ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

ACADEMIC YEAR – 2024-2025

MINI PROJECT: AI BASED TOURISM RECOMMENDATION AND PLANNER

CODE:

1. **Main.py:**

```
from fastapi import FastAPI
from typing import Dict
from fpdf import FPDF
import google.generativeai as genai
from app.payments.payments import create_payment_intent

app = FastAPI()

# ✅ Configure Gemini API
GEMINI_API_KEY = "AIzaSyAjOZL82IBO7q-rtySTOMNMM5Ez91aohzY"
genai.configure(api_key=GEMINI_API_KEY)

# ✅ Store user sessions & PDFs
user_sessions: Dict[str, Dict] = {}
download_links: Dict[str, str] = {}

# ✅ Function to call Gemini API for travel insights
def get_gemini_response(prompt: str):
  try:
    model = genai.GenerativeModel("gemini-1.5-pro")
    response = model.generate_content(prompt)
    return response.text.strip()
  except Exception as e:
    print(f"Gemini API Error: {e}")
    return "Sorry, I'm unable to process your request at the moment."

# ✅ PDF Generation Function
def generate_pdf(data: dict):
  pdf = FPDF()
  pdf.set_auto_page_break(auto=True, margin=15)
  pdf.add_page()
  pdf.set_font("Arial", style="B", size=16)
  pdf.cell(200, 10, f"Trip Itinerary for {data['destination']}", ln=True, align="C")

  pdf.set_font("Arial", size=12)
  pdf.ln(10)

  pdf.cell(200, 10, f"Budget Range: {data['budget']}", ln=True)
  pdf.cell(200, 10, f"Number of Adults: {data['adults']}", ln=True)
  pdf.cell(200, 10, f"Number of Children: {data['children']}", ln=True)
  pdf.cell(200, 10, f"Duration: {data['days']} days", ln=True)
  pdf.cell(200, 10, f"Interests: {data['interests']}", ln=True)
```

```
    # 🔥 Gemini API Analysis for Budget Breakdown & Sightseeing
    travel_insights = get_gemini_response(f"Give me a detailed budget breakdown and top sightseeing spots for a trip to {data['destination']} with a budget of {data['budget']}")

    pdf.ln(10)
    pdf.multi_cell(0, 10, travel_insights)  # AI-generated content

    pdf_path = f"trip_itinerary_{data['destination'].replace(' ', '_')}.pdf"
    pdf.output(pdf_path)
    return pdf_path

# ✅ Chatbot Logic
def get_chatbot_response(user_id: str, user_input: str):
    user_input = user_input.lower()

    if user_id not in user_sessions:
        user_sessions[user_id] = {"stage": "destination", "data": {}}

    session = user_sessions[user_id]
    stage = session["stage"]

    if stage == "destination":
        session["data"]["destination"] = user_input
        session["stage"] = "budget"
        return "Great choice! What's your budget range for the trip?"

    elif stage == "budget":
        session["data"]["budget"] = user_input
        session["stage"] = "adults"
        return "Got it! How many adults are traveling?"

    elif stage == "adults":
        session["data"]["adults"] = user_input
        session["stage"] = "children"
        return "Thanks! How many children will be joining?"

    elif stage == "children":
        session["data"]["children"] = user_input
        session["stage"] = "days"
        return "Noted! How many days will the trip last?"

    elif stage == "days":
        session["data"]["days"] = user_input
        session["stage"] = "interests"
        return "Awesome! What kind of activities or experiences interest you? (e.g., sightseeing, adventure, food)"

    elif stage == "interests":
        session["data"]["interests"] = user_input
        session["stage"] = "complete"
        return "Thank you! We have all the details now. Would you like to generate a PDF itinerary? (yes/no)"

    elif stage == "complete":
        if "yes" in user_input:
            pdf_path = generate_pdf(session["data"])  # Generate PDF with AI data
            download_links[user_id] = pdf_path
            session["stage"] = "payment"
            return f"Your itinerary is ready! Download here: {pdf_path}. Type 'pay' to proceed with the payment."
        else:
            session["stage"] = "destination"
            return "Would you like a different plan for the same location or a completely new one? (same/new)"

    elif stage == "payment":
        if "pay" in user_input:
            payment_data = create_payment_intent(amount=5000)  # Example ₹50.00
            if "payment_url" in payment_data:
                session["stage"] = "paid"
                return f"Payment link generated! Complete your payment here: {payment_data['payment_url']}"
            else:
                return "Payment failed. Please try again."

    elif stage == "paid":
        return f"Payment confirmed! Download your itinerary here: {download_links.get(user_id)}"

    return get_gemini_response(user_input)  # Fallback AI response

# ✅ API Endpoints
@app.get("/chat")
async def chat(user_id: str, prompt: str):
    response = get_chatbot_response(user_id, prompt)
    return {"response": response}

@app.post("/payment", include_in_schema=True)
async def process_payment(data: dict):
    return {"message": "Payment received"}

@app.get("/")
async def root():
    return {"message": "Welcome to the chatbot API! Use /chat?user_id=<user_id>&prompt=<your_message> to
chat."}

#run the app using uvicorn app.main2:app --reload
```

2. Chatbot.py:

```python
import google.generativeai as genai

# Hardcoded API key
gemini_api_key = "AIzaSyAjOZL82IBO7q-rtySTOMNMM5Ez91aohzY"

# Configure Gemini API
genai.configure(api_key=gemini_api_key)

# Function to call Gemini API
def get_gemini_response(prompt: str):
    try:
        model = genai.GenerativeModel("gemini-1.5-pro")
        response = model.generate_content(prompt)
        return response.text.strip()
    except Exception as e:
        print(f"Gemini API Error: {e}")
        return "Sorry, I'm unable to process your request at the moment."
```

3. Payments.py:

```python
import stripe

# Set your secret key directly (not recommended for production)
stripe.api_key =
"sk_test_51R9KNP3zcDeHgkU0ozizyyUkG2xudrSvGOWIFdwNhH7jyIrFRql0Mm8jKleVwkel5HAtM00hT8TSnFcf
U5eh22XQ00IFtp7w83"

def create_payment_intent(amount, currency="inr"):
    try:
        amount_in_paisa = int(amount * 100)  # Convert INR to paisa (Stripe requires
smallest unit)
        intent = stripe.PaymentIntent.create(
            amount=amount_in_paisa,
            currency=currency,
            payment_method_types=["card"]
        )
        return {"client_secret": intent.client_secret}
    except stripe.error.StripeError as e:
        return {"error": str(e)}
    except Exception as e:
        return {"error": f"Unexpected error: {str(e)}"}
```

4. Import_google.py

```python
import google.generativeai as genai

genai.configure(api_key="Insert_your_API_key")

models = genai.list_models()
for model in models:
    print(model.name)
```