api/call/history/<call_id>/: Return the full chat transcript for a specific call ID.

⬔ Tech & Tools:

- **Backend:** Python (Django + Django REST Framework)
- **AI Libraries: Twilio Python SDK** (for TwiML and API calls), **LLM SDK** (e.g., google-genai or openai) or a **mock-AI utility**.
- **Database:** SQLite or PostgreSQL.
- **Dev Tools: ngrok** (or similar service) to expose your local Django webhook endpoint to Twilio.

⬔ Bonus Features (Optional):

- **Tool Calling:** Give the AI agent a simple **"tool"** to query a piece of mock data (e.g., "The weather in City X is 25°C") and use this information in its spoken response.
- **Agent Persona:** Define a clear, consistent persona for the agent (e.g., "You are a polite, expert property consultant for Task 1").
- **Asynchronous Logging:** Use **Celery** or **Django Channels** to asynchronously log the conversation steps to the database, ensuring the main Twilio webhook response is fast.
- **AI Insights:** Analyze the stored call transcripts to provide simple insights, e.g., "The average call length is 45 seconds."

Duration:

10 - 15 hours (Take-home assignment)

⬔ Deadline:

⏰**5 Days** from the date of task assignment.

Please share the following upon completion:

1. GitHub repo link
2. Short deployment or demo video (optional but preferred)
3. README with setup steps

⬔ Communication:

If you need any clarification, feel free to reach out during the task duration.

We look forward to seeing your implementation!

⬔ Bonus Note:

Completing **both Task 1 (AI Prediction)** and **Task 2 (Group Chat)** will **significantly increase your hiring chances** — as it demonstrates full-stack capability in **Django, API design, and AI integration**.

Candidates are encouraged to complete both if time permits.

—

Team DigitalB.