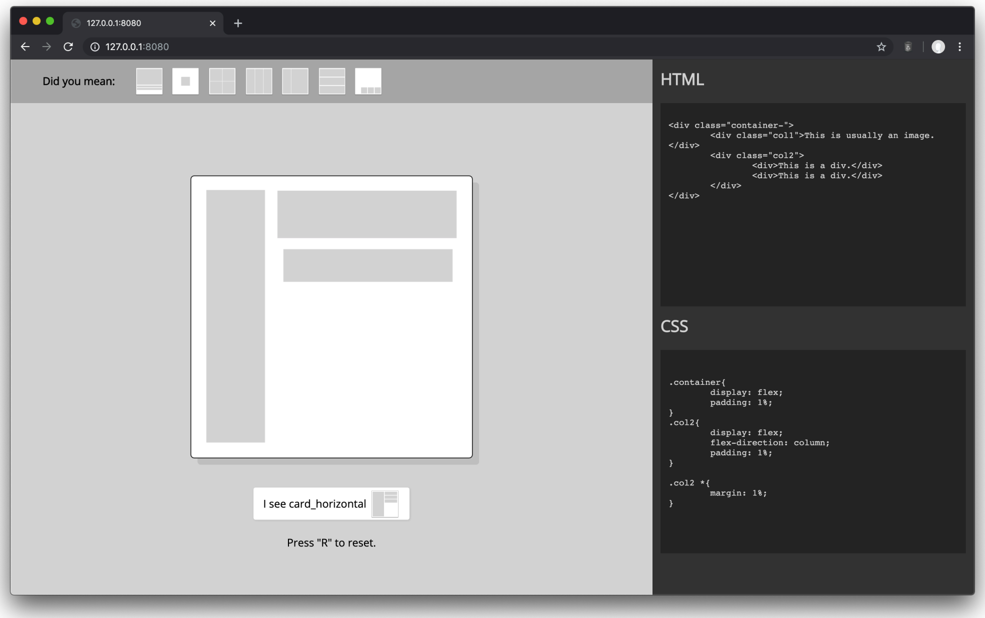
PUI Final/ Web-Dev-Pictionary

Supawat Vitoorapakorn

PUI Final: Web-Dev-Pictionary

Part 1: Pitch



HTML

```
<div class="container-">
        <div class="col1">This is usually an image.
</div>
        <div class="col2">
                <div>This is a div.</div>
                <div>This is a div.</div>
        </div>
</div>
```

CSS

```
.container{
        display: flex;
        padding: 1%;
}
.col2{
        display: flex;
        flex-direction: column;
        padding: 1%;
}

.col2 *{
        margin: 1%;
}
```

Like Pictionary, but for flex-boxes.

Front-end development is not easy. For both new students and veterans alike, there are millions of properties and tags out there, and choosing the right one isn't easy, especially when it comes to flex-boxes. While flex-boxes are used everywhere in the modern web, they are notorious for their crazy amount of properties for even the simplest design pattern.

I've created Web-Dev-Pictionary as a way to quickly and visually to overcome my struggle with flex-boxes. Sketch your design pattern, and the AI will give you the HTML and CSS code. This saves everybody time from trying to guess the right keyword to Google. Web-Dev-Pictionary AI is not meant to replace front-end developers but rather it's meant to augment developers and assist new students.

Part 2: Interactions and Usage

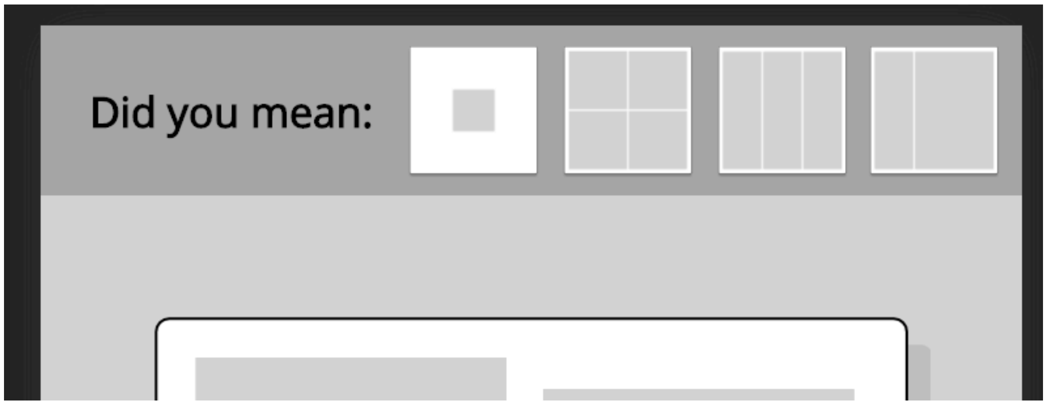
Interaction Type:
Click and drag to sketch items in divs.

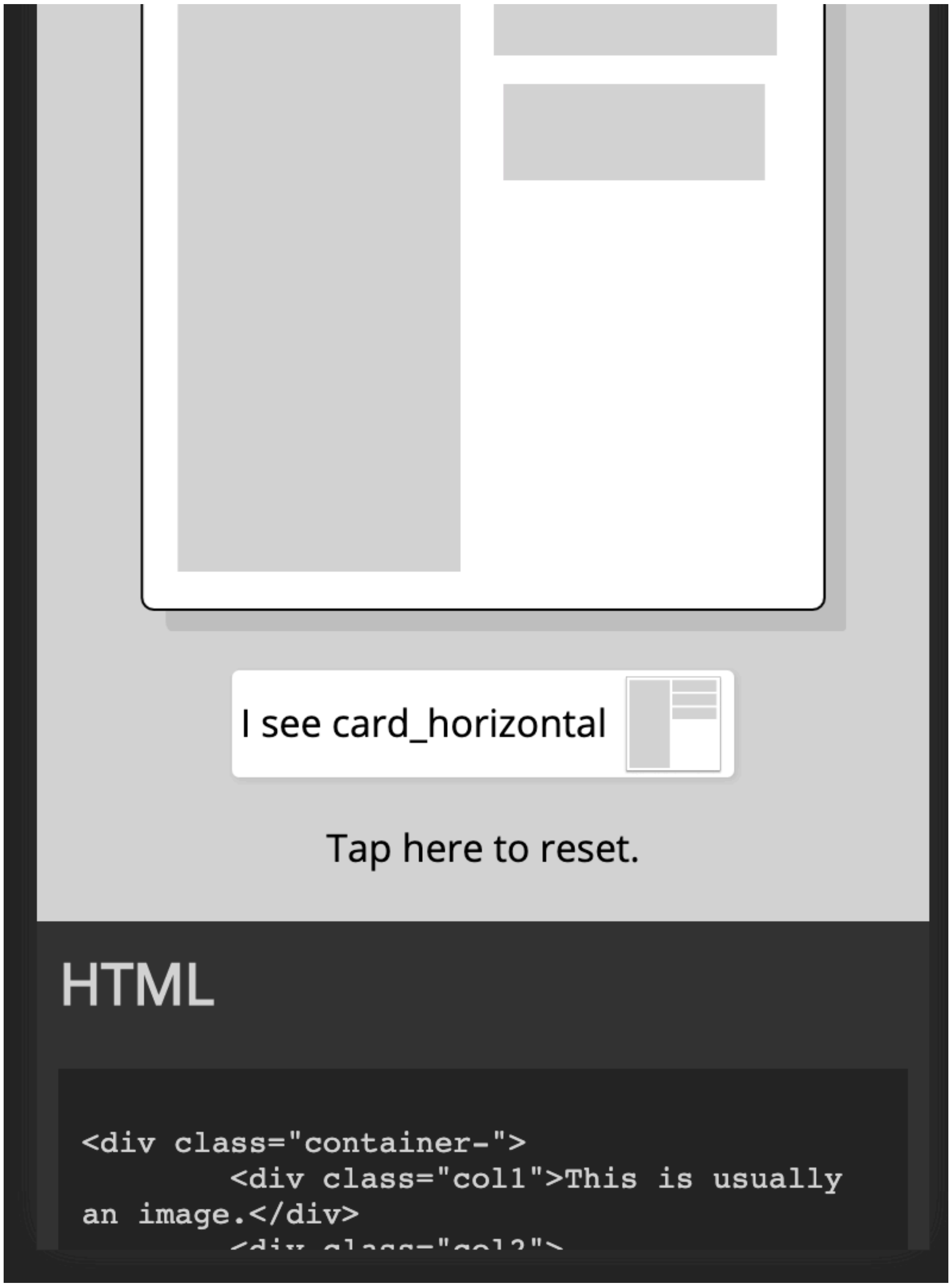
Case Study:
Know what a design pattern looks like, but don't know what it's called? Well... draw one! Draw cards, grids, columns, etc. The AI is pretty robust in recognition, currently its trained to recognize 9 common design patterns.

Part 3: Tools Used
List of Tools p5js - For allowing users to draw on a canvas in a constrained manner.
Google Teachable Machine - For recognizing what the user draws on a canvas.
Jquery - For quickly manipulating DOM element.
Jquery UI - For animations and delighting micro-interactions.

Part 4: Iterations
Machine learning is a "garbage in, garbage out" system. To reduce noise, I constrained the user to a pre-drawn div. This was also done in the interest of time.

Part 5: Challenges
1. Promises/Async Await: I have to learn the concept of "promise". Some weird thing about asynchronous, and callbacks I don't really understand still.
2. Responsive design while connecting p5Js with Google's AI. Training the ML was super easy, but connecting it with p5js's canvas was difficult because there were many syntaxes and concepts I am not familiar with: < canvas >, model.predict(), captureStream(), etc.





HTML

```
<div class="container-">
        <div class="col1">This is usually
an image.</div>
```

Responsive Design