Intellihack NextGen

Task 03

Group Name: Cryptonovate

Contents

Question 03

Scenario:

You work for a Software development Company that develops chatbots using LLMs (Large Language Models). Smart Bank is a famous bank in Sri Lanka which gives customers different types of loans according to their eligibility. The institution has decided to implement a loan support system to assist customers with inquiries about their eligibility for loans and to provide information about the loan application process. As an AI Engineer of the company, you are asked you to develop a loan support chatbot for the Smart Bank. (Loan descriptions are provided.)

Functionality:

Loan Eligibility Check: The system should allow users to check their eligibility for different types of loans based on various criteria such as credit score, income level, employment status, and existing debts.

Loan Products Information: Users should be able to obtain information about the different types of loans offered by the institution, including their features, interest rates, repayment terms, and eligibility requirements.

Application Process Guidance: The system should guide users through the loan application process, providing information about the documents required, the steps involved, and the timeline for approval and disbursement.

FAQs and Troubleshooting: The system should provide answers to frequently asked questions about loans, such as how to improve credit score, what to do if an application is rejected, and how to calculate loan repayments.

Personalized Recommendations: Based on the user's financial situation, the system should offer personalized recommendations for suitable loan products and tips for improving eligibility.

Code

```
from dotenv import load_dotenv
load_dotenv()  # Optional, for Streamlit app configuration

import streamlit as st
import os
import google.generativeai as genai

os.environ["GOOGLE_API_KEY"] = "AIzaSyBpxppXt_LOWCOdykw4QDRmXtyxN76ING0"
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
```

```
class LoanSupportChatbot:
    def _init_(self):
        self.loans = {
            "Overdraft Facilities": {
                "description": "An overdraft is a way to manage cash flow with
interest. It's linked to your current account and lets you use extra funds up to
your limit. You can manage limits through online banking.",
                "features": [
                    "Quick and easy to arrange",
                    "Cash available when needed",
                    "Pay interest only on what you use",
                    "Sole proprietors can apply or adjust limits"
                ]
            },
            "Housing Loans": {
                "description": "We offer various housing loan schemes. Please
select a scheme for more information.",
                "schemes": {
                    "Housing Loan Scheme": {
                        "description": "Provides financing for buying land,
constructing a house, renovating, or purchasing a house/condominium.",
                        "quantum_of_loan": "Maximum of Rs. 50 Million",
                        "repayment_period": "Up to 25 years",
                        "eligibility": "Sri Lankan Citizen above 18, resident,
not a defaulter, various professions",
                        "security_collateral": "Mortgage over the property",
                        "conditions_apply": "Contact the nearest branch or call
centre for more information.",
                        "contact_info": "Please contact the relevant branch for
more information."
                    },
                    "Housing Loan for University Staff": {
                        "description": "Offers a special loan for university
staff with a minimum of 5 years of service.",
                        "quantum_of_loan": "Academic/Non-Academic Staff Grade:
Rs. 2.0 Million, Non-Staff Grade: Rs. 1.0 Million",
                        "conditions_apply": "Smart Bank holds the authority to
change or revise any condition."
                    },
                    "Housing Loan Scheme for Permanent Cadre Employees": {
                        "description": "Designed for Sri Lanka's permanent cadre
salaried employees.",
                        "repayment_period": "Maximum 25 years",
```

```
                        "security_collateral": "Primary mortgage over the
property to be developed",
                        "conditions_apply": "Smart Bank holds the authority to
change or revise any condition."
                    }
                }
            },
            # Add other loan types here...
        }


def get_response(question, chat):
    response = chat.send_message(question, stream=True)
    return response, chat


def main():
    """Streamlit Application"""

    st.set_page_config(page_title="Q&A and Loan Support Demo")

    st.header("Smart Bank Assistant")

    chatbot = LoanSupportChatbot()

    # Initialize session state for chat history and chat object if they don't
exist
    if 'chat_history' not in st.session_state:
        st.session_state['chat_history'] = []
    if 'chat' not in st.session_state:
        model = genai.GenerativeModel("gemini-pro")
        chat = model.start_chat(history=[])
        st.session_state["chat"] = chat

    user_input = st.text_input("Input: ", key="input")
    submit_button = st.button("Ask the question")

    if submit_button and user_input:
        # Get the chat object from session state
        chat = st.session_state['chat']

        # Get the response and updated chat object
       response, chat = get_response(user_input, chat)

        # Update the chat object in session state
```

```
        st.session_state['chat'] = chat

        # Process Gemini response
        st.subheader("The Response is")
        for chunk in response:
            st.write(chunk.text)
            st.session_state['chat_history'].append(("You", user_input))  # Add
user input to chat history
            st.session_state['chat_history'].append(("Bot (Gemini)", chunk.text))

        # Check if user query asks about loans
        if any(loan_type in user_input.lower() for loan_type in
chatbot.loans.keys()):
            for loan_type, loan_info in chatbot.loans.items():
                if loan_type in user_input.lower():
                    st.subheader(f"{loan_type.title()} Information:")
                    st.write(loan_info["description"])
                    if "features" in loan_info:
                        st.subheader("Features:")
                        for feature in loan_info["features"]:
                            st.write(f"- {feature}")
                    if "schemes" in loan_info:
                        st.subheader("Available Schemes:")
                        for scheme, scheme_info in loan_info["schemes"].items():
                            st.write(f"- {scheme}: {scheme_info['description']}")
                            if "quantum_of_loan" in scheme_info:
                                st.write(f"Quantum of Loan:
{scheme_info['quantum_of_loan']}")
                            if "repayment_period" in scheme_info:
                                st.write(f"Repayment Period:
{scheme_info['repayment_period']}")
                            if "eligibility" in scheme_info:
                                st.write(f"Eligibility:
{scheme_info['eligibility']}")
                            if "security_collateral" in scheme_info:
                                st.write(f"Security/Collateral:
{scheme_info['security_collateral']}")
                            if "conditions_apply" in scheme_info:
                                st.write(f"Conditions Apply:
{scheme_info['conditions_apply']}")
                    # Update chat history for loan-related interaction
                    st.session_state['chat_history'].append(("Bot (Loan
Support)", loan_info["description"]))
                    break
        else:
```

```
            st.write("Please specify a loan type to get information.")

        # Update chat history display
        st.subheader("The Chat History is")
        for role, text in st.session_state['chat_history']:
            st.write(f"{role}: {text}")


if _name_ == "_main_":
    main()
```

Code Explanation

This code is for developing a loan support chatbot using Streamlit and Google's GenerativeAI (Gemini) API. Let's break down the key components and functionality:

Environment Setup:

The code starts by importing necessary modules: dotenv, streamlit, os, and google.generativeai.

It sets up the Google API key required for using the GenerativeAI API.

LoanSupportChatbot Class:

Defines a class named LoanSupportChatbot that holds information about different types of loans offered by Smart Bank.

Each loan type has a description and may have additional features or schemes.

get_response Function:

This function is responsible for interacting with the GenerativeAI API (Gemini) to generate responses to user queries.

It takes a user question and a chat context, sends the question to the Gemini model, and returns the response.

Main Function (main):

This function is the core of the Streamlit application.

It sets up the Streamlit app layout, including the title and input elements.

Initializes the LoanSupportChatbot instance and session state for chat history and chat object.

Handles user input: when the user submits a question, it retrieves the chat object, sends the user input to Gemini for a response, and updates the chat history.

Processes Gemini's response, displaying it to the user.

If the user's query is related to loans, it extracts relevant loan information from the chatbot's data and displays it.

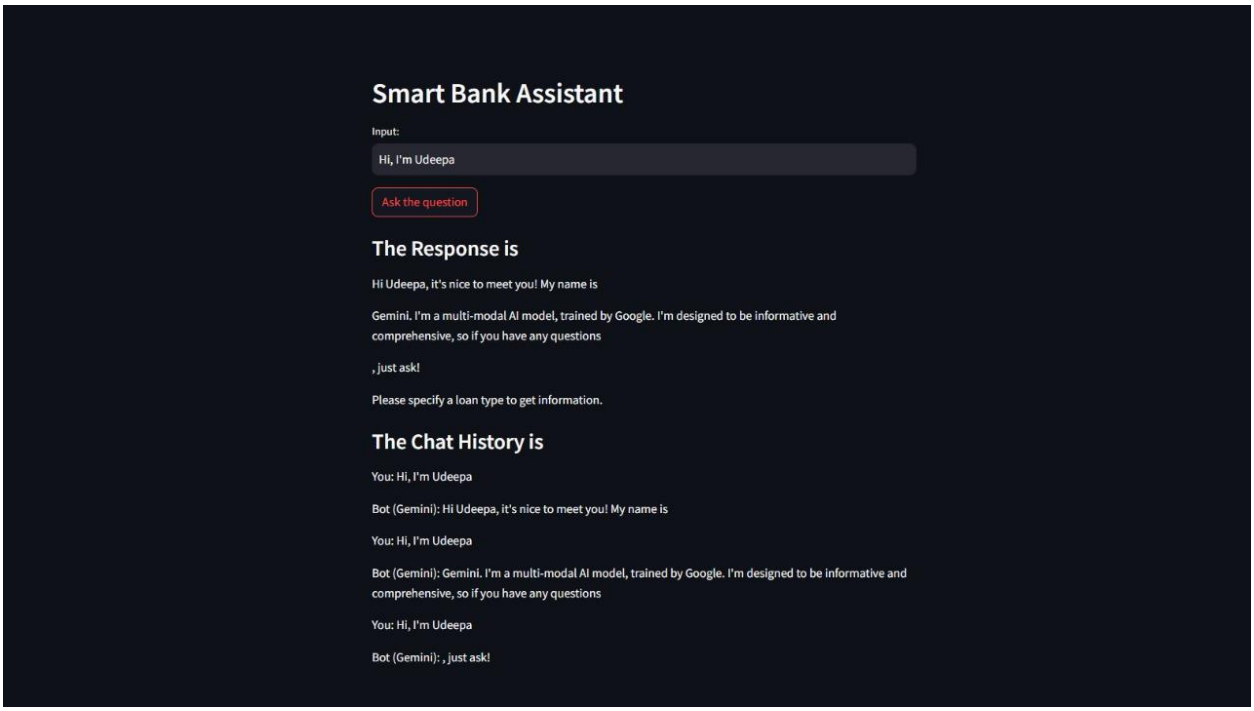
Updates and displays the chat history.

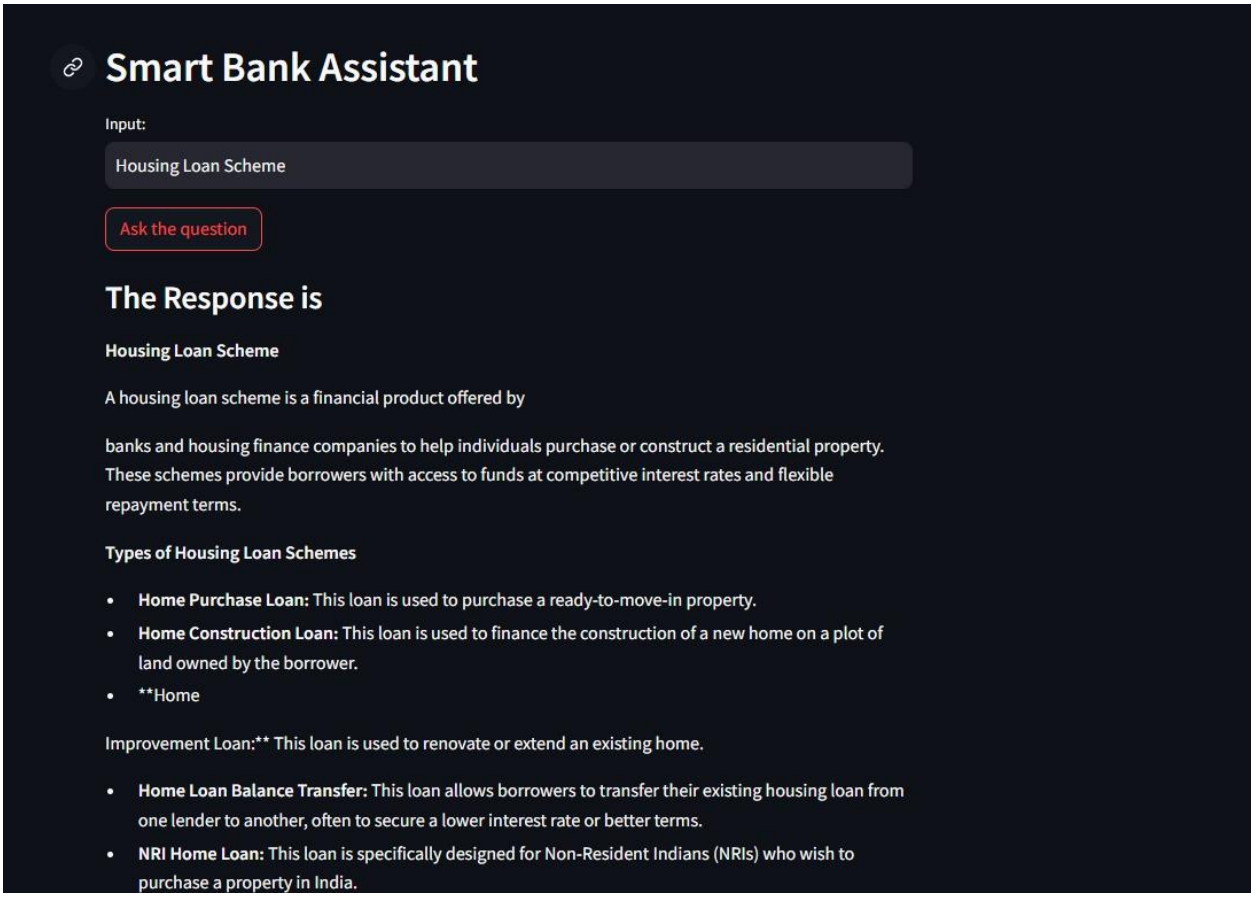
Executable Block:

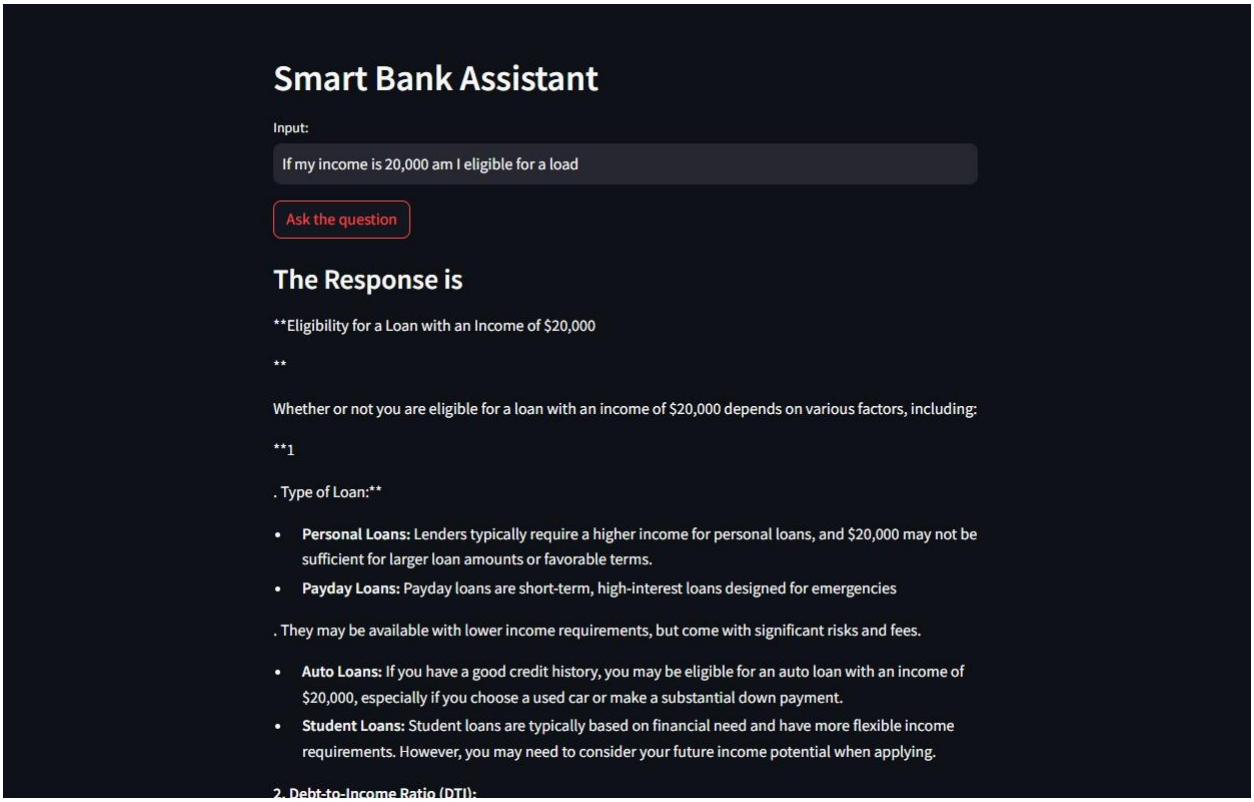
Checks if the script is being run directly (__name__ == "__main__") and then calls the main() function.

Overall, this code sets up a Streamlit application that acts as a front-end interface for a chatbot powered by Google's GenerativeAI. The chatbot provides loan support to users by answering their questions and providing information about different types of loans offered by Smart Bank.

Output


Smart Bank Assistant
Input:
Hi, I'm Udeepa
Ask the question
The Response is
Hi Udeepa, it's nice to meet you! My name is
Gemini. I'm a multi-modal AI model, trained by Google. I'm designed to be informative and comprehensive, so if you have any questions
, just ask!
Please specify a loan type to get information.
The Chat History is
You: Hi, I'm Udeepa
Bot (Gemini): Hi Udeepa, it's nice to meet you! My name is
You: Hi, I'm Udeepa
Bot (Gemini): Gemini. I'm a multi-modal AI model, trained by Google. I'm designed to be informative and comprehensive, so if you have any questions
You: Hi, I'm Udeepa
Bot (Gemini): , just ask!


Smart Bank Assistant
Input:
Housing Loan Scheme
Ask the question
The Response is
Housing Loan Scheme
A housing loan scheme is a financial product offered by
banks and housing finance companies to help individuals purchase or construct a residential property. These schemes provide borrowers with access to funds at competitive interest rates and flexible repayment terms.
Types of Housing Loan Schemes
Home Purchase Loan: This loan is used to purchase a ready-to-move-in property.
Home Construction Loan: This loan is used to finance the construction of a new home on a plot of land owned by the borrower.
**Home
Improvement Loan:** This loan is used to renovate or extend an existing home.
Home Loan Balance Transfer: This loan allows borrowers to transfer their existing housing loan from one lender to another, often to secure a lower interest rate or better terms.
NRI Home Loan: This loan is specifically designed for Non-Resident Indians (NRIs) who wish to purchase a property in India.



Approach Taken

Crafting a sophisticated Conversational Q&A Chatbot using the Gemini Pro Free API entails leveraging Google's advanced language models to integrate cutting-edge natural language processing capabilities into applications. This approach enables developers to create chatbots that dynamically adapt to user queries, offering intelligent and context-aware responses.

By harnessing the power of the Gemini Pro API, developers can tap into state-of-the-art language models, enabling their chatbots to understand and respond to a wide range of user inquiries with high accuracy and relevance. This advanced functionality empowers chatbots to handle complex conversational scenarios, providing users with a seamless and intuitive interaction experience.

In essence, choosing to utilize the Gemini Pro Free API for building a Conversational Q&A Chatbot offers developers access to robust natural language processing capabilities, enabling the creation of intelligent, context-aware, and highly responsive chatbots that enhance user engagement and satisfaction.

SetUp Gemini Pro AI

After setting up the Gemini Pro API and creating the necessary environment files, follow these steps to run the chatbot application:

Initialize the Virtual Environment: Open your terminal and navigate to the directory where you created the chatbot project. If you haven't done so already, create a virtual environment.

Craft the Requirements File: Create a requirements.txt file and add the necessary packages for the project

Write the Chatbot Code: Create a Python script (e.g., app.py) and paste the provided code for the chatbot application into this file.

Run the Streamlit Application: Execute the following command in your terminal to start the Streamlit application

Also we have to use API key.