Project Assignment

Integrate our Llama-Based AI Model with WhatsApp Cloud API, User Management & Stripe Payment

---

1. Overview & Goals

We want to offer a specialized, paid WhatsApp chat service using our Llama-based AI model. Users will message us on WhatsApp, then receive a prompt to pay if they are not yet authorized. Once they've successfully paid through Stripe, they gain access to the AI-driven conversation (the "role-play experience").

Key objectives:

1. **WhatsApp Cloud API Integration**: Connect our AI Model with WhatsApp so users can send messages to our official number and receive AI-generated replies.
2. **User Management**: Implement a simple system that tracks user identities (linked to their WhatsApp phone number) and whether they have paid or not.
3. **Stripe Payment**: Provide a link to a Stripe checkout page. After successful payment, mark the user as "paid" in our system, granting them full access to the AI bot.

---

2. Technical Requirements

1. **WhatsApp Cloud API Setup**
    - Use Meta's official WhatsApp Cloud API with our registered business number.
    - Configure a Webhook endpoint (e.g., in Flask) to receive incoming WhatsApp messages.
    - Implement message sending so our bot can respond to the user's WhatsApp messages.
2. **User Management**
    - Each user is identified by their **WhatsApp phone number** or a unique user ID in the database.
    - We need a simple database (can be SQL or NoSQL) to store:
        - user_id (the phone number or an internal identifier)
        - payment_status (e.g., "paid" or "unpaid")

        - other relevant info (e.g., name, creation date, etc.)

3. **Stripe Payment Flow**
    - For **unpaid** users, the bot replies with a message containing a link to the Stripe Checkout page.
    - Implement the Stripe Checkout so they can pay for a subscription or one-time purchase (to be determined).
    - After successful payment, Stripe should send a **Webhook** event (e.g., payment_intent.succeeded or checkout.session.completed).
    - At that point, our backend updates the user record to payment_status = "paid".
    - Now the user can access the AI conversation.
4. **Flow**

1. **User sends a WhatsApp message** → Our Cloud API endpoint receives it.
2. We check in our database whether the user is paid or not.
3. If unpaid, the system sends a reply like "You need to subscribe. Please follow this link: [StripeCheckoutURL]."
4. Once the user pays successfully, we receive the Stripe webhook, update payment_status = "paid".
5. For subsequent messages, the user can then engage with the AI fully (the AI model is available to paid users only).

5. **AI Integration (Llama Model)**
    - The Llama model is already running in our environment (VPS + Flask + RAG).
    - After a user is confirmed as paid, their WhatsApp messages are relayed to the Llama-based system.
    - The AI's response is then sent back via the WhatsApp Cloud API.
    - Please ensure the overall pipeline remains stable and responds within acceptable time limits for WhatsApp.

---

3. Implementation Steps

1. **WhatsApp Cloud API Setup**

- Obtain the WhatsApp Business credentials (Phone Number ID, WhatsApp Business Account ID, Access Token).
- Configure the required Webhook endpoint in our Flask app, and set the Callback URL in Meta's Developer Dashboard.
- Make sure we can parse incoming messages (phone number, message text).

2. **Basic Database & User Model**
   - Create a users table/collection with fields:
     - id (primary key)
     - phone_number (unique)
     - payment_status (string or boolean)
     - optional fields...
   - Provide functions for creating/updating user records:
     - get_or_create_user(phone_number)
     - update_payment_status(user_id, status)
3. **Stripe Integration**
   - Create a Product/Price in Stripe (either subscription or one-time).
   - Implement a route (e.g., /create-checkout-session) that generates the Stripe Checkout session link, returning it to the user.
   - Create a Stripe Webhook endpoint (e.g., /stripe-webhook) to handle checkout.session.completed events.
     - On success, update the user's payment_status = "paid".
4. **Message Flow in Flask**
   - In the WhatsApp Webhook route (e.g., /whatsapp-webhook):

1. Parse the incoming JSON to get the user's phone number and message text.
2. Check if user exists in DB (if not, create them with payment_status="unpaid").
3. If payment_status="unpaid", respond with "Please subscribe. Here's your payment link."
4. If payment_status="paid", pass the message to the Llama-based AI function.
   - Return the AI's response to the user via the WhatsApp Cloud API.

5. **Logging & Error Handling**
    - Log inbound/outbound WhatsApp messages (for debugging).
    - Ensure any failure (e.g., Stripe webhook fails to update user) is captured in logs.
6. **Optional**: Send Additional Info (images, etc.) if needed. Start with text messages only.

---

4. Acceptance Criteria

1. **End-to-End Payment Flow**:
    - User messages WhatsApp → Receives payment link → Successfully pays → Payment status updated → Gains access to the AI.
2. **Valid Webhook Integration**:
    - We can see in logs that when user completes Stripe checkout, the system updates their status automatically.
3. **AI Responses**:
    - Paid users' messages are forwarded to the Llama-based system, and the system's responses appear on WhatsApp.
4. **Maintainable Code**:
    - Clear separation of concerns (WhatsApp webhook, payment, AI logic, DB interactions).
    - Adequate error handling and logging.

---

5. Timeline & Deliverables

1. **WhatsApp Cloud API + Basic DB**
    - Set up Cloud API credentials, first simple webhook route, DB for users
2. **Stripe Checkout**
    - Implement route to create a session, link to user, test the flow
3. **Stripe Webhook**
    - Update user's payment_status after payment completes
4. **Integration & Testing**

- Ensure the message flow to Llama-based AI is triggered only for paid users
- Conduct final test runs with real or sandbox environment

Additional Notes

You get 1 week for delivery this project.