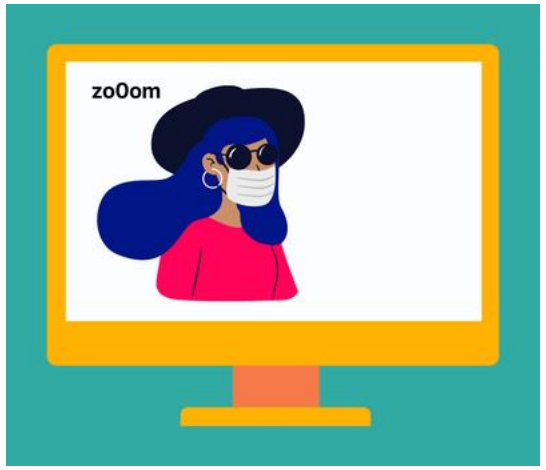
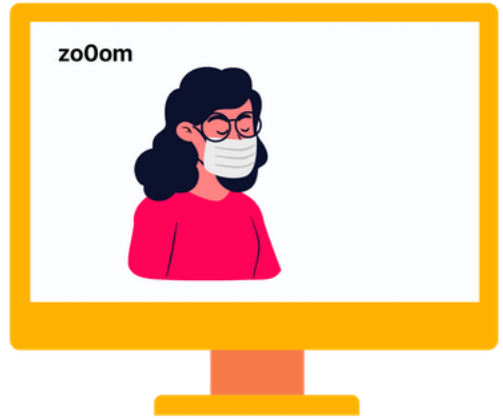
Emotion & Gesture Tracking Browser Extension for use in Classrooms

Caleb Duke, Allison Frye , Cassidy McManus,
Ryan Rau, Gage Robinson

The Team





- With the rise in popularity of online classes, students are finding it more difficult to stay engaged while attending lectures.
- The lack of in-person interaction leads to a decreased drive to pay attention to material being presented.
- Professors are also finding it difficult to gauge how students are doing in lectures due to a lack of visual and verbal representation of the student body.

PROBLEM

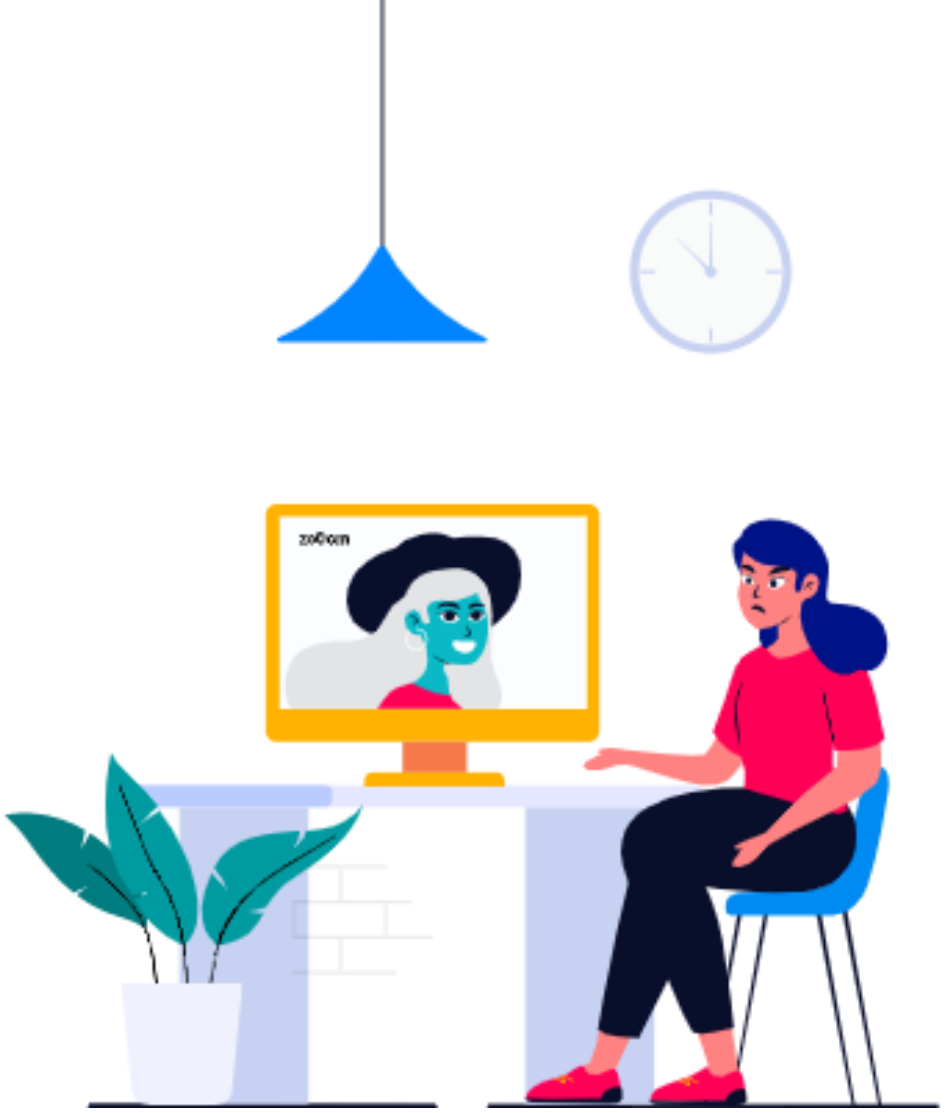
- This style of instruction does not make for the best environment for the students.
- Students often feel awkward being the only one with their camera and microphone on when they want to ask a question.
- Many students will likely perform worse in a course due to this radically different environment.

PROBLEM

- Without a proper solution to this issue, we will see a decline in academics and social skills.
- Many students feel like they are missing out on the college experience because of this new way of delivering lectures.
- By not having a concrete solution, we will lose motivation for holding a productive learning environment.
- We can help solve this issue by emulating human interaction in the online classroom.

PROBLEM

- Develop an application that allows for interpersonal connections in a classroom during virtual learning
- Bridge the current emotional and physical gap between students and instructors
- Use facial and gesture recognition to track student's facial expressions, emotions, and presence to provide live feedback of the student's activity for the instructor



OBJECTIVE

- Developers have largely relied on chat, microphone, and camera usage for in-session student feedback
  - This implementation can be seen in Zoom, Wimba, Adobe Connect, and Blackboard Ultra
  - Platforms may also allow students to display emoticons or raise their hand virtually
- Non-vocal student feedback is still largely under-represented in these platforms
  - While camera usage allows students to display posture and expression, they often share screen real estate with the Slides or other lecture aids
  - Chat and microphone usage are both online alternatives to vocal feedback
    - Facial expression is not preserved through these mediums
    - Usage can often bring lectures to a halt, unlike in-person non-vocal feedback
- How can our project address these issues?
  - Using Amazon's Rekognition, or another computer vision implementation, to analyze facial expression
  - As a Chrome plug-in, we plan to supplement existing platforms and use their current strengths

RELATED WORK

Student Requirements

- Allow students to login and direct them to a modal asking for a classroom password. LDAP may be incorporated.
- Ask students if they may have access to the camera.
- Allow students to use Zoom as usual.

Teacher Requirements

- Allow teachers to login, while generating a classroom code to send to students.
- Allow teachers to conduct a Zoom based class experience while being able to view live reactions.
- At the end of a session, a report will be generated to allow the teacher to analyze their students emotional state over the class period

Requirements/Use Cases/ Design Goals

Architecture: Overview

Students

Login into Zoom via Chrome and activate our extension and login

AWS Rekognition
or
PyTorch
or
Teachable Machine

Send emotional responses

Database

Send data to Reporting Site

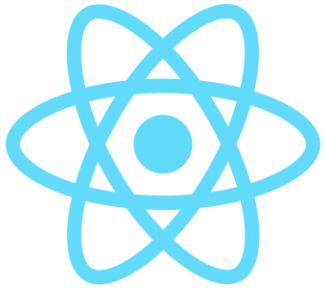
React App
(either embedded in extension or accessed via personalized link)

Final Report
Will display all statistics collected throughout lecture

Live Feed
Will display current emotions

- API, Database, Reporting Site
- Machine Learning (ML) Models
- Chrome Extension

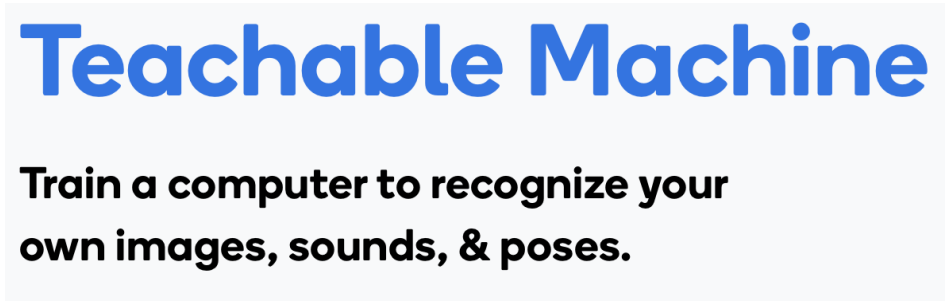
- Database
  - Stores if user is happy, confused, or left
  - MySQL
- APIs
  - Allows us to send emotions to database from AWS Rekognition, send values from database to reporting site
  - NodeJS or Flask (undecided)
- Reporting Site
  - Allows teacher to view current statistics and will generate final report after lecture ends
  - ReactJS



*all hosted on VM provided by the University

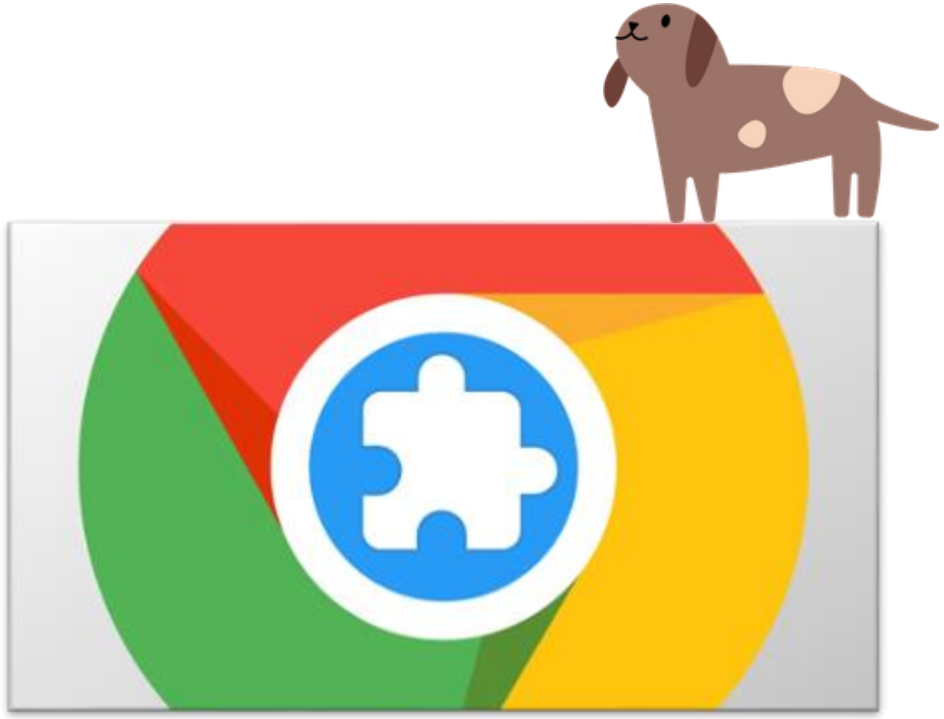
API, Database, Reporting Site

- Amazon Web Service's (AWS) Rekognition
  - Facial gesture recognition
- Custom Models
  - Google Teachable
  - PyTorch



MACHINE LEARNING MODELS

- Purpose
  - Allows user to sign into our app and show correct functionality (student/teacher) views
  - Creates an entry point to the reporting site
    - Two Options:
      - Have a React App embedded into the Chrome extension
      - Generate a link to a React App based on the teacher's ID



CHROME EXTENSION

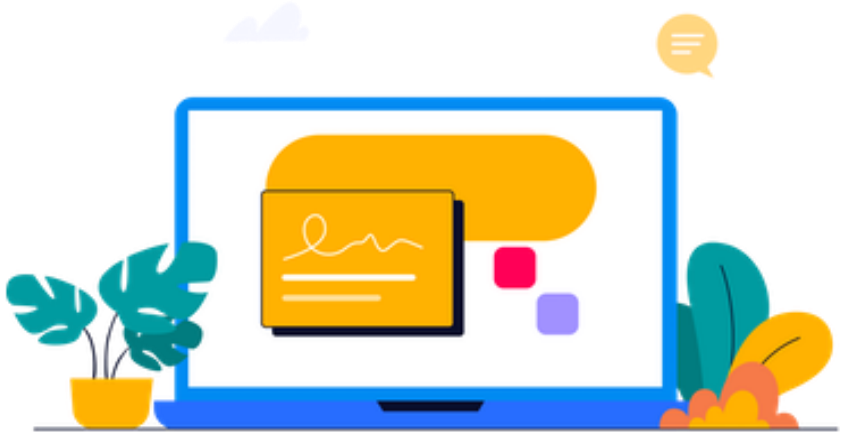
- Sprint 1: Preliminary Research
- Sprint 2: Initial Setup
- Sprint 3: Backend, Extension, ML Development
- Sprint 4: Merge Components Together
- Sprint 5: Expand ML & Start Reporting Page
- Sprint 6 & 7: ML Expansion & Additional Features
- Sprint 8: Final Touches, Report, & Optimization

SCHEDULE & TASKS

FACILITIES & EQUIPMENT

- Laptop with webcam (users)
- Access to AWS Rekognition (for our group)

- **Browser Extension**: Browser extension which has access to the user's camera and sends that data to our backend API. Extension will also provide entry point to the reporting site
- **Backend**: API + Database Schema Documentation: Document with all the APIs' endpoints, inputs, and outputs. Included will also be the languages used. In addition to API information, the database schema and language will be listed.
- **Machine Learning** (Emotion & gesture recognition) **models**: If we go with PyTorch or Google Teachable
- **Design Document**: Document explaining how the three components interact with one another. All technologies will be listed.
- **Final Report**
- **GitHub Repository List**: As we are taking the microservices approach, each component will have their own repository, thus a location in which all links are consolidated will be given.



DELIVERABLES