Photobooth

- Part 1 was hard; well done!
- Part 2:
  - Tuesday: Add automatic Google Cloud Vision labels, filtering by label.
  - Favorites are 2 points extra credit (buttons can do nothing otherwise).
  - 25 points total, comprehensive.
  - TAs will do interactive grading.
  - Can be handed in late, by midnight 6/7, for 2 points off.

Exams

- Midterm 6/2; TAs will proctor it
- I will email you random seat numbers the night before, to your ucdavis.edu address
- Will cover today’s material
- If you are satisfied with your scores on the two midterms, you can skip the final
- As soon as your Photobooth and midterm are graded, I can give you your course grade (so far) so you can decide

Cloud Vision API – the magic

Google Web services

- The Cloud Vision API is one of many Google Web services
- To use it, you need a paying developer account on Google, and you’ll need to set that up with a credit card
- Instructions on Web site with Part 2 instructions

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>1.0-16.00 (USD/1000)</th>
<th>16.01-64.00 (USD/1000)</th>
<th>64.01-256.00 (USD/1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Detection</td>
<td>Free</td>
<td>21.00</td>
<td>21.00</td>
</tr>
</tbody>
</table>

Lots of ways to use GCV API

- We’ll use the very simplest
- Google issues you an API key (a code that identifies us)
- You include the API key in the URL of every GET or POST request you make:

url = 'https://vision.googleapis.com/v1/ images:annotate?key=??????????????????';
What are the security issues?

- Someone snooping on the internet can see the API key, and make expensive GCV calls that get charged to your card
- Not clear it would be worth it to anyone...
- Google has several other authentication mechanisms that avoid this problem, which we will not use

What are the security issues?

Using the API

- Follows the usual 4-step plan
  - Make up request
  - Set up callback
  - Send off request
  - Handle result in callback
- But as usual has it’s quirks

The HTTP request

- Use a POST request
- JSON in body gives information about request, and URL of image to analyze

Node request function

- To build a server HTTP request using Node, the usual way is to use the node request module
- npm request
- This gives us the request function
- The functionality here is exactly the same as using the XMLHttpRequest object in the browser, but because this is the Web everything looks different...

Where are the four parts?

request(

  url: url,
  method: "POST",
  headers: {"content-type": "application/json"},
  json: requestObject,

  // second operand is callback
  APIcallback
);
Callback function

```javascript
function APIcallback(err, APIresponse, body) {
    if ((err) || (APIresponse.statusCode !== 200)) {
        console.log("Got API error");
    } else {
        APIresponseJSON = body.responses[0];
        console.log(APIresponseJSON);
    }
}
```

Example response

```javascript
{
    labelAnnotations: [
        {
            mid: '/m/026bk',
            description: 'dance',
            score: 0.8921945
        },
        {
            mid: '/m/05qjc',
            description: 'performing arts',
            score: 0.87477195
        },
        {
            mid: '/m/06ntj',
            description: 'sports',
            score: 0.7928343
        },
        {
            mid: '/m/02jjt',
            description: 'entertainment',
            score: 0.7739482
        },
        {
            mid: '/m/02_5v2',
            description: 'quinceañera',
            score: 0.70231736
        }
    ]
}
```

How should we use this?

- Request to GCV goes in Step 3 of this sequence
- Photo upload handler function in server (case 3 in tripleThreatServer)
  - Be sure to put it after the image is fully uploaded (in the form.on('end', function() { .... }) function
  - Why?
- Callback for GCV goes where?

How should we use this?

- Request to GCV goes in Step 3 of this sequence
- Photo upload handler function in server (case 3 in tripleThreatServer)
  - Be sure to put it after the image is fully uploaded (in the form.on('end', function() { .... }) function
  - Why?
- Callback for GCV goes where?
- Also in the upload handler function, since it needs to send a response back to browser; it needs the other response object for the browser’s POST request!
Warning! TWO response objects

- Make sure they have two different, descriptive names, such as browserResponse and APIresponse
What to do on the browser side?

- Get browser response, extract labels
- Make image source be server URL, display unfaded
- Show labels
- Add onclick function to hamburger, deletion x’s

Faking it when not needed

- If you’re worried about racking up costs, we can have the function calling the API fake it when not needed
- Make a global Boolean variable “LIVE”
- When true, do real call with real callback

Fake call

```javascript
if (LIVE) {
    ...
} else {
    setTimeout(fakeAPIcallback, 2000);
}
```

- setTimeout() sets a callback that will be called in 2000 ms (2 seconds)
- fakeAPIcallback displays fake data in correct format

It’s all downhill from there!