Python Assistant Chatbot

By
Anthony Tran, Robert Statham, Yoonkyung Lee

https://youtu.be/vNicB2U7Bn4
https://github.com/hxt200010/cs6320_project.git

About the Project (Proposal):

- Our project is an intelligent chatbot that helps our users improve their coding skills through conversation practice and interaction. It will provide users with information from the relevant documentation using **Retrieval Augmented Generative (RAG)** to pull information from a knowledge base of Python library documentation to ensure an accurate and context-aware response.

Goal:

- Our goal is to fulfill all 3 below features from the application that help users learn better every day:
    - **Interactive Learning:** Engages users in real-world conversation
    - **Everything Python assistance:** Provides library recommendations based on user needs
    - **Programming Correction:** Helps users find the relevant solution for user questions

Scope:

- **More prettier UI**
    - Make a nice user interface using Next.js
- **One Stop Shop for Python Documentation**
    - Bot can handle any popular Python library and show it to the user
- **Check program efficiency**
    - Choose the top 3 best documents based on the user inquiry
    - AI will provide a suggestion with an explanation

Team Members:

- We will work together to implement Retrieval Augmented Generation to enhance the accuracy of our model, we will work on further details regarding issues and errors that come up.
- We will access Vector Database APIs to implement the project
- We will spread out data gathering and combine them.
- We will use Databricks for data aggregation and cleaning

Data Source:

- Python Documentation

YOUTUBE LINK:

https://youtu.be/vNicB2U7Bn4

GITHUB LINK:

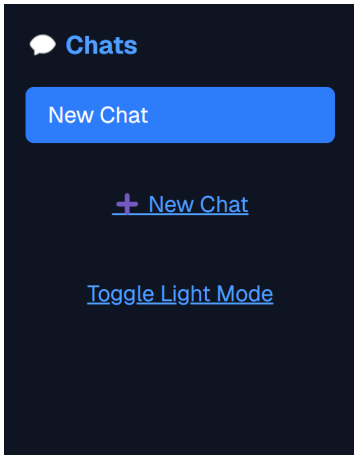
https://github.com/hxt200010/cs6320_project.git

Progress Report

- **Technology & Dataset that we use:**
  - We used **FastAPI, LangChain, OpenAI, FAISS, Tesseract OCR, Python** for our project
  - We used this Dataset to work our project, this dataset contains all information about Python
- **Backend Overview**
  - The code contains these components:

    - **FastAPI Server Setup**
      - Serves as the backend API
      - Configured with **Cors middleware** to allow frontend access from any origin
    - **Documentation Loading**
      - Automatically extract the python-3.13-docs-text.zip archive if not already extracted.
      - Reads all .txt files from the extracted directory
      - Wraps each document into LangChain's Document objects for downstream processing
    - **Document Splitting & Embedding**
      - Use **RecursiveCharacterTextSplitter** to divide large text files into manageable chunks
      - Generate **Vector embedding** for each chunk using OpenAIEmbeddings
      - Stores and indexes them using **FAISS** for fast retrieval
    - **Conversational QA Chain**
      - Use ConversationalRetrievalChain from LangChain to enable **contextual conversation**.
      - Maintains chat history with **ConversationBufferMemory**
      - Enforces strict answering rules via a **custom prompt template:**
        - Only answer Python-related questions
        - Reject or deflect anything outside of Python documentation
        - Justify the answer by referencing specific parts of the docs
    - **API Endpoints**
      - Post/ask: Accepts a user question and returns a Python-based answer with sources
      - Post /ask-image: Accepts an image file, extracts any text (like Python code) using **Tesseract OCR /** interprets the code (if valid), and answers based on Python documentation context
  - **Intelligence Feature:**
    - **Contextual Memory:** Maintains conversation history to preserve the thread of discussion
    - **Image-to-code Understanding:** Use **pytesseract** + **act.parse()** to determine if OCR-extract content is valid Python syntax
    - **Strict Domain Enforcement:** Ensures the assistant **only answers questions from the Python documentation**
- **Frontend Overview**
  - **Features**
    - **Chat Interface:** Users can send text or image-based questions to the bot

        - **Bot Response:** Displayed in real time using typing animation effect
        - **Multi-Session Support:** Users can create and navigate between multiple chats
        - **Dark / Light Mode:** Users can toggle between dark and light mode seamlessly using Tailwind's dark: class
        - **Markdown Rendering:** Bot response support formatting like code, list, bold, etc…
        - **Image Input Handling:**
            - Drag / drop and file upload
            - Clipboard paste detection
            - Thumbnail preview + delete option
    - **Enhancement Byond Basic UI**
        - **Framer Motion Animation:** Smooth transition for the new message
        - **Image Preview + Cleanup UI:** Ensures users see and manage attachments before submission
        - **Scroll Tracking + Control:** "Scroll to bottom" Button only appears when relevant
        - **Request Locking with useRef:** Prevents request spamming during backend processing
    - **API Integration**
        - Connected to the FastAPI backend through:
            - **Post / Ask:** for text-only queries
            - **Post / ask-image:** For OCR-based image questions
            - Appends **chat_id** for session continuity
- **Accomplishment:**
    - **Basic Implementation:**
        - Accomplished implemented basic logic, such as extract data from the documentation, train the model, and implemented RAG method
        - Basic Front end, such as chat interaction/interface,
        - show clear distinction between user and bot.
        - Organize and clean.
    - **Beyond Baseline:**
        - **Bot answers improvement**
            - More professional
            - More detail oriented
            - Answer more carefully through the use of more advanced AI model
            - It now can read text from image, extract and interpret Python code using Tesseract OCR
            - Bot knows Python through Documentation, nothing else



        - **Improvement on User Interaction:**
            - Conversation memory stores chat memory using **ConversationBufferMemory**
            - Bot gradually reply to user so it create a more natural way between conversation
            - **Dark / Light mode:** suitable for working both night and day using Tailwind's dark
            - **Multi-Session support:** Users can create and navigate between multiple chats
            - **Image Input Handling:**
                - Drag / Drop and file upload
                - Clipboard paste detection

        - Thumbnail preview + delete option
    - **Framer Motion Animation:** Smooth transition for new message
    - **Scrolling Tracking + Control:** “Scroll to Bottom” button implemented
    - **Stop while replying:** Ability to stop the bot reply mid-way.
- **Challenges**
    - **App problem:** App crashed when switching chat session
    - **Pasted Image not aligned perfectly:** Future implementation will fix the issue
- **Lesson Learned:**
    - We sometime over complicated our project while the basic feature is under-performing
    - We implemented main features like basic frontend and backend one by one
    - We then add the framer motion animation to ensure smoother application
    - We then add other extra features (dark / light mode), stop button after everything got fix
    - Our future improvement will involve adding features one by one instead of an all-in-one fix all method like what we are doing.
- **Future Accomplishment Goal:**
    - **Fix any recurring issue:** Issues such as crashing, glitching will be fixed for next release
    - **Add an additional features such as:**
        - **User log in:** User will be able to sign up / log in using their account
        - **Support custom doc:** User will be able to upload their doc and ask the bot to answer according to the doc, instead of just one document
        - **Separate between Code and text:** Code and text will have different fonts to differentiate them, so it’s easier to read for our users.
        - **Create our own LLM for our app instead of using ChatGPT / OLLAMA:** We are on the way to create a large language model (LLM) for our app.
        - **Mobile compatibility:** Future implementation will bring the ability to use the app via mobile devices
        - **Text-to-Speech / Speech-to-Text Support:** Ability to let users speak, then transform into texts for easier use.
- **Team Distribution:**
    - **Robert Statham**
        - Designed overall project concept and goal
        - Collected data
        - Identified and implemented useful technologies relevant for our use case
        - Implemented LLM and RAG Backend
    - **Anthony Tran**
        - Implemented and improve Front-end features such as:
            - **Scroll Tracking + Control:** “Scroll to bottom” Button only appears when relevant
            - **Streaming Bot Replies:** Bot replies gradually character-by-character for a natural conversation effect
            - **Framer motion implementing:** Implement smoother user interface
        - **Improve backend chatbot performance:**
            - **Improve bot replies:** Better, more polite, always end up with an emoji
            - Using chatgpt 4.0 to enhance bot performance and provide a more detailed oriented response to our user.
    - **Yoonkyung Lee**
        - Contributed to project ideation and topic selection
        - Assisted with data preprocessing and backend integration
        - Provided UI/UX feedback and supported technical documentation
- **Test trial:**
    - We let over 20 trial users try out our apps, including family members and club members.

    - We take their feedback seriously and we work on the app and bring their feedback to life.
    - We have successfully improved our app from their feedback with the following features:
        - **Smooth Transition:** Ensure smooth user interface
        - **Multi-session support:** Support multi-session chat
        - **Dark / Light mode:** We recognize that our users love dark/light mode, so we successfully added these features into our app.

Self Scoring:

Anthony Tran

- 80 points - Innovation & Creativity: I designed front end features including stop button, scroll down button
- 10 points - I'm making sure the bot answer properly with emoji at the end of the line.
- 10 points - I made youtube video for the demo.

Robert Statham

- 70 points - Innovation & creativity: I designed the overall project concept
- 20 points - I gathered the relevant data and relevant technologies (libraries)
- 10 points - I helped test the app

Yoonkyung Lee

- 60 points - Innovation & Creativity: I contributed to the topic selection process
- 20 points - I validated the accuracy and relevance of the bot's responses through extensive testing
- 20 points - I gathered diverse user feedback and provided actionable ideas to improve the project.

Conclusion:

- We are thankful for many of our supporters who gave us much feedback and directions to move forward. With what we have done, we are confident that our app will make life easier and more productive with our ever-improving documentation reader app.