WhatsApp LLM Bot CRUD Assignment

Subject: Demo To Dos - WhatsApp LLM Bot CRUD

In a nutshell the discussion points and the to do's for demo are as follows also attached the ppt for workflows :

1. You have to create an Entry Point that could be a QR code for the user to gain access.

2. Via this Entry Point the user lands on your WhatsApp Chat window and a chat session is initiated.

3. Discover and choose an LLM model/models that connects with WhatsApp APIs and acts as a Bot to respond to the customer.

4. These LLM models should be conversant and comprehensive in understanding and responding in Hindi, English and the blend of both Languages (Hinglish).

5. These LLM APIs will connect with a backend logic to connect to the DB and perform CRUD and Fetch operations on the DB .

6. Take 5 Restaurant's data and create a table/record in the DB with Fields- Resto Name, Menu (Dishes), Portion sizing of the dishes, Price, Pic of the dish, Address and Google PIN location

of the Resto also the addresses and Google PINs of the branches if there be any.

7. The DB will also have a USER record, where user's name, DOB, email address, Addresses where he/she mostly gets delivered to classify it as office, hostel, PG, college etc. by giving them tags , his food choices like Veg or Non Veg, Food Preferences as Chinese, South Indian etc. Any health condition(diabetes, Blood Sugar etc) or allergies with food, and days that they

don't eat non veg etc. Creating a user profile.

8. Create 2 Records/Documents for the USER Table /Collection at the backend in SQL/NoSQL DB (MongoDB)

9. For a user who lands on your WhatsApp first check if it's a returning user or a New User.

10.If it's a New User then collect the values for the variables in Point (6)

11. Let the LLM ask what they want to order today and based on their reply such as "I want Kuch spicy in Chinese jo heavy na ho"

12. Select the dishes from the restaurant table/collection that the LLM Classifies as "Kuch + Chinese + Spicy + Not Heavy" and the backend logic selects these dishes from the Resto

Table/Collection and forms a set of those dishes with their details such as Dish name, Dish picture, Portion sizes and prices. and create a queue of these.

13. Sort the queue of the dishes based on the distance from the user and the restaurant from which that dish is displayed in ascending order of the distance (that's min distance to the farthest

distance)

14. Display the dishes in sets of 4 in a flat grid of 2x2 each cell is called elements that have dish details : Dish Pic, Dish Name, Portion Sizing it offers (Half/Full), Prices and the Resto Name

it's coming from.

15. After Each Grid, give the user an Action Trigger named "View More" to send the next 4 dishes from the distance sorted list curated at the back in above Point (12)

15. Each Element when downloaded on WhatsApp will expand into a full frame Picture with the details mentioned in Point (14)

16. Once the user selects the dishes, create a cache queue of dishes at the backend to update the Temp Order Table/Document

17. Selection can happen through multiple ways such as User can write "2 Chowmeins 2 Chilli Paneer and 1 Chicken chilly" in chat window as a plain text or reply to a pic sent in the 2X2 grid

writing" Iske 2 half" or "2 half Noodles and 1 Full Chilly Paneer" or 2 Chowmein + 2 Garlic Noodles +1 Manchurian etc. Use your creative design thinking to cover the probable responses

by the user

18. If the user replies "2 Veg Chowmein" and LLM is not able to zero down on the dish selection then again Ask the user like "Which Veg Chowmein among the above" with the pic of those

those items that are veg chowmeins from the collection shown in different grid above along with the Resto Name.

19. If the user is latent in responding after a series of selections then LLM should ask the user after 5 seconds if there is something else you want to look at or is that the complete order?

20. If the user responds, show me something else on top of it. Then Show the suggestions that go along with the dishes. (Recommendations that LLM shall classify that go well with the above

dishes in selection)

21. If the user chooses a certain dish from the recommendations then show them the options based on the selected dishes in the same ascending order of the distance as done above

22. If the user selects some options add them in the queue and ask the user if that's all for the order or he needs to explore some more, show then the recommendations if the user asks to

explore more on what the user is indicating at Ex: Also show me in Mughlai then Curate Mughalai queue and sort in ascending order and 4 elements of the temp list as created in Point

(16) is sent to the user in 2x2 grid format with View More Trigger

23. Once the user indicates finishing the order either by responding that's all once your LLM asks the user anything else or should we finish the order, create the list of the selection

maintained in Point (16) in text format along with portion size, Dish Name, Resto Name, Qty, Price and send it to the user in form of a text list on WhatsApp.

24. Ask the user if he/she wants to Confirm the Order or Edit the order

25. If the user selects to Edit the order then ask the user what he/she would want to edit --> User might say Add this and Remove those 2 dishes or reduce the quantity saying make it half or just one of these than 2.

26. Update the Temp Order Table created in Point (16)

27. Once User exceeds a latency of 5 seconds ..Prompt the user " I consider you're done editing the order" with an Option of Yes or No.

28. User if Presses No then Prompt the user to ask what all things would he/she want to edit and update

29 If User Presses Yes then send the updated order list (Point 16) in Text format as done in Point (23)

30. Prompt again the user on confirmation of the order or Edit of the order as done in Point (24)

31. If this time user selects on Confirming the order then create the QR Code linked to your bank account with the updated Bill Amount and send this QR code along with the total of the Text

order table to the user

32 Track and capture the UPI acknowledgement if the payment is made or not

33 If the UPI Acknowledgement fails --> Prompt the user for the Failed Transaction and Ask him to Pay again on the QR and track for Success or Failure

34 If the acknowledgement is Success, ---> Create the PDF Invoice at the backend and send it to the user on WhatsApp with a Thank You Note and

35 Create this Order Record in the Order Table with Order Details, Dish Details, Customer Details, Delivery Address details, Google PIN of both Customer Delivery Loc and Resto Loc,

Distance in Kms, Resto Details and UPI acknowledgement ID, UPI Status along with Date/Time Stamp

36. Create a dashboard at the backend to view these User Table , Order Table , Resto Table data and records with date and time stamp.

<u>Add in Model and API keys LLM Bot Assignment</u>

Model Name : Claude 3.0 (Find the model Opus,Sonnet or Haiku.

API Key : sk-ant-api03-
DHDjAaU5fq0nRLNmeZ6SI561jOa9VTBOtbsGFNImCHabeRDhxUMTl7_Ui7G81CfrvaR1DjA6p9g4SFrrBe86
AA-HGEG4wAA

Comprehensive Plan No (1) to Execute the assignment:

To complete this project, I'll need a comprehensive plan that includes both frontend and backend development. Below is a roadmap outlining the tasks, tools, and codes needed for each step:

Frontend:

1. **Create Entry Point and WhatsApp Integration:**
    - Tools: Twilio API for WhatsApp integration, QR Code Generator.
    - Code: HTML/CSS for QR code, JavaScript for Twilio integration.

Backend:

2. **Choose LLM Model and WhatsApp API Integration:**
    - Tools: WhatsApp Business API, Hugging Face Transformers library for LLM.
    - Code: Python for backend logic.
3. **Set Up Database:**
    - Tools: MongoDB for NoSQL database.
    - Code: Define schema and CRUD operations.
4. **Populate Restaurants and User Data:**
    - Code: Write scripts to add restaurant and user data to the database.
5. **Identify New or Returning Users:**
    - Code: Check database for existing users.
6. **Collect User Data for New Users:**
    - Code: Capture user inputs via WhatsApp and store in the database.
7. **Handle User Orders:**
    - Code: Process user requests, filter dishes, and handle selections.
8. **Handle Latency and User Interaction:**
    - Code: Implement timeouts, prompts, and dynamic responses.
9. **Handle Order Confirmation and Editing:**
    - Code: Manage user responses for order confirmation or editing.
10. **Generate QR Code for Payment:**
    - Tools: UPI API for payment, QR Code generation library.
    - Code: Generate QR code with bill amount linked to bank account.

11. **Track UPI Acknowledgement:**
    - Code: Listen for payment status updates from UPI API.
12. **Generate PDF Invoice:**
    - Tools: PDF generation library.
    - Code: Generate invoice with order details and send to user.
13. **Track and Store Order Data:**
    - Code: Store order details in the database.

Additional Features:

14. **Dashboard Development:**
    - Tools: Web framework (Flask, Django), Charting library (Plotly, D3.js).
    - Code: Create a web dashboard to view user, order, and restaurant data.

Model and API Keys:

15. **LLM Bot Integration:**
    - Model: Claude 3.0 from Hugging Face.
    - API Keys: Hugging Face API for model access.

Workflow:

- User scans QR code to start a chat session on WhatsApp.
- LLM Bot interacts with the user in Hindi, English, or Hinglish.
- User provides preferences and selects dishes.
- Backend processes user input, filters dishes, and manages orders.
- Payment is initiated via QR code, and UPI acknowledgement is tracked.
- Invoice is generated and sent to the user.
- Order details are stored in the database.
- Dashboard provides an overview of user, order, and restaurant data.

Notes:

- Ensure security measures for user data and payment transactions.
- Test thoroughly for different user scenarios and edge cases.
- Continuously monitor and improve the system based on user feedback and usage patterns.

Break down the roadmap into a step-by-step guideline for plan (1):

Step 1: Setup

1. **Setup Environment:**
    - Set up your development environment with necessary tools and libraries.
2. **Create Entry Point:**
    - Generate a QR code for users to initiate the chat session.
    - Use Twilio API for WhatsApp integration.

Step 2: Backend Development

3. **Choose LLM Model:**
    - Select the Claude 3.0 model from Hugging Face.
    - Get API keys for model access.
4. **Set Up Database:**
    - Choose MongoDB for NoSQL database.
    - Define schema for user, restaurant, and order data.
    - Implement CRUD operations.
5. **Populate Database:**
    - Add restaurant and user data to the database.
6. **Handle User Sessions:**
    - Identify new or returning users.
    - Collect user data for new users.
7. **Handle User Orders:**
    - Process user requests and filter dishes based on preferences.
    - Implement logic to handle dish selection and ordering.
8. **Handle User Interaction:**
    - Manage latency and user responses.
    - Prompt users for confirmation or further actions.
9. **Generate QR Code for Payment:**
    - Integrate UPI API for payment.
    - Generate QR code with bill amount linked to bank account.
10. **Track UPI Acknowledgement:**
    - Listen for payment status updates from UPI API.
    - Handle success or failure of payment.
11. **Generate PDF Invoice:**
    - Use a PDF generation library to create an invoice.
    - Send the invoice to the user.
12. **Track and Store Order Data:**
    - Store order details in the database.

Step 3: Frontend Integration

13. **WhatsApp Integration:**
    - Integrate Twilio API for WhatsApp.
    - Implement frontend logic to handle user interactions.

Step 4: Additional Features

14. **Dashboard Development:**
    - Choose a web framework (e.g., Flask, Django).

    - Create a dashboard to view user, order, and restaurant data.
    - Use a charting library for data visualization.

Step 5: Testing and Deployment

15. **Testing:**
    - Test the system for different user scenarios and edge cases.
    - Ensure security measures for user data and payment transactions.
16. **Deployment:**
    - Deploy the system on a server.
    - Continuously monitor and improve the system based on user feedback and usage patterns.

Notes:

- Ensure thorough testing of each component and integration points.
- Maintain code quality and security throughout the development process.
- Document the system architecture, APIs, and deployment process for future reference.
- Collaborate with designers for UI/UX design if needed.

Comprehensive Plan No (2) to Execute the assignment:

Roadmap

1. **Entry Point**: Create a QR code that users can scan to initiate a WhatsApp chat session with your bot.
2. **WhatsApp Integration**: Integrate your bot with the WhatsApp API to enable communication via the chat interface. You can use libraries like `whatsapp-web.js` for Node.js or the official WhatsApp Cloud API.
3. **Language Model Selection**: Choose a suitable language model (LLM) that can understand and respond in Hindi, English, and Hinglish. You can explore options like GPT-3, Claude, or other multilingual models.
4. **Backend Development**:
    - Set up a backend server using a language like Node.js or Python.
    - Integrate the chosen LLM with your backend through APIs or libraries.
    - Connect the backend to a database (SQL or NoSQL) to store restaurant data, user data, and order information.
    - Implement the necessary logic to perform CRUD operations on the database based on user inputs and LLM responses.
5. **Data Preparation**:
    - Create a database table/collection for restaurants with fields like name, menu, portion sizes, prices, images, addresses, and Google PIN locations.
    - Create a database table/collection for users with fields like name, date of birth, email, addresses, food preferences, allergies, and health conditions.
    - Populate the database with sample data for restaurants and users.
6. **User Management**:
    - Implement logic to identify new users and returning users based on their WhatsApp number or other identifiers.
    - For new users, collect the required information (name, preferences, etc.) and store it in the user table/collection.
7. **Order Processing**:
    - Develop logic to process user inputs (text or image replies) and extract relevant information like dish names, quantities, and preferences.
    - Implement algorithms to filter and sort dishes based on user preferences, distance from the user, and other criteria.
    - Generate grids or carousels to display dish information (images, names, prices, etc.) to the user.
    - Handle user interactions like selecting dishes, modifying orders, and confirming orders.
    - Integrate with a payment gateway (e.g., UPI) to process orders and generate invoices.
8. **Dashboard**: Create a dashboard or admin interface to view and manage user data, restaurant data, and order information.

Required Tools

- **Programming Language**: Node.js or Python

- **Web Framework**: Express.js (for Node.js) or Flask (for Python)
- **Database**: SQL (e.g., MySQL, PostgreSQL) or NoSQL (e.g., MongoDB)
- **WhatsApp Integration Library**: `whatsapp-web.js` (for Node.js) or `python-whatsapp-web.js` (for Python)
- **Language Model API**: API key and integration library for the chosen LLM (e.g., GPT-3, Claude)
- **Payment Gateway Integration**: Library or SDK for the chosen payment gateway (e.g., UPI)
- **Web Server**: Nginx or Apache (for hosting the backend server)
- **QR Code Generation Library**: e.g., `qrcode` (for Node.js) or `qrcode` (for Python)

Code Structure:

```
project/
├── server/
│   ├── app.js (or app.py)
│   ├── routes/
│   │   ├── whatsapp.js (or whatsapp.py)
│   │   ├── restaurant.js (or restaurant.py)
│   │   ├── user.js (or user.py)
│   │   └── order.js (or order.py)
│   ├── controllers/
│   │   ├── whatsapp.js (or whatsapp.py)
│   │   ├── restaurant.js (or restaurant.py)
│   │   ├── user.js (or user.py)
│   │   └── order.js (or order.py)
│   ├── models/
│   │   ├── restaurant.js (or restaurant.py)
│   │   ├── user.js (or user.py)
│   │   └── order.js (or order.py)
│   ├── utils/
│   │   ├── llm.js (or llm.py)
│   │   ├── payment.js (or payment.py)
│   │   └── qrcode.js (or qrcode.py)
│   └── config.js (or config.py)
├── client/
│   ├── index.html
│   ├── css/
│   │   └── styles.css
│   └── js/
│       └── app.js
├── database/
│   └── schema.sql (or schema.js for NoSQL)
└── README.md
```