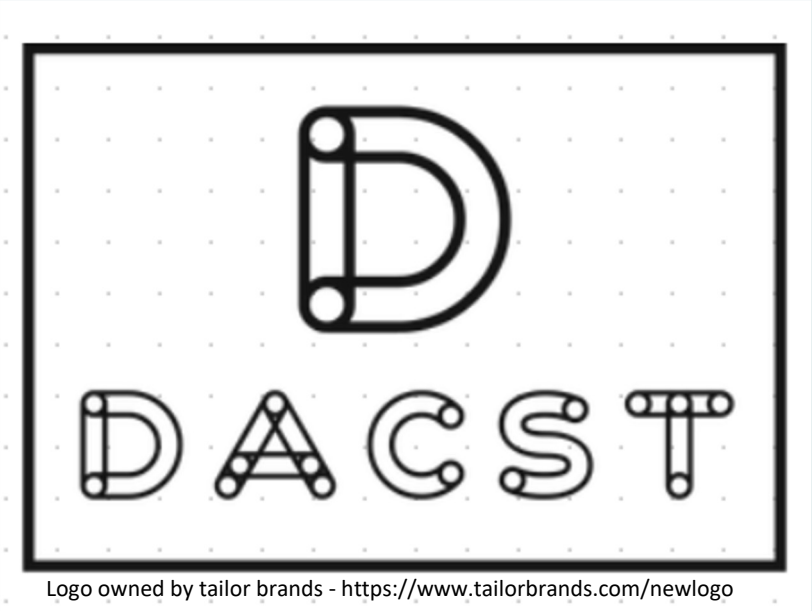


DACST Database Plans

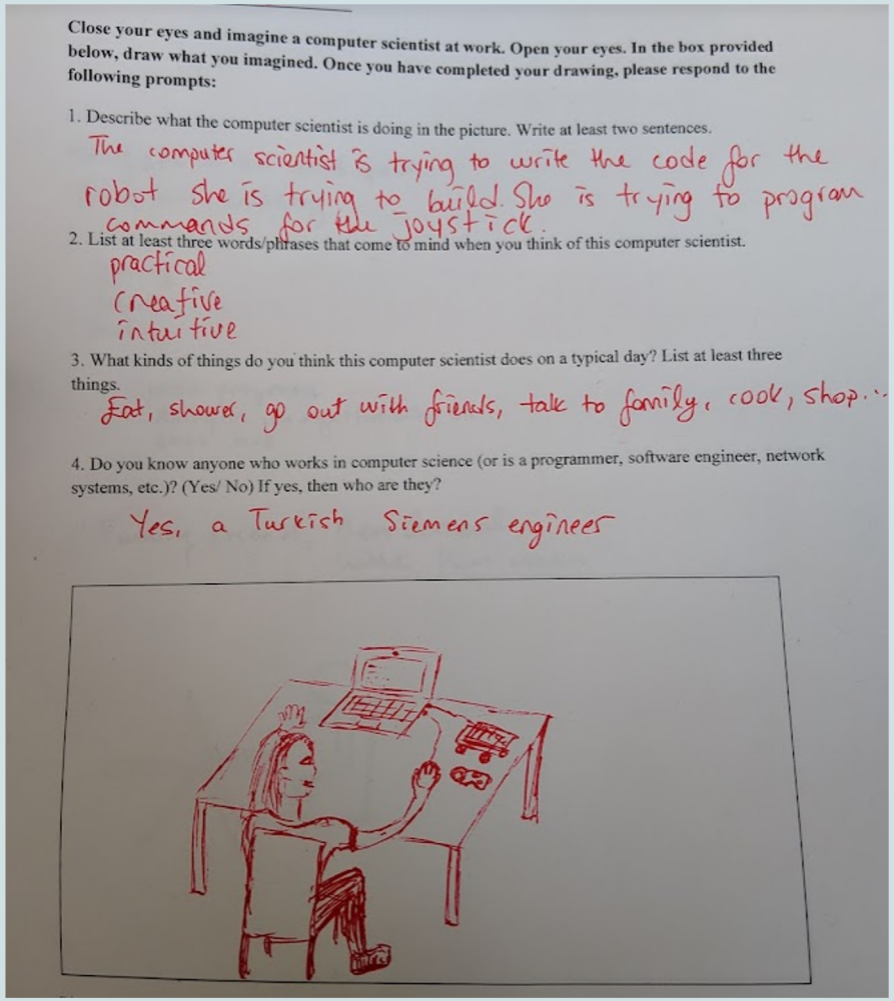
M. Cowan, E. Cohen, and S.A. Doore

Fall 2021

Project Description

- The product is being designed to host the data for a form of the “Draw-A-Scientist" Test (DAST) which is used to assess perceptions of scientists. The origin of this study is a paper by C.D. Martin in 2004 where she identified the harm in perceiving computer science as “hardware, software, and programming”.

- In this specific project the purpose is to understand students’ perception of Computer Scientists. The following information was collected from students:
  - A self-drawn image of a computer scientist
  - Text describing the picture and other information on computer scientists
  - Some student information (gender, class, semester)



Close your eyes and imagine a computer scientist at work. Open your eyes. In the box provided below, draw what you imagined. Once you have completed your drawing, please respond to the following prompts:

1. Describe what the computer scientist is doing in the picture. Write at least two sentences.

The computer scientist is trying to write the code for the robot she is trying to build. She is trying to program commands for the joystick.

2. List at least three words/phrases that come to mind when you think of this computer scientist.

practical
creative
intuitive

3. What kinds of things do you think this computer scientist does on a typical day? List at least three things.

Eat, shower, go out with friends, talk to family, cook, shop...

4. Do you know anyone who works in computer science (or is a programmer, software engineer, network systems, etc.)? (Yes/ No) If yes, then who are they?

Yes, a Turkish Siemens engineer

Goals and Scope

- Organizing this data in a well defined database will be useful for further analysis of this existing data as well as for storing new data in the same database

- Since users of the database will be a mix of skilled programmers and students the UI we created allows for easy interaction with the database (no need to learn a query language)

- We need a document-oriented database to manage the various types of data.

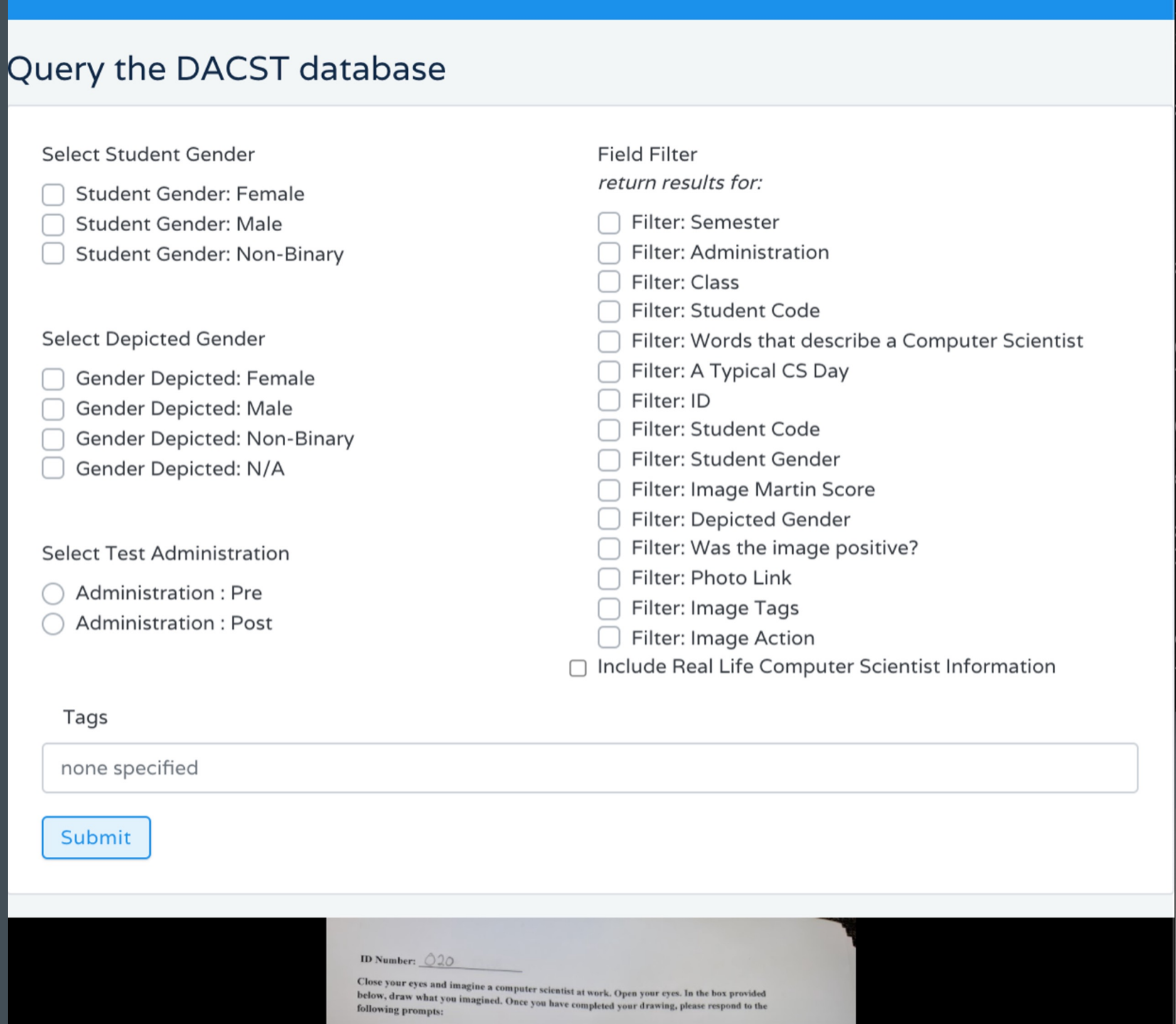
Team Member Responsibilities/Output

Because we were a small team we both worked on the backend and frontend. We both worked on researching MongoDB and structuring the database because neither of us had experience originally. Additionally, we researched website design because we had limited experience with that as well. Here are our specific contributions:

- Collected data from client and retrieved client view requirements
- Set up the github repository
- Created the website, including all functionalities (query, download, and connection with the database)
- Created google form that dumps responses into google sheets which can be connected to MongoDB for easy future data entry
- Input data initially through csv to json converter, so we wouldn’t have to do data entry manually
- Updated website aesthetics with a mobile-first design.
- Updated queries and download functionalities
- Implement “tagging” functionality (read and write to mongo from UI)

Client and User Overview

- Our client: Prof.S. A. Doore
- Users
  - Researcher needs:
    - Query the data by tags and/or filters
    - Download appropriate data as ".csv"
  - Student doing data entry needs:
    - easily add Tags to existing documents
    - bypass manual entry via google form/google sheets



E/R Diagram and Data Design

Data Design :

Key Value Pairs:

___*STUDENT*___

CODE (student_id)

GENDER

SEMESTER

CLASS

ADMINISTRATION

THREE WORDS/PHRASES

COMPUTER SCIENTIST ACTIVITIES

EMBEDDING 1 - FEW:

___*IMAGE*___

MARTIN SCORE

IMAGE LINK

IMAGE GENDER

POSITIVE IMAGE

PICTURE DESCRIPTION

TAGS

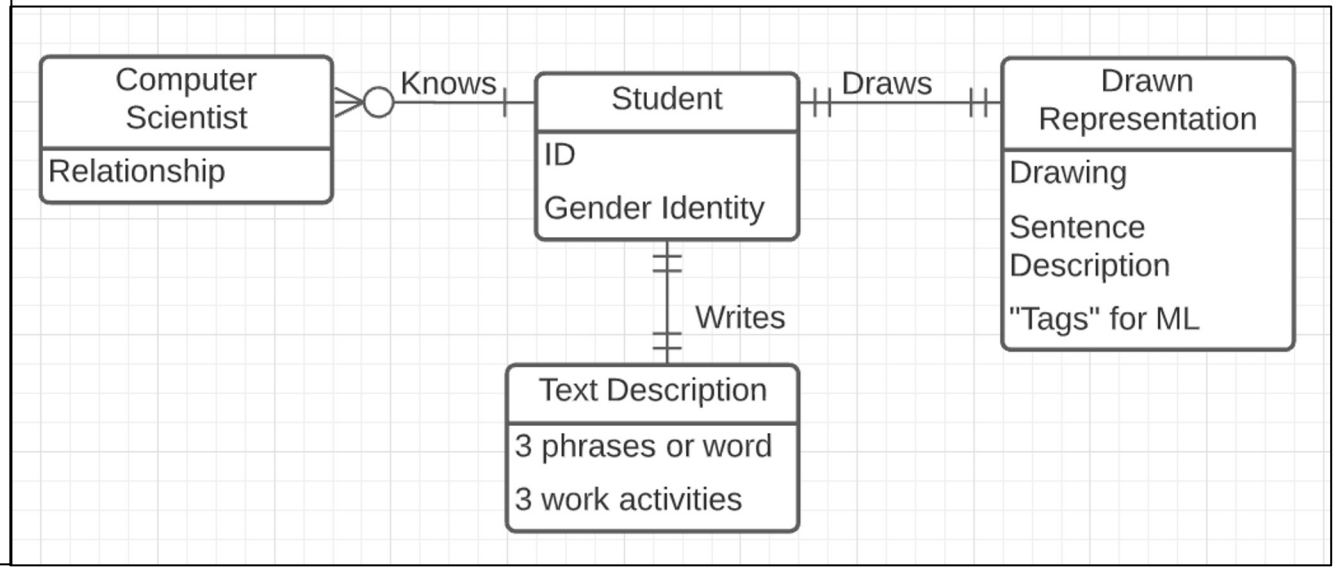
REFRENCING many/ infrequent data:

CS_ID

CS_ID

KNOWN COMPUTER SCIENTIST

CS INFO



SYSTEM FEATURE 1: Document Data Storage

Description

- The MongoDB database will hold text and image data, with associated tags for each response

Stimulus

- The stimulus for this feature is the input responses, and the response is the populated database

Functional Requirements

- MongoDB allows for the insertion, update, and deletion of responses in the database
- UI should facilitate these operations

SYSTEM FEATURE 2: Querying without MQL

Description

- The ability to search through the data for the desired information based on tags that are populated through the input data and based on filters

Stimulus

- The stimulus is searching by tags for keywords, and the response is the returned search results

Functional Requirements

- User should be able to query based on tags and filters

SYSTEM FEATURE 3: Export Data in ‘.CSV’ format

Description

- The user should be able to select data on a large scale and retrieve the information in a .CSV file
- This feature is important for analyzing stored data.

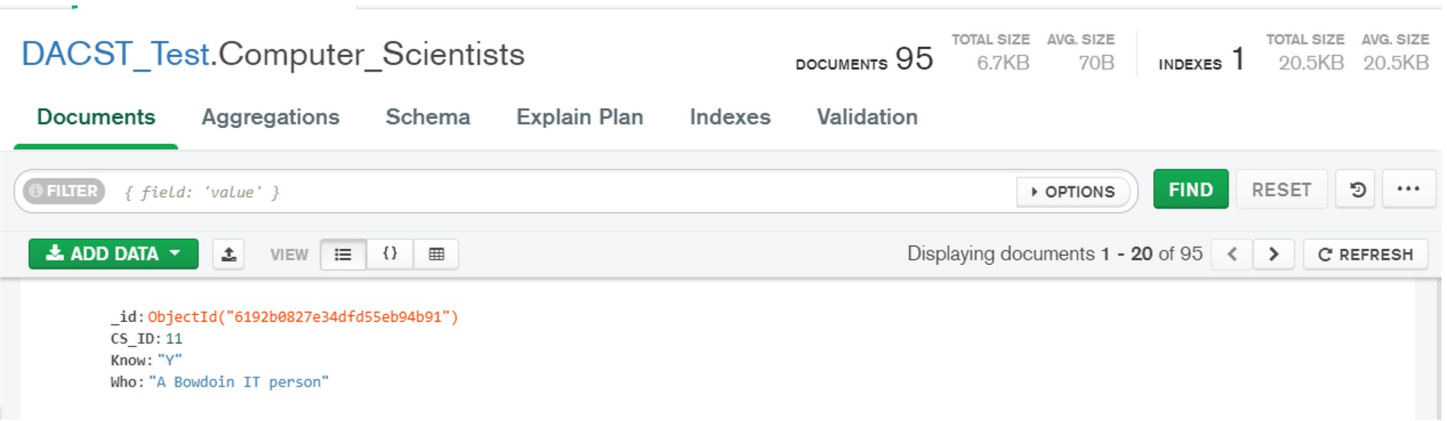
Stimulus

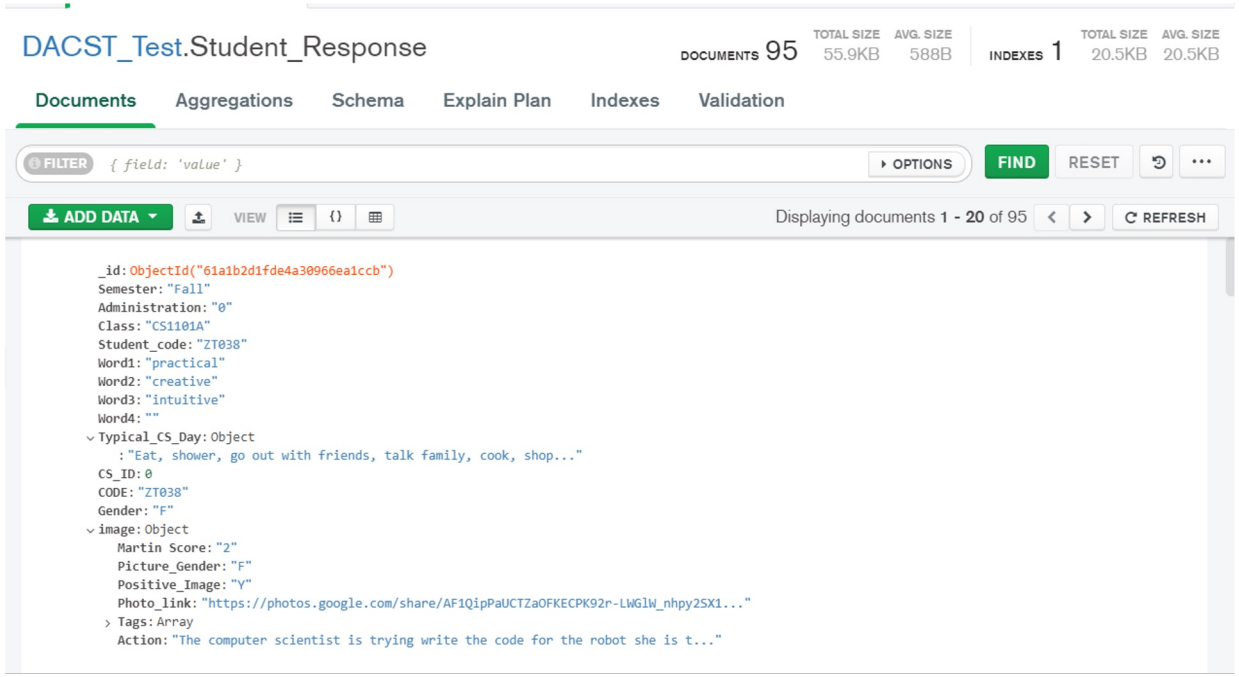
- The stimulus is the user selecting the data to be exported as ‘.CSV’ and the response is bundling the data in a new .CSV file that is then returned to the user

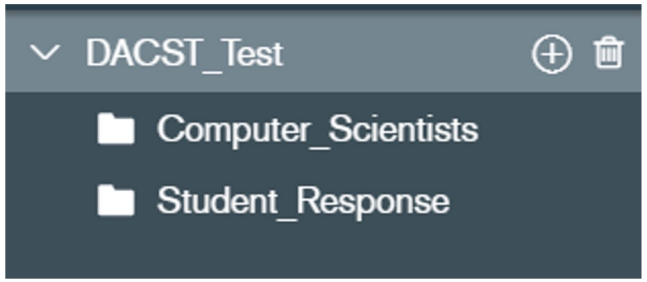
Functional Requirements

- Selected data is downloaded to downloads folder as a ‘.csv’

Database Implementation





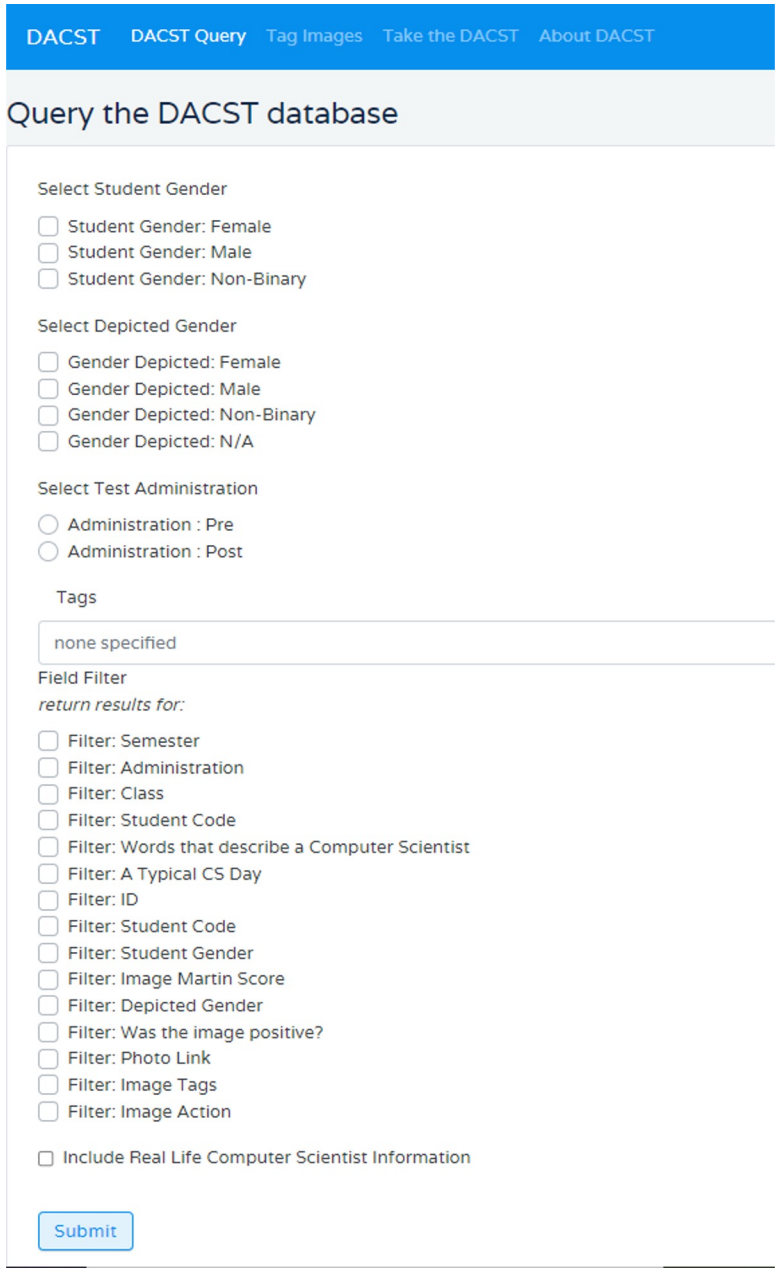


User Interface and Demo

[LINK TO UI (HEROKU)]

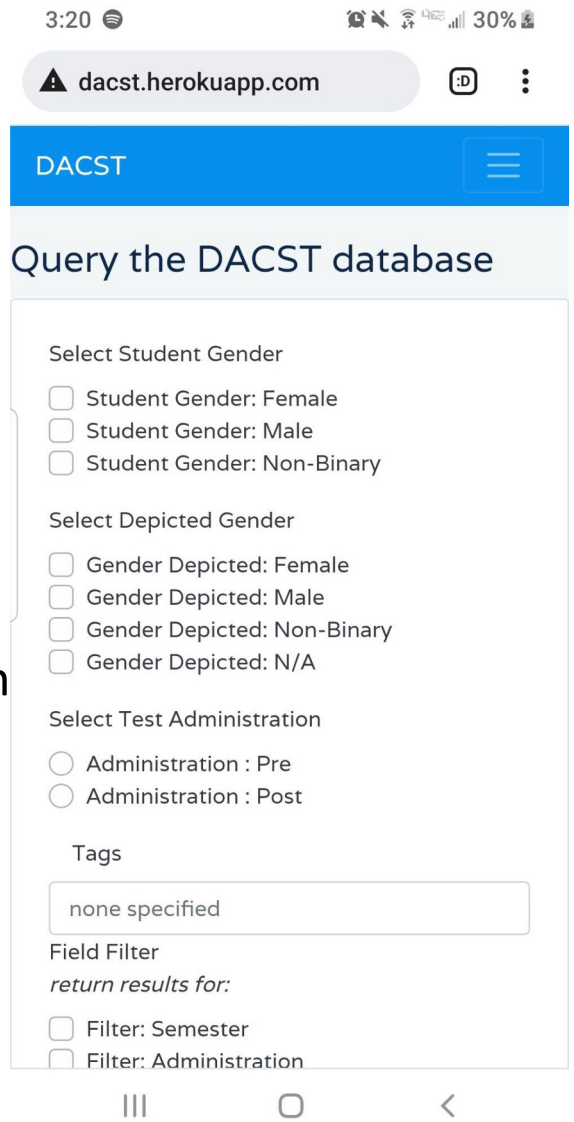
Tabs:

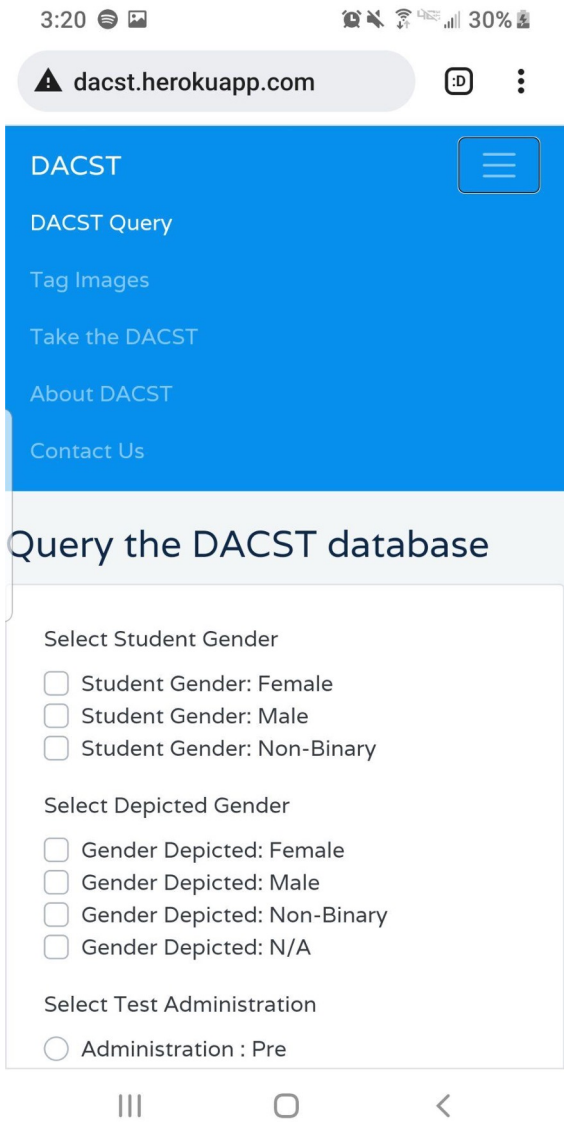
- DACST Query
  - Querying
- Tag Images
  - View/Add/Remove tags for each image
- Take the DACST
  - Link to the google form where you can take the DACST
- About DACST
  - Short description of what the DACST is

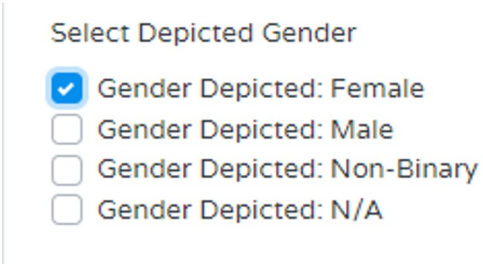


Inclusive Design Features (Hardware/Software)

- Color scheme
  - Simple, contrasting colors on website
- Mobile first design
  - Dynamic sizing for a variety of screen sizes
- Screen reader support
  - in the case that a section of the screen is not explicitly labeled, each element has an “aria-label”
- Check boxes
  - We implemented check boxes, versus a slider or an option to control click for multiple choices, because it was the most intuitive

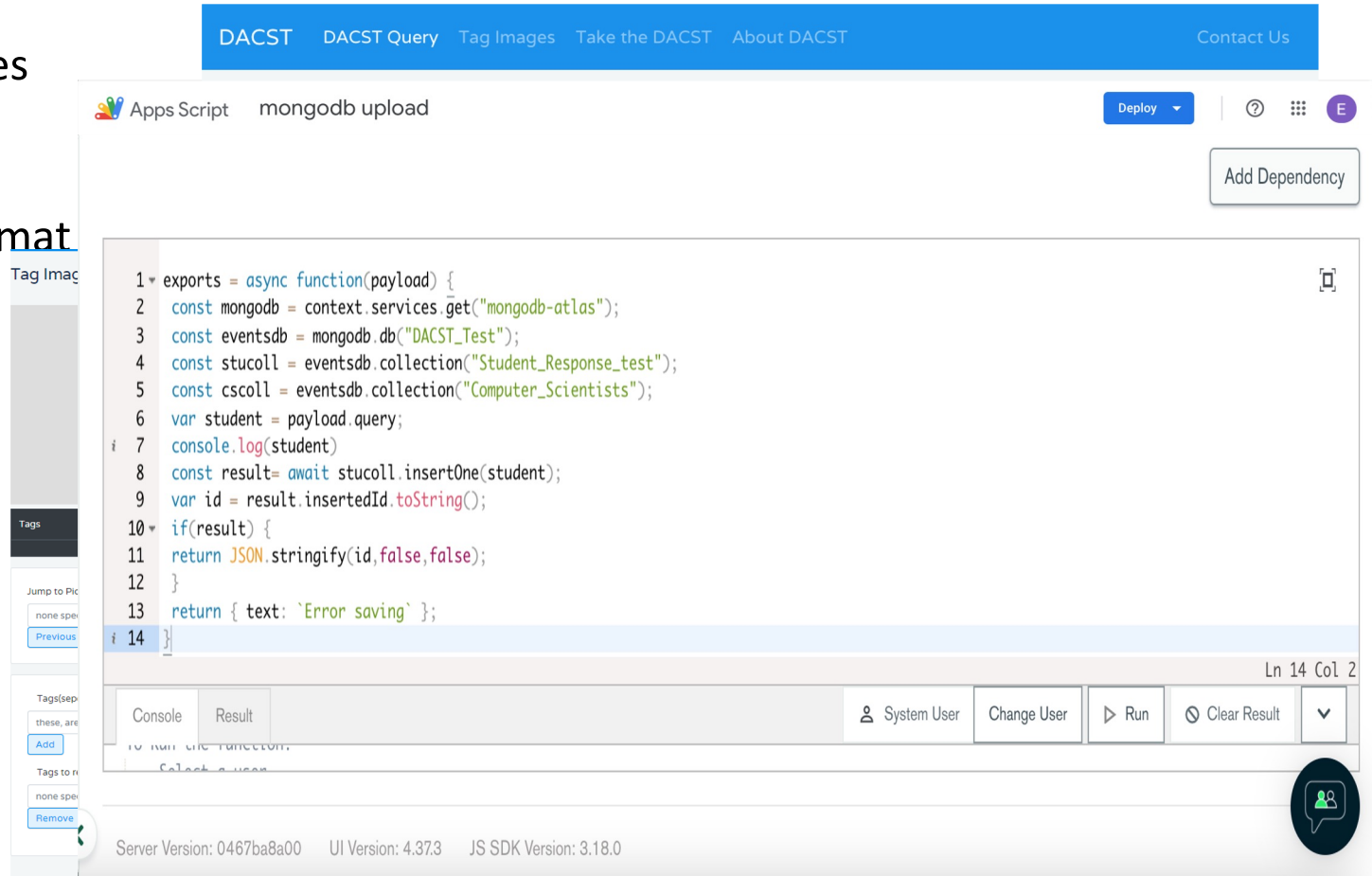






Design Constraints and Limitations

- The initial download button did not serve to user (download stayed on server side).
  - create an attachment object with the servers downloaded csv to serve to user
- Google photos doesn’t let you embed images
  - Include link to image for user
- Google Script language breaks JSON doc format for automatic uploads to mongodb
  - unsolved



Lessons Learned/Next Steps

- Lessons
  - Having two collections made implementing querying slightly more difficult
  - Connecting to a security minded company like google may not be possible
- Next Steps
  - Implementations:
    - Add more querying categories (is image positive?, less than | greater than | equal to martin score, Semester)
    - In tag image page : add martin score slider and “is image positive?” checkbox
  - Testing:
    - Test on screen reader device for more accessibility

Acknowledgements & Questions

Thank you Professor Doore for your help on this project!