CS 490: Software Engineering
Project 1 Specifications

September 16, 2020

1 Description

Welcome to your first project for CS490!

In this project, you will set up the main toolchain for the class and use it to build and deploy a simple web app that shows information about your favorite recipes and related quotes. These resources will be dynamically generated from external services.

2 Submission Information

2.1 When

- Milestone 1 is due (including filling out the Milestone 1 Google Form) before **September 22nd, 2020 at 11:59pm**. You must submit a screenshot of your app in the Cloud9 IDE as well as your Github link to get full credit.
- Milestone 2 is due (including filling out the Milestone 2 Google Form) before **September 29th, 2020 at 11:59pm**. You must submit your Heroku deployment as well as your Github link to get full credit.

2.2 How

You will turn in this project by filling out the following Google Forms:

1. Milestone 1: https://forms.gle/PPuYen6z8Y1jy34B6
2. Milestone 2: https://forms.gle/HCyq7hqmUjSULLF1A

3 Goals

1. Set up tools

   The primary goal of this project is to set you up with a common web development toolchain, and one that we will be using for the first half of the class. In particular, the main tools we will be using include: Flask (in Python), Git and GitHub, Heroku, and AWS Cloud9. Mastering the basics using these technologies will not only be required to complete the rest of the course, it will also enable you to hack on and build more projects of your own.

2. Familiarize with external APIs

   The secondary goal is to get you familiar with using external APIs. Specifically, we will be combining the Twitter and Spoonacular REST APIs in this project. A large portion of modern software engineering is building on and reusing others' work and you will of course do the very same in CS490!

4 High-Level Requirements

4.1 Create an AWS Cloud9 workspace

To start developing, you'll first need to log into your AWS Educate Cloud9 IDE. If you have not yet set this up, please do this immediately by following the directions on Canvas.

4.2 Track your work in Git

This will be needed both for deploying your app as well as submitting your code. We will be doing a crash course in class but check out these tutorials for more:

- https://try.github.io/
- https://learngitbranching.js.org/

4.3 Build your web app

After Milestone 2, the app should *at the minimum* display the name of a recipe, link to the original source for that recipe, show an image of the dish, and include a relevant twitter quote. Each of these items should be retrieved dynamically using external APIs (namely, Spoonacular and Twitter), and every time you refresh the page, a new random recipe should be loaded. You may hardcode names of dishes, but the actual recipes and tweets should be fetched dynamically using the APIs. For example you can hardcode "ice cream", "tres leches", "rasmalai", etc, but the Spoonacular and Twitter APIs should fetch relevant results based on those keywords. See below for more information and detailed technical specs.

Be creative when building this project! Twitter is a huge corpus of text, and Spoonacular, of recipes. Both their APIs provide massive functionalities and filters. Play with them to get a consistent set of quotes and recipes that will match a theme that you are passionate about (e.g. "chana masala", "sweet potatoes", or "dessert"). Have fun while building this project, and make something you're ultimately proud of, and would display on your résumé.

5 Technical Specifications

5.1 Milestones

There are two milestones to Project 1. The first milestone will allow you to get basic HTML, randomization of tweets, source control, and the Twitter API working. The second milestone will allow you to get the Spoonacular API working, deploy on Heroku, and beautify the application using CSS/HTML. The first milestone will handle much of the initial set up so that you do not need to worry about technical difficulties while implementing the more difficult Milestone 2 objectives.

Milestone 2 is significantly harder than Milestone 1, so I highly recommend getting started with Milestone 2 early!

5.2 Minimum Viable Product

To earn 100% of the technical points on this project, your project must satisfy the following criteria. Milestones are indicated with [M1] or [M2].

- [M2]Uses the endpoint "/" (the main one) to serve a page where dynamic quotes and recipe information are shown. This page meets all of the following criteria:
  - Quotes and recipe information are fetched via the above APIs, not stored in advance, and are random for at least 7 page loads.
  - All quote and recipe information as described on rubric appear on the page.
  - Attribution link to recipe (directs to original website (not Spoonacular) appears somewhere on page and is clickable.
  - Quotes and attribution links are legible for at least 7 page loads; you may need to adjust things such as font family, font size, text color, text location, and CSS3 filters like background brightness and blur to meet this.
  - If you use a background image, it is easily seen (the image can be stretched if needed). Feel free to hard-code dimension numbers.
- [M1] Is written in Flask (https://damyanon.net/getting-started-with-flask-on-cloud9/, http://flask.pocoo.org/docs/0.12/quickst
- [M1] Fetches quotes dynamically using Twitter's timeline or search REST APIs (https://dev.twitter.com/rest/public, https://dev.twitter.com/rest/public/search)
  - Documentation on authentication:
    https://dev.twitter.com/oauth/application-only.
  - For documentation on OAuth authentication over HTTP, check out https://2.python-requests.org/en/master/user/authenti 1-authentication.
  - You may also use a wrapper library like http://www.tweepy.org/.
- [M2] Fetches recipe data dynamically using Spoonacular's REST API (https://spoonacular.com/food-api/docs).
  - Getting started:
    https://spoonacular.com/food-api/docs#Authentication.
    Make sure you follow directions to "authenticate" via an API key!
- [M1 and M2] Passes in the correct URL and parameters for both API calls.
  - You should use the HTTP API python library "requests".
- [M2] Stores and references API keys, secrets, and login credentials using Heroku config vars (https://devcenter.heroku.com/article vars)
  - Make sure they're not checked into code to avoid a 20% penalty!!
  - See http://softwareengineering.stackexchange.com/a/182074 for more information.

5.3 Penalties

The following scenarios will result in an overall penalty towards your Project 1 grade.

- **API Keys checked into repository at any time, even if you delete them later** will result in a 20% deduction from your Project 1 grade.
- **Any type of user-visible error page on Heroku that you didn't make related to API calls (even if the call didn't return anything or you're over your daily limit)** will result in an automatic 10% deduction from Project 1 grade. Make sure this doesn't happen by using default values for the recipe, image, and tweets in case there is an error fetching them!
- **Turning in Project 1 late, even by a minute** may result in the following penalties:
  - 1 minute to 24 hours late: 20% deduction from your Project 1 grade
  - 24 hours and 1 minute to 48 hours late: 40% deduction from your Project 1 grade
  - 48 hours and 1 minute late or longer: Automatic 0 as your Project 1 grade

5.4 Stretch Features

The following features are strictly optional, and will count for no credit until you satisfy all of the above requirements.

Stretch features will be assessed for extra credit points, and there is no cap to the number of points you may earn through stretch features. These are merely suggestions of features to implement, but you can earn extra credit through implementation of any additional features. In fact, thinking outside the box and implementing useful features not mentioned in this spec is probably more likely to earn more extra credit than just implementing the below features! Please feel free to run your ideas of additional features by the instructor, TA, and classmates by posting in #project1.

Note that most of these stretch features involve technologies and APIs that we have not covered (and in some cases will not ever cover) in class. The basic stretch features listed below can be done with JavaScript in Flask, which you can learn more about here:
https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-xx-some-javascript-magic
Some ideas of stretch features may include:

- [Backend] Static/Dynamic Search
  1. [Basic] User has the ability to enter a dish in a search field (for example, "sweet potato casserole"). Once they press enter, the site will use relevant results from Twitter and Spoonacular to display a recipe/photo/tweets about that dish.
  2. [Intermediate] User has the ability to start typing letters into a search field (for example, "sw"). As the user types, autofill answers will populate suggestions of dishes (for example, "sweet potato casserole", "swedish meatballs") that the user may click on.
  3. [Advanced] User has the ability to enter the name of an ingredient in a search field (for example, "chickpeas"). Once they press enter, the site will search Spoonacular for dishes with that ingredient and display one randomly (for example, "chana masala").
- [Frontend] Content Carousel
  1. [Basic] Pictures and tweets will swap out every few seconds for the dish. There is no animation.
  2. [Intermediate] Pictures and tweets will swap out every few seconds for the dish using a fade in/fade out animation.
  3. [Advanced] Multiple pictures are laid in a collage view, and multiple tweets are featured as a word cloud.
- Your suggestions here! Anything goes!

6 Deployment and Submission

6.1 Deploy your app using Heroku

This will launch your app to the world. See documentation here: https://devcenter.heroku.com/articles/git. Optionally, come up with a creative app name related to your theme!

6.2 Submit your code via GitHub

You'll need a GitHub-hosted repository to turn in your code; see documentation at https://help.github.com/articles/create-a-repo/. Ensure that:

- Owner is the course organization (https://github.com/NJIT-CS490).
- Name is project1-<your-email>, where <your-email> should be replaced with your email (sr66 should be project1-sr66, for instance).

You are required to incrementally keep GitHub up to date, but you only need to have 3 pushes to GitHub via Git for each milestone in this project. Regardless, if you're doing things correctly, you will probably have at least 10 commits and you'll be pushing code continuously!

6.3 Document your work

Your code submission must include a file named README.md at the top level. The file should be in Markdown (https://github.com/ad p/markdown-here/wiki/Markdown-Cheatsheet). The README should thoroughly detail all technologies, frameworks, libraries, and APIs you used for this project, and explain how someone who forks the code would be able to get set up on the project. Your README must also answer the following questions:

- What are at least 3 technical issues you encountered with your project? How did you fix them?
- What are known problems, if any, with your project?
- What would you do to improve your project in the future?

7 Rubric and Grading

As detailed in the course syllabus, this project will be worth 10% of your final grade. Those points break down as follows. Each feature will either be due during Milestone 1 or Milestone 2 (as indicated by [M1] or [M2] below).

- [M1 and M2] 5 points: Setup/Upkeep
  - 3 points: GitHub repository has at least 3 substantial, spaced-out pushes from Git at least 1 day apart each, each with descriptive commit messages detailing what was changed
  - 2 points: GitHub repository is named by your email (project1-sr66)
- [M1] 15 points: Web App Functionality: Twitter

  - [M1] 3 points: server-side component of web app is written in Flask in Python
  - [M1] 2 points: quotes related to specific foods are dynamically generated via Twitter's REST API per load
  - [M1] 5 points: quotes are different across 7 page loads, but relevant to 7 different foods
  - [M1] 5 points: quotes display metadata including all of the following: tweet contents, author, date, and time.
- [M2] 30 points: Web App Functionality: Spoonacular
  - [M2] 15 points: recipe data are dynamically generated via Spoonacular REST API per load
  - [M2] 5 points: quote and recipe pairings are different across 7 page loads
  - [M2] 4 points: quote and recipe pairings form consistent theme across 7 different loads
  - [M2] 1 points: recipe title appears on page
  - [M2] 2 points: recipe serving/prep time appears on page
  - [M2] 2 points: image of recipe appears on page
  - [M2] 2 points: ingredients of recipe appears on page
  - [M2] 2 points: link to recipe page on appears on page
  - [M2] 2 points: any and all links are fully visible for at least 7 page loads
- [M2] 10 points: HTML/CSS Styling
  - [M2] 10 points: HTML/CSS used to "beautify" page. Your page doesn't have to look like the example, but it should look reasonably well styled. This can include rounding corners, laying out DIVs to separate tweets from recipes, adding a background, creating animations, using custom fonts, etc. Be creative in how you make your application look better - there is no right or wrong way as long as it's aesthetically presentable!
- [M2] 10 points: Deployment
  - [M2] 10 points: web app is deployed via Heroku
- [M1 and M2] 10 points: Code Styling
  - 10 points: code is styled well and organized
- [M1 OR M2, not both] 50 points: Code Explanation (Discussion with Instructor/TA)
  - 50 points: after either Milestone 1 or Milestone 2 (not both), you will get a calendar invite to a meeting where you will need to explain your code so far (e.g. if 5 lines/sections are deleted for 10 points each, explain what they did and why you need them).
- [M1 and M2] 30 points: Documentation
  - 10 points: README.md explains ALL STEPS of how to deploy the app and what technologies to install
  - 10 points: acknowledgement of at least 2 known problems and how to address them if you had more time in README file. If no known problems exist, what additional features might you implement, and how? Note that saying "I'd use Javascript" does not count for credit - we're looking for detailed explanations of what technologies, what files, endpoints, method names, etc.
  - 10 points: detailed description of 3+ technical issues and how issues were solved in readme file. These should be more descriptive than "I debugged it" - talk about your process, what you searched, link to useful resources that helped you, etc.
- [M1 and M2] Penalties (Deducted points)
  - -20% of Project 1 grade: API Keys checked into repository at any time, even if they are deleted in a later commit
  - -10% of Project 1 grade: Any type of user-visible error page on Heroku that you didn't make related to API calls (even if the call didn't return anything or you're over your daily API limit)
  - -20% of Project 1 grade per day: Turning in Project 1 late, even by a minute may result in the following penalties:
    * 1 minute to 24 hours late: 20% deduction from your Project 1 grade
    * 24 hours and 1 minute to 48 hours late: 40% deduction from your Project 1 grade
    * 48 hours and 1 minute late or longer: Automatic 0 as your Project 1 grade

  Submitted projects that do not run, are private, or not in the org folder will be deemed late until they are fixed (20% off per day)

8 Common Pitfalls

8.1 I can't access Cloud9 app in browser (Oops page)

Chances are, you will need to bind to port 8080 in your Flask configuration (https://damyanon.net/getting-started-with-flask-on-cloud9/). You may also want to make sure you are running your app (by running `python name_of_app_file.py` in your terminal).

8.2 I made some changes and saved in Cloud9, but they're not showing up on the preview

Chances are, you will need to restart your webserver. If you don't want to have to do this every time you may run your app in debug mode (http://stackoverflow.com/a/17322636). If that doesn't work, your browser may be caching your app. Hard refresh your preview tab (not your AWS tab).

8.3 I renamed my project and now Heroku is broken

Heroku app names are stored in two places; the first is in their database so they can show you your apps dashboard, and the second is in your local Git repository as one of the remotes. If these two don't match, then chances are things will be broken. Please read through this for details on how to fix the problem:
https://devcenter.heroku.com/articles/renaming-apps

8.4 [Errno 98] Address is already in use when I try to start my web server

Web servers all need to listen on the same port (port 80), so if you see this, then chances are you already have another web server running in a tab somewhere. You can only have one at a time! If you can't find another one running somewhere, chances are it may be running in another tab and/or browser. The first answer at https://stackoverflow.com/questions/19071512/socket-error-errno-48-address-already-in-use may be helpful.

8.5 I tried loading my deployed app on Heroku but the page is blank/an error page

You'll need to debug this by looking at the app logs from Heroku. To view the logs, simply `cd` into your web app's directory in AWS (yes, AWS), and run `heroku logs --tail` in your Cloud9 environment (this command is also seen on your Heroku page that errored out).

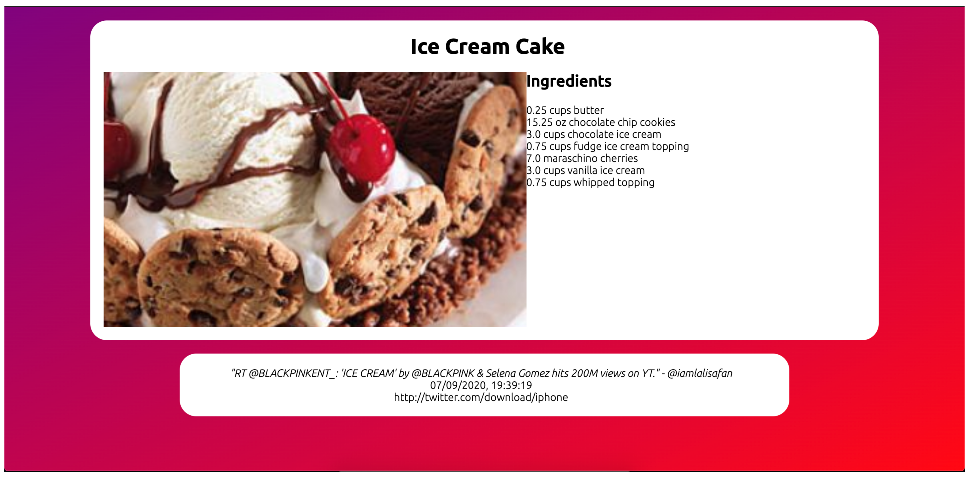
9 Exemplar Submission

I created this project and deployed it on Heroku myself. However, it is **not complete and would not earn full points**. For example, the project is currently missing a link to the recipe page, there is no recipe serving/prep time, and the margins between the recipe and the image are too small. That being said, this screenshot is meant to provide you an example of what a nearly finished project might look like. **Your project does not need to mirror this one - you can get full points for making aesthetically different front-end design choices!** This is simply meant to give you guidance if you are stuck, but I urge you to use your own creative muscles so that you can develop your own eye for design and product.

Note that in this screenshot, you will not see that a different recipe/tweet is loaded every time the page is refreshed.

By the end of the first milestone, you only need to have the bottom text box completed, with the Tweet, author, date, and time. This does not need to be beautified or styled via CSS, but that will save you significant time for Milestone 2.

By the end of the second milestone, you must complete the entire application, including the recipe (with title, ingredients, image, URL to the recipe, and serving size), and the tweet (with contents, author, date, and time).



Ice Cream Cake

Ingredients

0.25 cups butter
15.25 oz chocolate chip cookies
3.0 cups chocolate ice cream
0.75 cups fudge ice cream topping
7.0 maraschino cherries
3.0 cups vanilla ice cream
0.75 cups whipped topping

"RT @BLACKPINKENT_: 'ICE CREAM' by @BLACKPINK & Selena Gomez hits 200M views on YT." - @iamlalisafan
07/09/2020, 19:39:19
http://twitter.com/download/iphone

10 Additional Questions and Clarifications

If you need help on the assignment, or are stuck on any part of it, don't hesitate to discuss with fellow students on Slack in #project1. We will monitor these channels and provide additional help if needed.

If you would like clarification on any part of this document, please contact us by posting in #project1. Depending on the nature of your clarification we may update this document.