# CPSC 449 - Web Back-End Engineering

Project 1, Fall 2021

due Friday, September 24 at 9:45 pm PDT

*Last updated Friday September 10, 3:20 pm PDT*

In this project you will familiarize yourself with Python and the HTTP protocol by implementing an HTTP client and server.

The project may be completed individually, or in a team of up to three students as long as all students are enrolled in the same section of the course.

## Platforms

You may use any platform to develop the code for this project, but per the [Syllabus](https://docs.google.com/document/d/1pnfCuLMI-5LxU5ZXGrKbNHO9LFtC_1tQ_C1Qez2p1sA/edit?usp=sharing) the test environment for projects in this course is [Tuffix 2020](https://github.com/kevinwortman/tuffix) with [Python 3.8.10](https://www.python.org/downloads/release/python-3810/). It is your responsibility to ensure that your code runs on this platform.

## Libraries and Code

This project must be implemented in pure Python with the [Python Standard Library](https://docs.python.org/3.7/library/), without recourse to third-party libraries.

Code from the Python documentation, [*A Whirlwind Tour of Python*](https://jakevdp.github.io/WhirlwindTourOfPython/), and [examples provided by the instructor](https://gist.github.com/ProfAvery) may be reused. All other code must be your own original work or the original work of other members of your team.

## Background

Startups have often been [described as the wild west](https://www.google.com/search?q=startup+%22wild+west%22&tbs=qdr:y), but as small companies grow, investors often see a need for [adult supervision](https://www.mercurynews.com/2017/08/23/commentary-when-adult-supervision-was-used-in-silicon-valley/) to bring some maturity and sound business decision-making to an often-chaotic environment.

In this particular situation, the engineers in your startup have been using sites like [FOAAS](https://foaas.com/) (*content warning*: profanity) to post [rude messages](https://www.cnn.com/2019/07/22/success/swearing-at-work/index.html) to internal company chats. Your new director of human resources is concerned that this behavior may create a [hostile work environment](https://resources.workable.com/stories-and-insights/hostile-work-environment-signs-fixes), will affect [recruiting](https://www.shrm.org/resourcesandtools/hr-topics/behavioral-competencies/global-and-cultural-effectiveness/pages/young-tech-workers-prepared-to-exit-unwelcoming-company-cultures.aspx), and could open the company to [legal liability](https://www.shrm.org/resourcesandtools/hr-topics/talent-acquisition/pages/eliminating-foul-language-at-work-as-easy-as-1-2-3.aspx).

You have been tasked with building a new version of the FOAAS service which continues to provide the same HTML pages as the original service, but with language suited to the professional environment.

## Integrating services with HTTP and JSON

Users in the company currently request HTML pages from FOAAS, but the service is capable of producing other data formats, including JSON. Your plan is to retrieve the requested URL from FOAAS, but pass its output through the [PurgoMalum](https://www.purgomalum.com/) service in order to render the text work-appropriate prior to returning it to the user as HTML.

We begin with a proof-of-concept: integrating the two services by writing a command-line utility named redact. If the command-line tool produces the correct text, then we know that the code functions correctly, and we can begin work on generating HTML through a server.

Implement the following command-line interface, taking the URL of a FOAAS endpoint as a command-line argument and printing the JSON it returns, with the message redacted:

prompt> **python3 redact.py**

Usage: redact URL

prompt> **python3 redact.py /because/ProfAvery**

{

"message": "Why? Because \*\*\*\* you, that's why.",

"subtitle": "- ProfAvery"

}

Taking the URL path from the command line with [sys.argv](https://docs.python.org/3/library/sys.html#sys.argv), use [http.client.HTTPSConnection](https://docs.python.org/3/library/http.client.html#http.client.HTTPSConnection) with the Accept: application/json request header and [json.loads()](https://docs.python.org/3/library/json.html#json.loads) to retrieve data from the FOAAS API, then send the message field to PurgoMalum’s /service/json endpoint. You will need to use [urllib.parse.quote()](https://docs.python.org/3/library/urllib.parse.html#urllib.parse.quote) to encode the message for inclusion in a URL, json.loads() to retrieve the result field and [json.dumps()](https://docs.python.org/3/library/json.html#json.dumps) with the indent parameter to print the combined JSON output.

## Dynamically generating HTML

Now that you have the back-end calls working correctly, you will need to generate the corresponding HTML for users to include in company chats. We will do this by implementing a [custom HTTP handler](https://gist.github.com/ProfAvery/51fc2c387a33f46504e220e62b9692ec).

Create a subclass of [http.server.BaseHTTPRequestHandler](https://docs.python.org/3/library/http.server.html#http.server.BaseHTTPRequestHandler) and implement the do\_GET() function. This function should retrieve the [path](https://docs.python.org/3/library/http.server.html#http.server.BaseHTTPRequestHandler.path) from the incoming request, use it to make the API calls to FOAAS and PurgoMalum that you implemented previously, substitute those values into a [multi-line f-string](https://realpython.com/python-f-strings/#multiline-f-strings), then return that string as an HTML response.

Use the HTML returned by FOAAS as a guide for your template. Preserve the Bootstrap styling and the link to foaas.com, but omit the <script> tags for Google Analytics. When your server is complete, browsing to <http://localhost:8080/because/ProfAvery> should yield the following:

# **Why? Because \*\*\*\* you, that's why.**

*- ProfAvery*

[foaas.com](https://foaas.com/)

Accessing the other message paths and filters from foaas.com should produce similar results. The only exception is the /operations endpoint. The service returns a more complicated JSON structure for this endpoint, and does not need to be exposed by your server.

## Submission

Your project should include summary information in a README file [as described in the Syllabus](https://docs.google.com/document/d/1pnfCuLMI-5LxU5ZXGrKbNHO9LFtC_1tQ_C1Qez2p1sA/edit?usp=sharing). Only one submission is required. Be certain to identify the names of all team members at the top of the README.

Submit a [tarball](https://computing.help.inf.ed.ac.uk/FAQ/whats-tarball-or-how-do-i-unpack-or-create-tgz-or-targz-file) (.tar.gz, .tgz, .tar.Z, .tar.bz2, or .tar.xz) file containing the following through Canvas before 9:45 pm PDT on the due date:

1. A README file as described in the Syllabus
2. The Python source code for your client and server

Do **not** include compiled .pyc files, the contents of \_\_pycache\_\_ directories, or other binary files. If you use git, this includes the contents of the .git/ directory. See [Git Archive: How to export a git project](https://www.atlassian.com/git/tutorials/export-git-archive) for details.

If you include other files in your tarball, I will not examine them unless your README states explicitly that they should be included in the evaluation of your project.

The Canvas submission deadline includes a grace period of an hour. Canvas will mark submissions after the first submission deadline as late, but your grade will not be penalized. If you miss the second deadline, you will not be able to submit and will not receive credit for the project.

**Note**: do not attempt to submit projects via email. Projects must be submitted via Canvas, and instructors cannot submit projects on students’ behalf.

See the following sections of the Canvas documentation for instructions on group submission:

* [How do I join a group as a student?](https://community.canvaslms.com/t5/Student-Guide/How-do-I-join-a-group-as-a-student/ta-p/468)
* [How do I submit an assignment on behalf of a group?](https://community.canvaslms.com/t5/Student-Guide/How-do-I-submit-an-assignment-on-behalf-of-a-group/ta-p/294)

### Grading

The grade for the project will be assigned on the following five-point scale:

#### Exemplary (5 points)

Results are correct; code and documentation are clearly written; organization makes it easy to review.

#### Basically Correct (4 points)

Results are (mostly) correct, but the code or documentation is not easy to follow or is not clear.

#### Right Idea (3 points)

The approach is appropriate, but the work has mistakes in code or documentation that undermine the correctness of the results.

#### Solid Start (2 points)

The work makes a good start, but does not include documentation or has fundamental conceptual problems in code that do not produce legitimate results.

#### Did Something (1 point)

The solution began an attempt, but is either insufficiently to assess correctness or is on entirely the wrong track.

#### Did Nothing (0 points)

Project was not submitted, submitted code belonging to someone other than the members of the team, or submission was of such low quality that there is nothing to assess.

Acknowledgements: this grading scale is drawn from the [general rubric](https://cs533.ekstrandom.net/f21/assignments/#general-rubric) used by Professor Michael Ekstrand at Boise State University.