Documentation: Agent Flow and Prompt Design for Credit Card Recommendation System

June 19, 2025

Overview

This documentation outlines the agent flow and prompt design for the Credit Card Recommendation System, developed as of 11:46 AM IST on Thursday, June 19, 2025. The system employs a Groq agent to handle intelligent question-and-answer (Q&A) interactions, integrated with a Streamlit frontend, Flask backend, PostgreSQL database, and a recommendation engine. The agent flow defines the sequence of operations, while the prompt design ensures effective communication between the user and the agent.

Agent Flow

The agent flow follows a structured process to gather user data and deliver recommendations:

Initialization

- **Trigger**: User clicks "Start" on the Streamlit frontend.
- **Action**: The frontend sends a `POST /start_session` request to the Flask backend.
- **Agent Role**: The Groq agent receives a call to `start_session()` via the backend, generating a unique session ID and the first question (e.g., "What is your monthly income in INR?").
- **Output**: Returns a JSON response (`{"session_id": "...", "question": "..."}`) to the frontend for display.

Q&A Loop

- **Trigger**: User enters an answer.
- **Action**: The frontend submits a `POST /submit_answer` request with `{"session_id": "...", "answer": "..."}` to the backend.
- **Agent Role**:
  - Processes the input via `process_answer(session_id, answer)`.

  - Validates the answer (e.g., checks for numeric input or positive values).
  - If invalid, generates an error message (e.g., "Please enter a valid income amount.") and increments the step.
  - If valid, stores the answer, increments the step, and formulates the next question based on the context.

- **Output**: Returns the next question or error message to the frontend.
- **Iteration**: Repeats for 8 questions, ending with "Ready to get recommendations?".

Recommendation Trigger

- **Trigger**: User responds to the final question.
- **Action**: The frontend sends a `POST /get_recommendations` request with `{"session_id": "..."}`.
- **Agent Role**: The agent's role concludes here, as the backend handles data retrieval and passes it to the recommendation engine. However, the agent's collected data is critical for this phase.

Error Handling

- **Condition**: Invalid user input detected.
- **Action**: The agent generates a context-specific error message via Groq and prompts the user to re-enter the answer.
- **Output**: Error message returned to the frontend for display.

Prompt Design

The prompt design ensures the Groq agent delivers clear, context-aware, and user-friendly interactions.

Initial Prompt (Session Start)

- **Purpose**: Initialize the session and pose the first question.
- **Text**: "You are an intelligent assistant for a credit card recommendation system. Start a new session and ask the user the first question to gather their financial profile. Begin with: 'What is your monthly income in INR?' Provide a session ID and format the response as JSON with 'session_id' and 'question' keys."
- **Example Output**: `{"session_id": "abc123", "question": "What is your monthly income in INR?"}`

Question Prompt (Q&A Loop)

- **Purpose**: Process user answers and generate subsequent questions.
- **Text**: "You are an intelligent assistant for a credit card recommendation system. You have a session with ID [session_id]. The user provided the answer '[answer]' to the previous question '[previous_question]'. Validate the answer (ensure it is a positive number for income, for example). If invalid, return an error message in JSON with 'error' key and increment the step. If valid, store the answer, increment the step, and ask the next question based on the context (e.g., spending habits, credit score). Use JSON format with 'session_id', 'step', and 'question' or 'error' keys. Questions should align with a sequence of 8 total, ending with 'Ready to get recommendations?'"
- **Example Output (Valid)**: `{"session_id": "abc123", "step": 2, "question": "What are your average monthly expenses in INR?"}`
- **Example Output (Invalid)**: `{"session_id": "abc123", "step": 1, "error": "Please enter a valid income amount."}`

Contextual Adaptation

- The agent adapts questions based on prior answers (e.g., if income is low, it may prioritize cards with no annual fees).
- The sequence includes: income, expenses, credit score, debt, spending categories, preferred rewards, employment status, and readiness confirmation.

Error Prompt

- **Purpose**: Handle invalid inputs gracefully.
- **Text**: "If the user's answer '[answer]' to '[previous_question]' is invalid (e.g., non-numeric for income), generate a clear error message. Increment the step and re-ask the question. Return JSON with 'session_id', 'step', and 'error' keys."
- **Example Output**: `{"session_id": "abc123", "step": 1, "error": "Invalid input. Please enter your monthly income in INR as a positive number."}`

Implementation Notes

- **Validation Logic**: The agent uses predefined rules (e.g., numeric checks, range limits) to validate answers, ensuring data integrity.
- **Step Management**: The `step` counter tracks progress through the 8-question sequence, stored and updated with each interaction.
- **Scalability**: The prompt design allows for future expansion of questions or criteria without altering the core flow.
- **Integration**: The agent interfaces with the backend via API calls, with data stored temporarily in session context for recommendation processing.