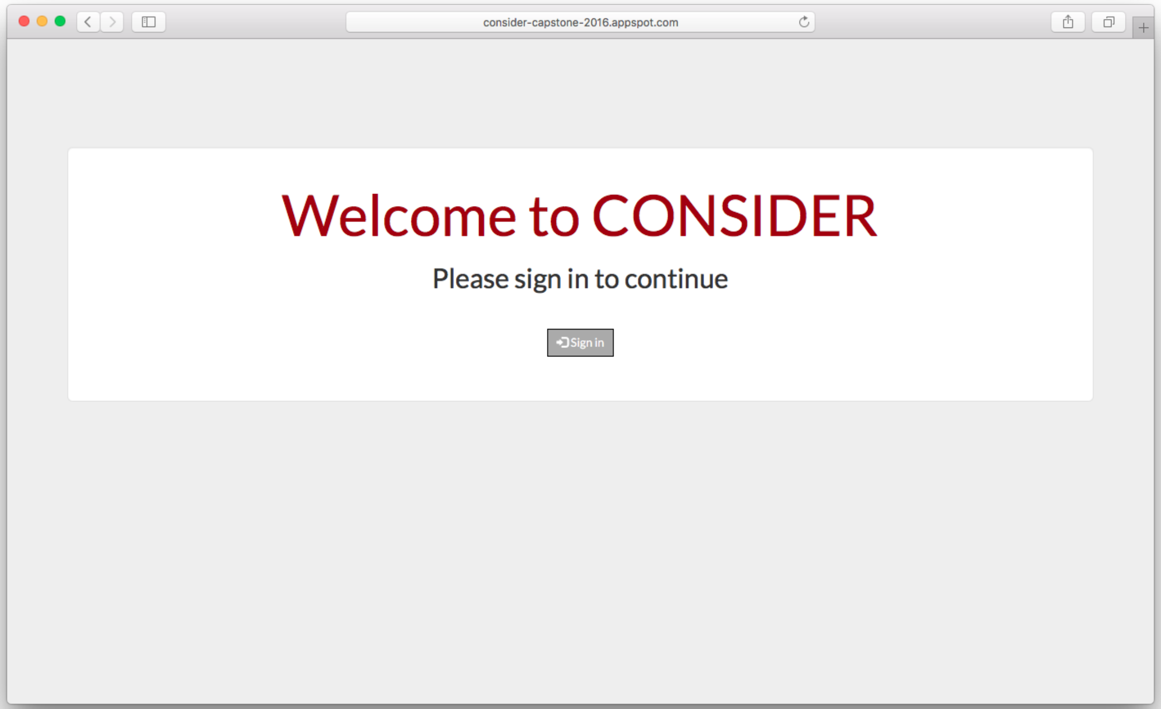
CONSIDER

Developer Guide

by **Andrew Clinton, Don Herre, Tyler Rasor, Dustin Stanley**

April 25, 2016


consider-capstone-2016.appspot.com
Welcome to CONSIDER
Please sign in to continue
Sign in

Intro

This document is intended to familiarize those who will be developing CONSIDER with the project code and to setting up the development and production environments. Due to the complexity of the frameworks used in CONSIDER, links to further resources are provided throughout this document.

Table of Contents

Codebase

This section is a high-level overview of the codebase for CONSIDER. This information is subject to change as further development occurs.

File Structure

There are three main directories for the CONSIDER project: docs, libs, and src. The **docs** directory is self explanatory containing the documentation source files. Inside the libs folder, a developer will find third party libraries which are utilized in the project. It should be noted that these libraries are currently just Python related ones. At the time of this document, all js libraries will be found in the **src** directory. The **src** directory contains the bulk of the code a developer will be working on. All templates will be found in the templates directory. Static assets such as javascript, fonts, and css will be found in the **static** directory. The **Instructor** folder contains all the controllers handling logic for the instructor views. Any other files which are found in the src directory but are not in any of the aforementioned subdirectories are either models, utility classes, or configuration classes.

Front-end

HTML

This project uses the Jinja2 (http://jinja.pocoo.org/docs/dev/) templating engine which is compatible with both Webapp2 and Google App Engine. As with most templating engines, it allows for looping mechanisms and logical if/else blocks which allows the developer to build succinct and modular html blocks.

CSS

To style the user interface CONSIDER uses the Bootstrap CSS Framework (http://getbootstrap.com/) for basic styling and several additional CSS files for further customization. All CSS code can be found in the src/static/css directory.

Javascript

Javascript can be found in one of two places in this project: inline and in a separate javascript file. In most cases, there will be js logic at the bottom of a template which handles Ajax requests or logic which will handle UI manipulation. One of the main cases where there is a separate file for the javascript is on the instructor rounds view. This view had a substantial amount of js logic going on which was moved to the instructor_rounds.js file to help keep the view clean.

Back-end

Routing

Webapp2 (https://cloud.google.com/appengine/docs/python/tools/webapp2) for Python allows for RESTful routes to be user defined in the main application file (src/consider.py currently). Traditional HTTP requests will be routed to the controller specified by the URL mapping, with GET and POST being routed to corresponding controller methods of the same name.

Models

CONSIDER uses NDB (https://cloud.google.com/appengine/docs/python/ndb/), Google App Engine's schemaless object datastore, to define the application's models. Data objects are stored as 'entities' with 'properties' instead of the traditional 'row' in a 'table'. NDB classes (currently defined in src/models.py) define the structure of each entity, including constraints (or lack thereof) on its properties. Each entity is given a unique key by which that data object can be retrieved (for example, the key for a particular section is included on student templates and is used to access that section object in the student controller). In addition to accessing an object from the datastore through its unique key, queries can be run to retrieve objects with a particular property and relationship to another datastore object (for example, this is how all response entities of a particular section and previous round are retrieved in the student controller). Once an entity has been pulled from the datastore, properties can be set directly and committed with a simple put() method.

Controllers

As mentioned before, the controller routing is defined RESTfully in the main application file. Controllers for the Instructor views have been split into individual files corresponding to the different views presented to the Instructor (e.g. the controller for adding students is in src/Instructor/students.py). Student views are managed by the Student.py controller in the main src directory. GET and POST methods across all controllers share similar patterns. First, a check is done to ensure that the currently logged in user fits the role of the controller being activated (i.e. only a Student can hit a Student route) and any discrepancy forces a redirect to the application home page. In GET requests, a dictionary of template values is built to be accessed by the view logic. In most POST requests, the page the request is made from contains NDB keys for the course and section which is then used to pull those entities from the datastore to be modified (or otherwise used to populate the response template). Additionally, in most POST requests an 'action' parameter is passed in from the view as well to denote what action to take (i.e. in src/Instructor/sections.py the 'actions' are 'add' or 'toggle'). Error checking is very thorough, and error messages are descriptively displayed to the console.

Setup

The following setup procedures work for Mac OSX and Ubuntu Linux systems.

Local Environment Setup

In order to develop and test the CONSIDER Web App locally you will need to install the Google App Engine SDK for Python on your development machine. Please note Google App Engine is under active development so these documents may not be completely accurate at the time of reading. If you are having difficulty performing the steps for setup contained in this document, you may wish to visit http://cloud.google.com and searching for Google App Engine.

Download and install the necessary software

1. Download and install Python 2.7 from http://www.python.org/download/ and install.
2. Visit https://developers.google.com/appengine/downloads and download the SDK for your platform.
3. Install the software (follow instructions for your platform).

Running the server on localhost

1. Unpack the zip file containing the project code.
2. Open up a terminal.
3. Navigate to the directory where you unpacked the project code.
4. Run the following command: `dev_appserver.py consider/`
5. Open a browser window and visit http://localhost:8000.
6. This is the Google App Engine console on your local machine.
7. Open a new tab or browser window and visit http://localhost:8080.
8. This is the web application running on your local machine.

Server Setup

If you will be pushing updates or would like to test CONSIDER in a simulated production environment you can follow the steps in this section.

Creating an account

If you are pushing updates for CONSIDER or already have a Google/Gmail account you intend to test on you can skip these steps.

1. Visit https://accounts.google.com/signup
2. Follow the instructions to make an account.

Create a Google App Engine project

If you are pushing updates for CONSIDER you may skip these steps.

1. If you have not already done so, sign into your Google account.
2. Visit https://console.cloud.google.com/projectselector/appengine.
3. Enter a **Project Name**.
4. Agree to the **Terms of Service**.
5. Click **Create.**
6. Make note of your new project's, project ID. This ID will be necessary for pushing up changes and is also the unique subdomain for your web application (i.e., **http://your-webapp-ID.appspot.com**). You can always visit https://console.cloud.google.com/home/dashboard to retrieve it.

Pushing your code to the server

1. If you have not already done so, unpack the zip file containing the project code.
2. Open a terminal window.
3. Navigate to the directory where you unpacked the project code.
4. Push the code to the server with the following command:

   ```
   appcfg.py -A <your-webapp-ID> update consider/
   ```

   NOTE: If you are signed into a Google account within your browser other than the one you are using for CONSDIER, you may need to run the following to switch accounts:

   ```
   appcfg.py -A <your-webapp-ID> update --no_cookies consider/
   ```

5. A browser window should open.
6. In the browser window sign into the Google account you are using for CONSIDER.
7. The code should now upload to the server (check your terminal for status messages).
8. Visit **your-webapp-ID.appspot.com** to see if your code has updated.

NOTE: If the server does not seem to be running your code, make sure the server is running the correct version (this is defined in your **app.yaml** in the root directory). You can visit **https://console.cloud.google.com/appengine/versions?project=your-webapp-ID**, and verify the correct version is being served.