Algorithmic Portrait

Prototype Report - by Audrey Coulombe

Link to github repository:
https://github.com/AudreyCoulombe/CART451_SemesterProject

Project Description and Intended User Experience

Algorithmic Portrait is an app allowing users to visualise how they are perceived by algorithms. They are first invited to take a quiz to “find out how algorithms see them”. At this point, the experience should be recalling quizzes on social media that invite people to "Learn which Avenger you are!" or "Take this quiz to find out your soulmate's initials!" (quizzes whose primary purpose is to collect people's data, not really to inform them about fundamental life concerns). As users answer questions, they should become increasingly impatient with the length of the quiz.

When they are finally done answering the questions, the app uses the answers to fill a sentence describing the person’s portrait and uses this description as a prompt to generate an image with DALL-E (Mini) backend, made available by Saharmor. Users then see an image that is supposed to represent them, which usually gives them a good laugh. Once the excitement is over, users are expected to better understand how their data was used and the primary goal of this app, which is to highlight and document the various biases, stereotypes, and inequalities amplified by AI.

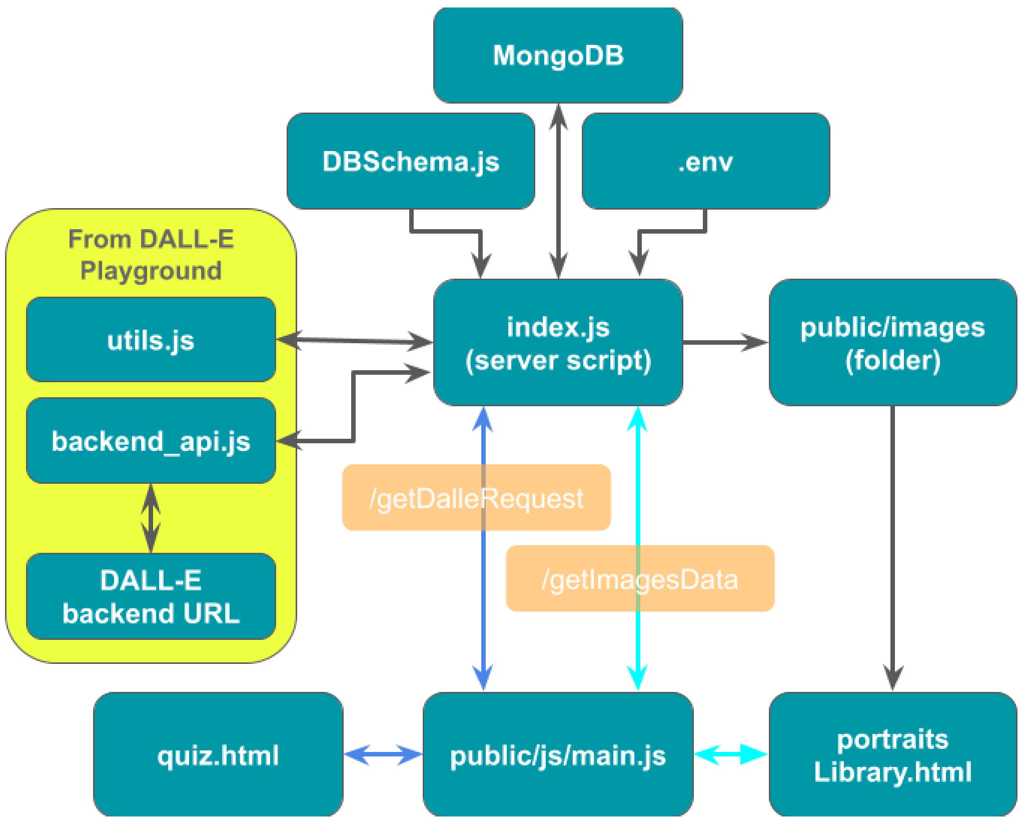
Indeed, an "About" section explaining what algorithmic biases are, where they come from and what their impact is will be displayed below the generated portrait. Users also have the option to report a bias in their portrait and enter descriptive keywords. Further developments will allow users to change some of their answers in the quiz and compare the resulting portraits to better understand the impact of each answer on the image and to help identify biases.

A portrait library containing all generated images is also accessible once the portrait is generated. Users will eventually have various options to filter portraits in this library, for example by specifying a certain answer to a question, or by choosing a specific keyword describing a reported bias. In addition, users will be able to report a bias in other portraits than their own.

Tests done so far have shown that many of the generated images are completely off or uncanny. However, these images are the ones that have generated the most excitement and interest from the testers. In itself, this misinterpretation of the person's description is a good demonstration of how analyzing data without context can lead to misclassifications, and how algorithms don't see people as the humans they really are, but rather as a set of statistics. I plan to highlight this in the "About" section.

Project Progress

Ultimately, the overall system should look like this:



What is done so far and missing components/features

- Index.js:
    - Creates a server using node.js built-in “http” module and express

framework
    - Establishes the connection with MongoDB using mongoose
    - Uses express middleware function “node-static” to make files from the public directory visible by the client
    - Uses Express to manage routes:
        - Runs handlePost() function when a post request is made to the /getDalleRequest url (request made from ajax function in main.js)
        - Runs handleImageData() function when a post request is made to /getImagesData url (request made from an ajax function in main.js)
    - handlePost() function:
        - Communicates with utils.js and backend_api.js to verify if the backend url is valid
            - If so, backend_api.js will send the text prompt to DALL-E backend and send back the generated image to index.js
        - Uses node:fs module to store images in the public/images folder
        - Sends generated image data to MongoDB using mongoose
        - Sends response (imgSrc, title, the number of reported biases and their associated keywords and the name of the person who generated the image) back to where the post request was made (ajax function in main.js)
    - handleImageData() function:
        - Communicates with MongoDB using mongoose to get back all of the images data sources, their associated description, the number of reported biases and their associated keywords and the name of the person who generated the image.
- main.js:
    - nextQuestion() function:
        - Changes display mode (block or none) of divs with quiz questions according to which question we are at
        - Stores answers to the questions in an array
    - getDalle() function:
        - Creates a description of the user’s portrait using answers to the quiz
        - Uses Ajax to make a post request to /getDalleRequest url and gets back the generated image data
        - Runs parseResponse(imgData) function when post request succeeds and passes the image data as an argument
    - parseResponse(imgData) function:
        - Appends the generated image to the results div in quizz.html
        - Displays buttons to allow users to go to the portraits library or to take quiz again
        - Displays an “About” section that explains what algorithmic biases are, where they come from and what their impact
        - Allow users to change some of their answers to the quiz and compare the generated images
        - Allow users to report a bias and input descriptive keywords

    - Uses Ajax to make a post request to /getImagesData url and gets back all of the images sources, their associated description, the number of reported biases and their associated keywords and the name of the person who generated the image.
- portraitsLibrary.html
    - Displays all images from the public/images folder with their associated data
    - Allows users to filter portraits by specifying a certain answer to a question, or by choosing a specific keyword describing a reported bias.
    - Allows users to report a bias in others' portraits and input descriptive keywords