AI Financial Advisor Agent on Telegram

Bootcamp by Ivy Professional School (www.ivyproschool.com)

Learn to create more such no / low code AI tools in 60 minutes and boost your Resume - Click here: https://www.youtube.com/ivyproschool/?sub_confirmation=1

High-level Steps to Create the Telegram AI Bot

1 Setup and Configuration

- **Install Python** (if you don't have it already) by following the steps from this short video: https://youtu.be/posFkGyrowo?t=2018
- **Create Telegram Bot Key and Google Gemini API Key** by following the steps given in this document: Materials
- **Download VS Code** (interactive development environment) from this link: https://code.visualstudio.com/download
- Install `telebot` and `google.generativeai (type: "CMD" in windows search, select the cmd command, it should open a black terminal window, type: install telebot google`
- Get API keys for Telegram Bot and Google GenAI
- Initialize the bot and configure the AI model

2 Handling Telegram Messages

- Create a `/start` command handler
- Use `bot.register_next_step_handler` to collect user responses

3 Collecting User Inputs

- Ask for user age, income, expenses, and financial goals
- Store responses in a dictionary

4 Generating AI-Based Financial Advice

- Formulate a detailed prompt using user data
- Use Google GenAI to generate personalized financial guidance

5 Deploying the Bot

- Run `bot.infinity_polling()` to keep the bot active
- Ensure API keys are secure (use environment variables)

Detailed Steps:

Open VS Code, File -> Open Folder (navigate to the folder where you've saved telegram_phase1.py, telegram_phase2.py, telegram_phase3.py files), Select folder.

Then, click on each .py (python) file within VS Code, and open it. Replace the Telegram BOT Token and Google Gemini API at the top of the code with your Bot Token and Google Gemini API key from the previous Setup step).

Then, save the file (Ctrl+S or File Save).

Open cmd in windows search. It should open a command line terminal. Cd to the folder where the .py files are saved. (Copy the folder path as shown in the video, and then type: cd <folder path>)

Then, follow the steps as shown in the video.

—---

Let's go through your code step by step and break it down so you can clearly understand how it works.

1. Importing Libraries

```
import os
import telebot
import google.generativeai as genai
```

- `os`: Allows interaction with the operating system (e.g., for environment variables).
- `telebot`: A Python library for interacting with Telegram's Bot API.
- `google.generativeai as genai`: Imports Google's Generative AI module for content generation.

---

2. Defining API Keys and Initializing Services

```
BOT_TOKEN = "7815614309:AAEfD2834TsdUrTxTYBS3_gjXBOQEvE6QDs"
GEMINI_API_KEY = "AIzaSyCHrX-61FtKU8689en6z_5tdfk2Fhl2LyY"
```

- These are **authentication tokens** required for Telegram (`BOT_TOKEN`) and Google's Generative AI (`GEMINI_API_KEY`).

In a real-world project, you **should not** hardcode these values. Instead, use environment variables like:

```
 BOT_TOKEN = os.getenv("BOT_TOKEN")
GEMINI_API_KEY = os.getenv("GEMINI_API_KEY")
```

-

---

3. Creating a Telegram Bot and Configuring Google GenAI

```
bot = telebot.TeleBot(BOT_TOKEN)  # Initialize the Telegram bot with the token
genai.configure(api_key=GEMINI_API_KEY)  # Configure Google Generative AI
model = genai.GenerativeModel("gemini-pro")  # Create an instance of the AI model
```

- `bot = telebot.TeleBot(BOT_TOKEN)`: Creates an instance of a Telegram bot.
- `genai.configure(api_key=GEMINI_API_KEY)`: Configures the AI service with the API key.
- `model = genai.GenerativeModel("gemini-pro")`: Loads the **Gemini Pro** model, which will generate responses.

---

4. Creating a User Data Storage

```
user_data = {}
```

- `user_data` is a **dictionary** that stores user inputs temporarily.
- Each user is identified by their **chat ID**, and their responses (age, income, expenses, goals) are stored inside this dictionary.

---

5. Handling the `/start` Command

```
@bot.message_handler(commands=["start"])
def start(message):
    user_id = message.chat.id  # Get the unique ID of the user
    user_data[user_id] = {}  # Create an empty dictionary for this user

    bot.send_message(
        user_id,
        "Welcome to Finance Bot!\n"
        "Let’s create a personalized financial plan for You.\n\n"
        "First, share your age:"
    )
    bot.register_next_step_handler(message, get_age)  # Move to next step
```

- **`@bot.message_handler(commands=["start"])`**: This is a **decorator** that tells the bot to run this function **when a user sends `/start`**.
- **`message.chat.id`**: Extracts the user’s **unique chat ID** from the message.

- **`user_data[user_id] = {}`**: Creates an empty dictionary to store the user's responses.
- **`bot.send_message(user_id, "...")`**: Sends a welcome message.
- **`bot.register_next_step_handler(message, get_age)`**: Tells the bot to **wait for the user's next response**, and then call `get_age` when a reply is received.

---

6. Handling User Input (Age)

```
def get_age(message):
    user_id = message.chat.id
    user_data[user_id]["age"] = message.text  # Store age

    bot.send_message(user_id, "💸 What is your monthly income (in ₹)?")
    bot.register_next_step_handler(message, get_income)
```

- **Stores the user's age** in the `user_data` dictionary.
- **Asks the next question** (monthly income).
- **Registers `get_income` as the next handler**, meaning when the user replies, `get_income` will process it.

---

7. Handling User Input (Income)

```
def get_income(message):
    user_id = message.chat.id
    user_data[user_id]["income"] = message.text  # Store income

    bot.send_message(user_id, "What are your monthly expenses (in ₹)?")
    bot.register_next_step_handler(message, get_expenses)
```

- Stores the **income** and then asks for **expenses**.
- Moves to `get_expenses`.

---

8. Handling User Input (Expenses)

```
def get_expenses(message):
    user_id = message.chat.id
    user_data[user_id]["expenses"] = message.text  # Store expenses

    bot.send_message(
        user_id,
        "What are your financial goals?\n"
        "(e.g., Buy a home, Child’s education, Retirement, Tax saving):"
    )
    bot.register_next_step_handler(message, get_goals)
```

- Stores the **expenses** and then asks for **financial goals**.
- Moves to `get_goals`.

---

9. Handling User Input (Goals) and Calling Google GenAI

```
def get_goals(message):
    user_id = message.chat.id
    user_data[user_id]["goals"] = message.text  # Store goals

    data = user_data[user_id]  # Retrieve all user data

    # Create a detailed prompt for AI
    prompt = f"""
    Act as a certified Indian financial advisor. Create a personalized plan for a
{data['age']}-year-old with:
    - Monthly income: ₹{data['income']}
    - Monthly expenses: ₹{data['expenses']}
    - Financial goals: {data['goals']}

    Provide advice tailored to India, including:
    1. Budgeting: Follow the 50-30-20 rule for Indian cost of living.
    2. Savings: Recommend PPF, FD, RD, or NPS.
    3. Tax Saving: Suggest ELSS, Section 80C deductions, HRA, or health insurance.
    4. Investments: Equity (stocks/MFs), gold, real estate, or digital gold.
    5. Debt Management: Handle loans (home, car, personal) or credit cards.
    6. Risk Assessment: Account for inflation (6-7%) and emergency funds.

    Use bullet points, ₹ currency, and terms like 'lakh'/'crore'. Avoid jargon.
    """

    try:
        response = model.generate_content(prompt)  # Generate AI response
        bot.send_message(user_id, f"📊 Your India-Focused Financial
Plan\n\n{response.text}")
    except Exception as e:
        bot.send_message(user_id, "⚠️ Error generating advice. Please try again!")
```

- The **prompt** is created with all user inputs.
- The **AI model is called** to generate a personalized financial plan.
- The bot **sends the AI-generated response** to the user.
- If there's an error, it notifies the user.

10. Running the Bot

```
if __name__ == "__main__":
    bot.infinity_polling()
```

- This **keeps the bot running indefinitely**, continuously listening for new messages.

How the Code Works in a Full Conversation

User Action	Bot Response
User sends `/start`	"Welcome to Finance Bot! First, share your age:"
User enters `30`	"💸 What is your monthly income (in ₹)?"
User enters `50000`	"What are your monthly expenses (in ₹)?"
User enters `20000`	"What are your financial goals?"
User enters `"Buy a house, Retirement"`	**(Generates AI-based financial advice)**

Key Takeaways

- **Telegram Message Handlers:**
  - `/start` command is handled by `@bot.message_handler(commands=["start"])`
  - Regular text responses are handled using `bot.register_next_step_handler`
- **Sequential Input Collection:**
  - The bot asks one question at a time and stores user inputs in `user_data`
- **AI Integration:**
  - User responses are sent to Google GenAI to generate a **custom financial plan**
- **Error Handling:**
  - If AI generation fails, the bot sends an error message.

This should help you fully understand how your bot works! Let me know if anything needs more clarification. 🚀