Привет. нужно написать код для приложения с фастапи которое будет принимать входящее сообщение, обрабатывать его и возвращать ответ.

###Логика обработки входящих сообщений:

# у каждого пользователя будет свой идентификатор который будет приходить вместе с сообщением.

# По каждому пользователяю нужно собирать и хрнаить историю всех сообщений и ответов + метаданные в которых будет указываться стадия общения (все общение идет по сценарию с разными стадиями, по типу воронки, стадии сменяются только вперед, обратной смены нет).

# Как только пришло новое сообщение нужно отправить это сообщение и историю сообщений в ИИ-модель, которая будет определять на какой стадии находится диалог.

# ИИ модель возвращает название стадии диалога и стадия проставляется в метаданных истории пользователя.

# Далее исходя из стадии в метаданных сообщение пользователя и стория перписки отправляется в одну из функций для отправки сообщений в ИИ (количество функций равно количеству этапов и в каждой фукнции свой промпт характерный для опредленной стадии).

# Функция с ИИ возвращает ответ, который сохраняется в историю сообщений и отправляется пользователю.

###СТадии общения и их описание:

1. Приветственное сообщение: начальное общение в чате, на этом этапе ставим в метаданных исторрии общения стадию общения == "первичное общение". тут происходит запрос фотографии для оценки, объяснение подхода к расчету стоимости. Если клиент прислал фотографию, то меняем в метаданных истории сообщений стадию общения == "выбор стиля"

2. СТадия Выбор стиля. Присылаем примеры работ: демонстрация различных стилей портретов для выбора клиентом. Клиент выбирает стиль: подтверждение выбранного стиля, предложение видео с демонстрацией портрета. Если клиент выбрал стиль, то меняем стадию в метаданных на "выбор сроков и города"

3. Стадия "выбор сроков и города". Уточняем срок и город клиента: проверка возможности изготовления к нужной дате и организации получения. Если города клиента нет в списке мастерских партнёров , сообщить, что в его городе есть пункт выдачи готовых портретов. как только срок и город согласованы то меняем стадию на "Расчет стоимости"

4. стадию "Расчет стоимости". Расчет стоимости: предоставление цен по регионам и техникам с учетом дополнительных лиц, объяснение условий оплаты. Клиент выбирает размер: подтверждение выбора размера,. Как только размер и все детали исполнения выбраны клиентом, то переводим в стадию "отправка реквизитов"

5. СТадия "отправка реквизитов". отправляем реквизитов для предоплаты. как только отправили переводим в стадию "предложение дополнительных услуг"

6. стадия "Предложение дополнительных услуг": обсуждение техники масла, акрила, акрилового и масляного фонов с их особенностями и дополнительной стоимостью. как только предложили и получили ответ меняем стадию на "предложение премиум холста"

7. стадия Предложение премиум холста: информирование о новой опции с объяснением преимуществ и дополнительной стоимостью. как только получили ответ меняем стадию на "предложение акции 1+1=3"

8. стадия Предложение акционной акции "1+1=3": мотивация заказать второй портрет с бонусом третьего бесплатно. как только получили ответ меняем стадию в метаданных на "заключительное общение"

стадия Заключительное сообщение: подтверждение получения предоплаты, информация о дальнейшем сопровождении заказа, установление новых контактов.

###Каркас кода:###

###Main.py  
  
from fastapi import FastAPI  
from .models import MessageIn  
from .storage import load\_histories, save\_histories  
from .logic import handle\_message  
  
app = FastAPI()  
  
  
@app.post("/message")  
async def process\_message(msg: MessageIn):  
 histories = load\_histories()  
  
 # Если пользователь новый  
 if msg.user\_id not in histories:  
 histories[msg.user\_id] = {  
 "messages": [],  
 "metadata": {"stage": "первичное общение"}  
 }  
  
 history = histories[msg.user\_id]["messages"]  
 metadata = histories[msg.user\_id]["metadata"]  
  
 # Сохраняем входящее сообщение  
 history.append({"role": "user", "content": msg.message})  
  
 # Обрабатываем сообщение через state machine  
 result = handle\_message(metadata["stage"], history, msg.message)  
  
 # Обновляем стадию  
 metadata["stage"] = result["next\_stage"]  
  
 # Сохраняем ответ  
 history.append({"role": "assistant", "content": result["reply"]})  
  
 save\_histories(histories)  
  
 return {  
 "reply": result["reply"],  
 "stage": metadata["stage"],  
 "history": history  
 }

###logic.py  
  
from typing import Dict, List  
from .stages import STAGES  
from . import ai\_client  
  
  
def handle\_message(current\_stage: str, history: List[Dict[str, str]], message: str) -> Dict:  
 from . import ai\_client  
  
 stage\_handlers = {  
 "первичное общение": ai\_client.handle\_stage\_primary,  
 "выбор стиля": ai\_client.handle\_stage\_style,  
 "выбор сроков и города": ai\_client.handle\_stage\_city,  
 "Расчет стоимости": ai\_client.handle\_stage\_cost,  
 "отправка реквизитов": ai\_client.handle\_stage\_payment,  
 "предложение дополнительных услуг": ai\_client.handle\_stage\_additional,  
 "предложение премиум холста": ai\_client.handle\_stage\_premium,  
 "предложение акции 1+1=3": ai\_client.handle\_stage\_promo,  
 "заключительное общение": ai\_client.handle\_stage\_final,  
 }  
  
 handler = stage\_handlers.get(current\_stage)  
 if not handler:  
 return {"reply": "Неизвестная стадия", "next\_stage": current\_stage}  
  
 result = handler(history, message)  
  
 reply = result["reply"]  
 ready\_to\_next = result.get("ready\_to\_next", False)  
 force\_next = result.get("force\_next\_stage")  
  
 next\_stage = current\_stage  
  
 # Явный форсированный переход (например, пришло фото на первом этапе)  
 if force\_next:  
 next\_stage = force\_next  
 elif ready\_to\_next and current\_stage in STAGES:  
 idx = STAGES.index(current\_stage)  
 if idx + 1 < len(STAGES):  
 next\_stage = STAGES[idx + 1]  
  
 return {"reply": reply, "next\_stage": next\_stage}

###models.py  
  
from pydantic import BaseModel  
from typing import List, Dict, Any  
  
  
class MessageIn(BaseModel):  
 user\_id: str  
 message: str  
  
  
class MessageRecord(BaseModel):  
 role: str # "user" | "assistant"  
 content: str  
  
  
class UserHistory(BaseModel):  
 messages: List[MessageRecord]  
 metadata: Dict[str, Any]

###stages.py  
  
STAGES = [  
 "первичное общение",  
 "выбор стиля",  
 "выбор сроков и города",  
 "Расчет стоимости",  
 "отправка реквизитов",  
 "предложение дополнительных услуг",  
 "предложение премиум холста",  
 "предложение акции 1+1=3",  
 "заключительное общение"  
]

###storage.py  
  
import json  
import os  
from typing import Dict, Any  
  
DATA\_FILE = "histories.json"  
  
  
def load\_histories() -> Dict[str, Any]:  
 if not os.path.exists(DATA\_FILE):  
 return {}  
 with open(DATA\_FILE, "r", encoding="utf-8") as f:  
 return json.load(f)  
  
  
def save\_histories(data: Dict[str, Any]):  
 with open(DATA\_FILE, "w", encoding="utf-8") as f:  
 json.dump(data, f, ensure\_ascii=False, indent=2)

###ai\_client.py  
from typing import List, Dict  
  
# Заглушка для вызова LLM (сюда можно подключить OpenAI / DeepSeek и т.п.)  
def call\_llm(prompt: str, history: List[Dict[str, str]]) -> str:  
 return f"(ИИ ответ): {prompt}"  
  
  
# ---- Определение стадии ----  
def detect\_stage(current\_stage: str, history: List[Dict[str, str]], new\_message: str) -> str:  
 *"""*  
 *Определение стадии по истории и новому сообщению.*  
 *В реальном случае сюда подключается LLM для классификации.*  
 *"""*  
return current\_stage # здесь оставляем как заглушку  
  
  
# ---- Функции стадий ----  
def handle\_stage\_primary(history, message: str) -> Dict:  
 prompt = """Ты — Дмитрий - виртуальный менеджер студии портретов на холсте.   
 Ты общаешься с клиентом в чате на стадии "Приветственное сообщение".   
   
 ⚠️ Основная цель на этом этапе — получить фотографию от клиента.   
 Пока фото не получено, ты должен работать с любыми возражениями или вопросами   
 и мягко возвращать диалог к необходимости отправить фотографию.   
   
 Используй только заранее подготовленные скрипты. Ничего не перефразируй, не придумывай новое.   
 Выбирай вариант, который соответствует ситуации.   
   
 Скрипты:  
   
 1. Если клиент прислал фотографию:  
 "Здравствуйте, меня зовут Дмитрий, я старший менеджер, буду помогать Вам по всем вопросам. Для Вашей фотографии отлично подойдет стиль художественная живопись, с имитацией техники масляных мазков - спокойный лаконичный стиль.   
 Сейчас пришлю для Вас примеры наших актуальных работ, посмотрите, что больше понравится и уже определитесь финально, а также оцените детализацию и качество отрисовки, далее обязательно сориентирую Вас по стоимости и нашим размерам."  
   
 2. Если клиент не прислал фотографию:  
 "Здравствуйте, меня зовут Дмитрий, я старший менеджер, буду помогать Вам по всем вопросам. Для расчета по стоимости мне нужна Ваша фотография, чтобы подобрать для Вас стилистику. Наши художники рисуют на графических планшетах вручную, не используют обработку фильтрами и чтобы оценить масштаб работы мне нужна Ваша фотография. Сможете прислать в чат?"  
   
 3. Если клиент сразу спрашивает цену:  
 "Здравствуйте, меня зовут Дмитрий, я старший менеджер, буду помогать Вам по всем вопросам. С радостью сориентирую Вас по стоимости, но мне необходимо понимать масштаб работы, для этого мы запрашиваем фотографию, далее задам пару вопросов, это не займет более 5 минут и вышлю для Вас стоимость и размеры. Просто мы не используем обработку фильтрами, наши художники рисуют на графических планшетах, поэтому прайс сразу не высылается. Кого бы Вы хотели изобразить на портрете?"  
   
 Формат ответа: JSON  
 {  
 "reply": "Здравствуйте, меня зовут Дмитрий, я старший менеджер, буду помогать Вам по всем вопросам. Для расчета по стоимости мне нужна Ваша фотография, чтобы подобрать для Вас стилистику. Наши художники рисуют на графических планшетах вручную, не используют обработку фильтрами и чтобы оценить масштаб работы мне нужна Ваша фотография. Сможете прислать в чат?",  
 "ready\_to\_next": true|false  
 }  
   
 Правила:  
 - reply = текст для сообщения   
 - ready\_to\_next = true, только если клиент уже прислал фотографию (условие выполнено и можно переходить к следующей стадии).   
 - ready\_to\_next = false, если фото ещё не получено — даже если клиент задаёт другие вопросы. В этом случае всегда используй скрипты так, чтобы возвращать диалог к необходимости фото.  
 """  
 response = call\_llm(prompt, history)  
  
 # Если клиент прислал фото (заглушка: проверяем слово 'фото')  
 ready\_to\_next = "фото" in message.lower()  
  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_style(history, message: str) -> Dict:  
 prompt = "Стадия 'Выбор стиля'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "стиль" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_city(history, message: str) -> Dict:  
 prompt = "Стадия 'Выбор сроков и города'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "город" in message.lower() and "срок" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_cost(history, message: str) -> Dict:  
 prompt = "Стадия 'Расчет стоимости'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "размер" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_payment(history, message: str) -> Dict:  
 prompt = "Стадия 'Отправка реквизитов'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "оплатил" in message.lower() or "перевел" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_additional(history, message: str) -> Dict:  
 prompt = "Стадия 'Предложение дополнительных услуг'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "нет" in message.lower() or "да" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_premium(history, message: str) -> Dict:  
 prompt = "Стадия 'Предложение премиум холста'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "нет" in message.lower() or "да" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_promo(history, message: str) -> Dict:  
 prompt = "Стадия 'Акция 1+1=3'."  
 response = call\_llm(prompt, history)  
  
 ready\_to\_next = "нет" in message.lower() or "да" in message.lower()  
 return {"reply": response, "ready\_to\_next": ready\_to\_next}  
  
  
def handle\_stage\_final(history, message: str) -> Dict:  
 prompt = "Стадия 'Заключительное общение'."  
 response = call\_llm(prompt, history)  
  
 return {"reply": response, "ready\_to\_next": False}