BOSTON UNIVERSITY

CEO

Chief Technology Advisor

Chief Information Officer

Intern

411 Project

Hunters and Gatherers

About Software Engineering

AdCresco: to grow towards, forward we shall march

Powered by Meth

Criteria:

1. It must utilize a database. A simple way to meet this requirement is to require a user to store profile information in the database. You'll also be using it as a cache.

2. It must correlate at least two publicly available data sets via API from the Internet. Examples might be weather/climate data from NOAA, crime statistics from the FBI, and so on. A great place to get started is https://apigee.com/providers?apig_cc=1, which is a repository of datasets and APIs, and https://www.programmableweb.com (my favorite). Another good place to search for data is http://data.gov. The City of Boston also has data available at data.cityofboston.gov. Your application must correlate these data sets in some way; for example, pull a user's playlist from Spotify and correlate it with a feed that has concert dates to alert the user of bands that they like that are playing nearby. Use of the Google Maps or Geolocation service does not count toward your two APIs.

3. It must use third-party authentication, for example logging in with Twitter or Facebook using OAuth.

4. It must have a decoupled architecture, similar to what we looked at in class during the 'dogfooding' lecture. The implication is that you'll need a front end and a back end, and the two will communicate via a RESTful interface. It's too early to discuss technologies, but this does mean that there will be JavaScript in the front end.

Since the back end is responding to requests and just returning data, it doesn't necessarily need to be in JavaScript…Python, Java, PHP, and so on would work. Additional constraints may be placed on the project as the semester progresses. At this point in the project you should not be thinking much of the technical aspects of the site...this part of the project is a business function. The technical design will come later. Just think about the user benefits for now, not the technical implementation. That being said, browsing through available APIs can be a good way to spark your imagination. A Google search for 'databases with public API' will give you a nice list in addition to those listed above. One member of your group should submit your repo link on Gradescope by the due date and time.

Code Shared:
https://colab.research.google.com/drive/1K1rPhu9QHU4tCZw6lvztTungC5UBqgPC

Project Proposal: Interactive Mannequin with AI Chatbot Integration

Project Overview: The objective of this project is to create an interactive mannequin that can engage in conversations with users. The mannequin will have an Arduino-based computer system integrated with various APIs to facilitate voice recognition, natural language understanding, text-to-speech, and speech-to-text capabilities. Additionally, a database will be implemented to store user information and conversation logs for personalized interactions.

Project Components:

Hardware Setup:

- Mannequin: A realistic mannequin with a speaker and microphone setup for interaction.
- Arduino: An Arduino board for controlling the hardware and interfacing with the APIs.
- Speaker and Microphone: Quality audio components for clear voice input and output.

Software Components:

- Voice Recognition API:
    - Options: Google Cloud Speech-to-Text, Microsoft Azure Speech Service, or IBM Watson Speech to Text.
    - Purpose: To recognize the user's voice and trigger the interaction when the specific user's voice is detected.
- Natural Language Processing (NLP) API:
    - Option: OpenAI GPT-4 or equivalent NLP model.
    - Purpose: To process and understand user queries and generate meaningful responses.
- Text-to-Speech API:
    - Options: Google Text-to-Speech, Amazon Polly, or Microsoft Azure Text to Speech.
    - Purpose: To convert text responses from the NLP model into natural-sounding speech for the mannequin to communicate with the user.
- Speech-to-Text API:
    - Options: Google Cloud Speech-to-Text, Microsoft Azure Speech Service, or IBM Watson Speech to Text.
    - Purpose: To convert user speech input into text for processing by the NLP model.
- Database Management:
    - Database System: Choose a suitable relational database system like MySQL or PostgreSQL.

- Purpose: To store user information (name, date of birth, interests) and conversation logs for personalized interactions and historical reference.

System Workflow:

- User speaks to the mannequin.
- Voice recognition API detects the user's voice and triggers the interaction.
- User's speech is converted to text using the speech-to-text API.
- The text input is sent to the NLP API for understanding.
- NLP generates a response based on the user's input.
- The response text is converted to speech using the text-to-speech API.
- The mannequin communicates the response to the user.

Database Implementation:

- Design a database schema to store user profiles and conversation logs.
- Implement data storage and retrieval functionality within the Arduino setup using a suitable database library.

Security and Privacy:

- Implement secure protocols for data transmission and storage.
- Ensure user data is encrypted and protected in compliance with relevant data protection regulations.

User Interface:

- Talking to mannequin

Project Timeline: The project can be broken down into several phases, with each phase focusing on different components and API integrations. A tentative timeline could be as follows:

- Phase 1: Hardware Setup and Voice Recognition Integration
- Phase 2: NLP Model Integration and Speech-to-Text Implementation
- Phase 3: Text-to-Speech Integration and Database Design
- Phase 4: Database Implementation and User Interface Development
- Phase 5: Testing, Debugging, and User Experience Enhancement