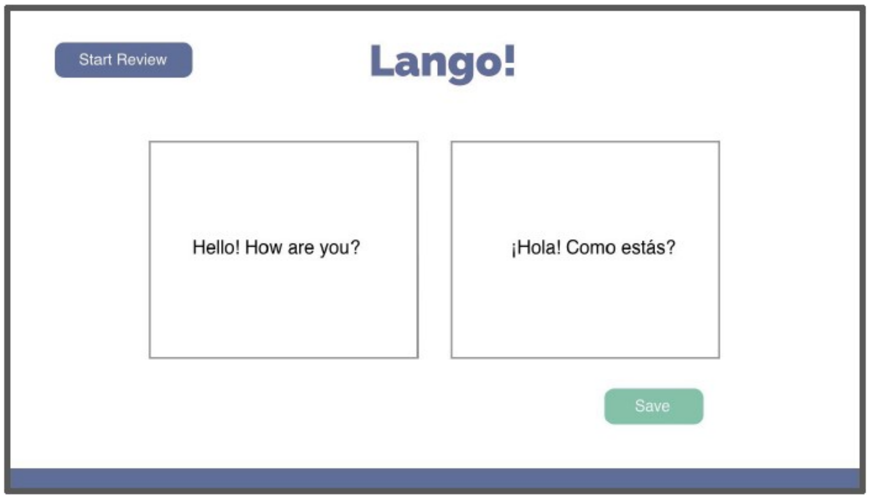
ECS 162 Spring 2019 Assignment 5

Please submit by 10PM, Thursday May 23, using Canvas.
We will accept submissions by teams of at most three. Everyone in the team should submit the same files, and will get the same grade.



In this assignment, we'll get started on our flashcard app. We'll do three main things: getting translations from the Google Translate API, setting up a database to store the flashcards, and starting a user interface with React.

Using the API

Let's get started with the API. First, you'll need to set up a project on the Google Cloud Platform and get an API key for Google Translate. Here are some instructions.

Next, let's try a sample request and response. On the server, from the command line, try running this sample code: testAPI.js (How do you run a Javascript program on the server? You know this.) You'll have to add your API key to make it work. You should see it display a phrase and its translation into Korean. To warm up, change it to use the language of your choice (if that is not Korean). Read up on the node response module so you understand what is going on here.

Now you're ready to teach your app to answer translation queries. When it gets a URL in this format:

```
  http://server162.site:port
  /translate?english=example phrase
```

it should return the translation in JSON, something like:

```
  { "English": "example phrase", "Korean": "예시 문
  구" }
```

How to accomplish this? When it gets the query, it makes a query of its own to the Google Translate API. And when the response comes back from the API, your server can send the response back to its browser.

Finally, for testing purposes, copy your palindrome Web page and make it translate the input, instead of making a palindrome. To set up for the next part, add a "Save" button.

Setting up the database

To set up the database, we have to write a node program that runs on the server, but which are not part of the actual Web server code. We'll use the sqlite3 module, which you have to install. Our database will have one table, with a row for every flashcard. Here is a starter program createDB.js, that sets up a flashcard database with one table. The table contains only one column, an ID number. Change it so that it includes

1. unique ID number (int),
2. English text (string),
3. translated text (string),
4. number of times shown (int),
5. number of times answered correctly (int)

The lecture on 5/10 will cover this. You should see the database file FlashCard.db appear in your server directory.

Next, let's load some cards into the database. Teach your server to respond to AJAX queries of the form:

```
http://server162.site:port/store?english=example
phrase&korean=예시 문구
```

by storing a flash card into the database. Times shown and times answered correctly should both be zero. Return a HTTP response with an empty body, to let the browswer know everything went well.

Finally, have the browser send the store request when the user hits the "Save" button. You should now be able to make and save cards. To test your program, you can use this function, which prints out the whole database (or look at it using sqlite3 from the command line).

Front end with React

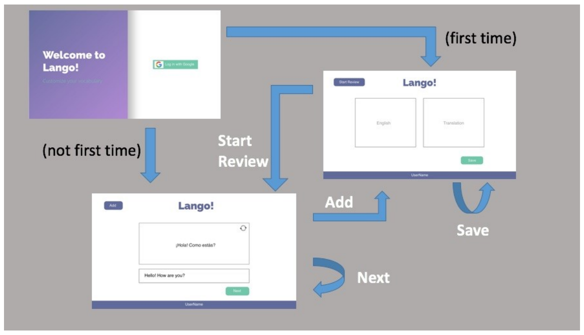
Now, let's get working on the UI. Here are are Jamie's complete designs, and here are her slides.

We will use React for the whole UI for this project. Here is my guide to a minimal React setup that should work with our server. The React documentation is very good, particularly the guide to the main concepts. You could start with this quick intro.

You are welcome to produce your production .js file or files using some other development process, and the TAs might be able to help you with that. However, we expect your project to run entirely on your server, not some other server that you installed.

Your HTML file should look almost exactly like the example from lecture, with the body containing a single div and possibly script tags. The entire DOM should be constructed using React. Your .css files will be exactly the same as if we were not using React.

I recommend you begin with the card creation page, and get it working with the AJAX requests and server code developed in the earlier steps. This is sufficient for this assigment, but if you want to move on to the flashcard review page as well, feel free.



Here is the behavior we are expecting. Entering English text and hitting return on the card creation page should update the translation. Pressing save should save the card to the database. Pressing start review should move to the flashcard review view; this should not be a new HTML page, just a re-drawing of the current page with different state.