Fairfield University

Stag Coders

Multitasking AI Chatbot

Connor Hehn, Reyes Huerta, David McNulty, Andrew Visceglia
Advisor: Dr. Sidike Paheding

Introduction

- Leveraging LLMs for the development of a multitasking AI chatbot that is adept at handling voice and text commands.
- **Problem Statement:** Deliver an accessible and multitasking AI chatbot application (the first of its kind), specifically tailored to meet the needs of the Fairfield University community, in particular, first-year students.
- **Objectives/Goals:** Listening to users, engaging in naturalistic conversations, conducting online searches, providing maps or directions for navigational purposes, and playing music.

Background

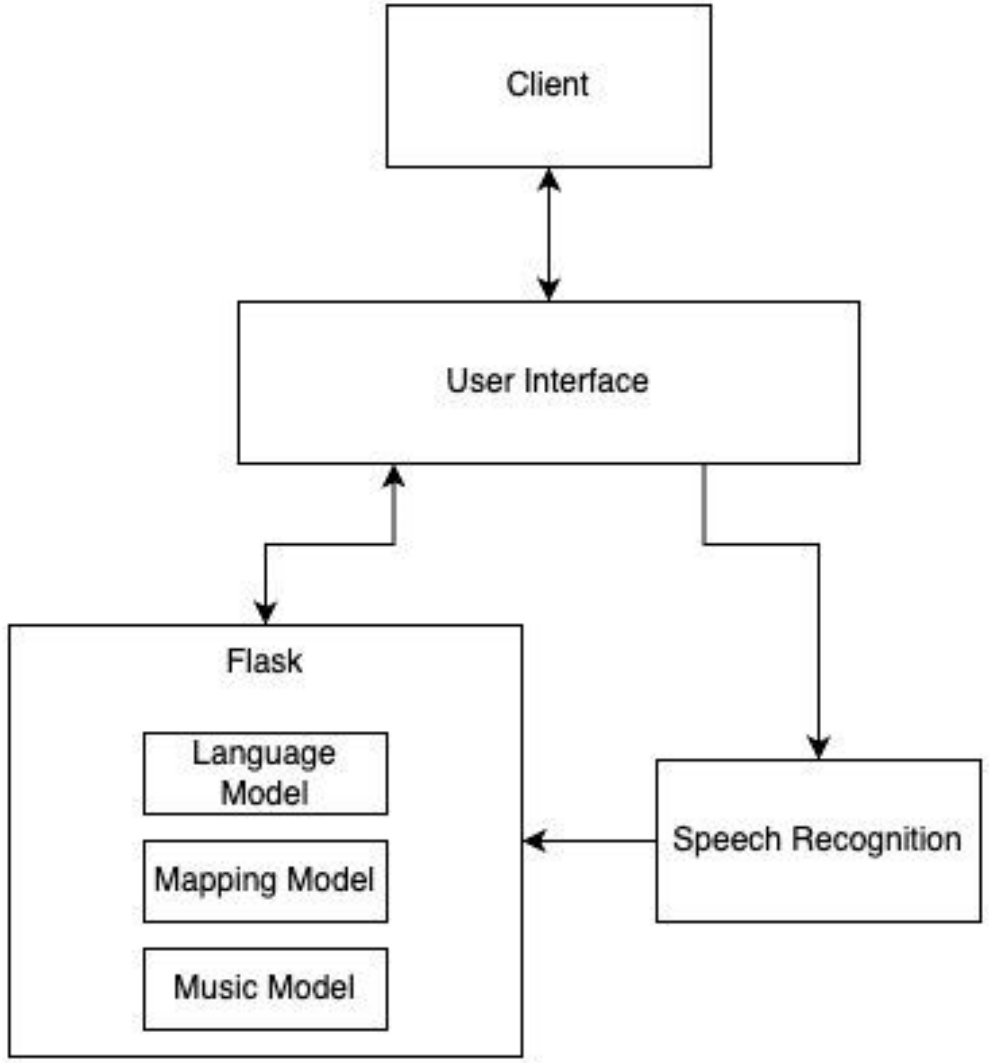
- In recent years, the rapid growth of AI has been fueled by the development of generative AI applications from OpenAI, Google, and Microsoft.
- IBM defines chatbots as AI-driven programs simulating human conversation through NLP.
- Transformer models (e.g., GPT-3) are essential for modern AI chatbot progression, with the attention mechanism enhancing contextual understanding.
- **Motivation:** No pre-existing chatbot application available to Fairfield University students.

Project Scope

- The overarching objective of this project is to deliver an accessible and multitasking AI chatbot application that is tailored to meet the needs of prospective Fairfield University students through multitasking functionality.
- **Multitasking functionality includes:**
  - Language selection
  - Speech recognition
  - Mapping
  - Music

Fairfield UNIVERSITY

Model Architecture



Language Model

- **Final language model chosen:** Mixtral-8x7B
- Pre-trained Generative Sparse Mixture of Experts architecture which enhances its ability beyond a simple GPT model.
- Advertised as the “best open-source model, with a performance comparable to GPT3.5” according to Mistral.
- **Capability with 5 languages:**
  - English
  - Spanish
  - French
  - Italian
  - German

Language Model Continued

- The model contains the following parameters which are fine-tuned for our application:
  - System prompt - initial context provided
  - Temperature - randomness of responses
  - Max tokens - maximum length of response
  - Top-p (nucleus sampling) - randomly selecting diverse output tokens
  - Repetition penalty - penalty for repeated tokens

Mapping Model

- GraphHopper API for geocoding and routing:
  - Open-source
  - Built on top of OpenStreetMap (OSM) data
  - Geocoding - converts textual address into geographic coordinates (latitude and longitude); finds exact locations based on user-provided address data
  - Routing - calculates routes between given coordinates for various modes of transportation (in our case, walking); returns detailed route information including distance, estimated time, and step-by-step directions
    - Support for polyline encoding of the route path

Mapping Model Continued

- **Python code:**
  - Geocoding function was created to make a request to the GraphHopper geocoding API - constructs URL for geocoding endpoint → GET request → parses JSON to extract coordinates
  - Dropdown widgets - allow user to select start and end locations from predefined list of locations on campus
  - Route calculation button - on click, route calculation function is called; retrieves selected start and end locations, obtains coordinates, constructs routing URL, GET request, parses routing API response to display proper output

Music Model

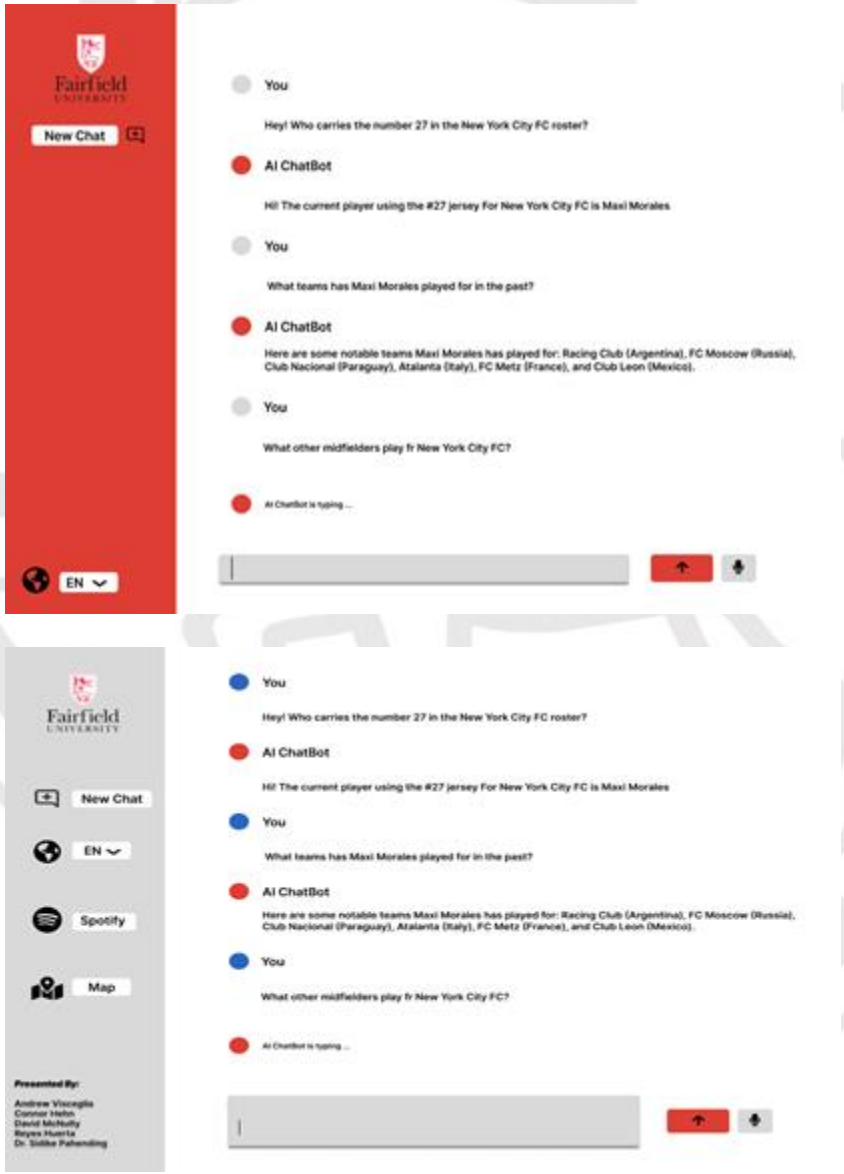
- **Python code:**
  - Music functionality was achieved via Spotify's web API
    - Decided to move forward with the Spotify API based on its free-of-cost nature, industry prevalence, and ease of use
  - The two API endpoints called in our Flask application consist of POST requests
    - /handle_spotify retrieves the access token to authenticate Spotify permissions, and extracts requested attributes (track name, album, artist, album image, track_url) in a JSON format.
    - /play retrieves the specific preview URL for the indicated track. A GET request is made to the Spotify API, and the returned result is played in the UI soundbar.

Speech Recognition Model

- **Enhanced Interaction with Speech Recognition:**
  - Users can seamlessly interact with the AI ChatBot using spoken commands.
  - Speech recognition functionality improves user functionality and accessibility.
- **Utilizing Web Speech API for Efficiency:**
  - The chosen model for speech recognition is webkitSpeechRecognition, part of the Web Speech API in modern web browsers.
  - When the record speech button is pressed, a new speech recognition object is created, capturing spoken text.
- **Efficient Implementation with JavaScript:**
  - Implemented in JavaScript, not the Flask backend written in Python.
  - JavaScript implementation ensures faster processing and shorter delays in speech recognition.

User Interface Design

- After going through a few revisions, our User Interface consists of:
  - An accessible new chat button for instant engagement.
  - Speech recognition button for a hands-free interaction.
  - Language selector for multilingual support.
  - Access music instantly with the Spotify button, leveraging our Spotify API integration.
  - Explore campus effortlessly with the mapping button for detailed navigation.
  - Overall placement, color palette, and design of our user interface.



Results and Analysis

- Our ongoing development encompasses an immersive chatbot experience, boasting a plethora of dynamic features:
  - Seamless speech recognition, enabling effortless communication
  - Intuitive language selection, allowing users to effortlessly switch between five languages (English, Spanish, French, German, Italian)
  - Interactive mapping capabilities enhancing user navigation
  - Spotify integration, providing users with personalized music listening
  - The application is currently hosted on Hugging Face Spaces and live to the public.

Live Demo

Stag Chatbot

Conclusions and Future Work

- Final version of chatbot application achieved all its desired functionality.
- Completely built from the ground up.
- Hosted on web application (Hugging Face Spaces).
- GUI includes buttons to easily navigate the chatbot functionality and select a language.
- User can manually type in message, select pre-existing prompt, or record speech.
- Explore unique mapping and/or music functionality.
- **Future versions:**
  - Connect application to a database → users can create their own account and concurrently interact with chatbot without interference
  - Ensure user data is handled safely and ethically.
  - iOS/Android application or integrate with “Fairfield U” app.
  - Explore brand new functionality/update pre-existing functionality.
  - Train the language model.

Questions

- Please feel free to ask any questions with the remaining time or email us at:
  - Connor Hehn: connor.hehn@student.fairfield.edu
  - Reyes Huerta: reyes.huerta@student.fairfield.edu
  - David McNulty: david.mcnulty@student.fairfield.edu
  - Andrew Visceglia: andrew.visceglia@student.fairfield.edu