Branch: master cpAI_EECS_498 / documentation_frontend.md Find file Copy path

koeul typo 0a0d541 1 minute ago

1 contributor

80 lines (49 sloc) 3.04 KB Raw Blame History

CPai Setup Guide

CPai is composed of 4 parts:

1. Dialogflow service provided by Google
2. Backend written in Python/Flask
3. Client written in Javascript(Node) - this is another backend that hosts Google Client for interacting with Dialogflow
4. Frontend written in Javascript(Vue)

Frontend structure

The frontend for CPai is a typical SPA (single-page web app) built in Vue.js, a reactive JS framework.

The practical entry point is `App.vue` which includes a Vue Router which automatically direct users to `/console` route which will display Console component implemented in `view/Console.vue` .

The Console component has a sub-component Chat implemented in `component/Chat.js` . `Chat.js` component will use the Vuex functions to handle any incoming/outgoing data.

Vuex

In addition to the core Vue.js modules, we also use Vuex, a state management library implementing FLUX pattern enforcing a unidirectional data flow.

As such, all external data will flow through Vuex modules found under `store` . In particular, `conversation.js` module contains an instance of `Axios` to send and fetch data from the Client that hosts Google API client.

For example, whenever there is an interaction involving Chat component, a timestamp will be appended to the url for the image version of form 1040 also stored in Vuex and Console component will reactively display a newly updated form 1040.

Speech-to-text and text-to-speech

Any STT and TTS functions are implemented in `SpeechService.js` . For STT, whenever there is a response from a browser, it will push the new data into our Vuex store notifying Chat component to reactively display the recongized speech in the text form and change the microphone icon.

CSS/Other plugins

We use Tailwind, a utility-based CSS framework and Vue Spinnner(https://github.com/greyby/vue-spinner) all licensed under MIT.

Vue Spinner is used in Chat component for displaying a loading state.

Frontend setup guide

The frontend can be built and *statically hosted* on the web. The build environment tested was Ubuntu 18.04 LTS running on AWS EC2 instance (the same setup used for client and backend).

1. Launch a new EC2 instance by choosing Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0fc20dd1da406780b (64-bit x86).
2. SSH into the newly launched EC2 instance and update the system (enter yes to any prompts):

```
> sudo apt update; sudo apt -y upgrade;
```

3. Install Node.js (this setup installed v13.13.0, the latest version)

```
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
sudo apt-get install -y nodejs
```

4. Install "yarn" (this setup installed v1.22) and restart the terminal.

~~~
curl -o- -L https://yarnpkg.com/install.sh | bash

1. Replace the urls for backend and client in `frontend/store/index.js`:

```js
16:         clientUrl: "http://localhost:3000/",
17:         backendUrl: "http://localhost:5000/",
~~~

5. Build the frontend

```
yarn install
yarn build
```

You may also run it locally using:

```
yarn serve
```

6. Upload the content inside `dist` folder to your choice of webserver.