Course-Brew User Documentation:  
Developer Guide

***Breakdown of System Design:***

This program is implemented as a Unix server that hosts a Flask framework for a Python backend, an Angular framework for the JavaScript front-end, and a SQLite database.

Each of these systems are separated on the server and interface with each other through JSON or SQL queries.

The location of the Python code is (app\_dir)\backend\src\sqlitedb

The file handlebars.py contains all of the prototypes for the external methods (i.e, those accessed externally by the Angular system) and includes the files that contain the implementations of those methods.

The current implementation files are: db.py, classes.py, courses.py, instructors.py, sections.py, warnings.py, test.py

This folder also contains schema.sql which is the SQLite template for all the databases used.

***Database***

The db.py file is used to initialize and retrieve the SQLite database as needed, its methods include: get\_db, close\_db, init\_db, init\_app, init\_db\_command

get\_db will return the currently opened database, or generate and return a new database if no database has been made by the system yet. Get\_db is the most commonly used database method, it is used every time another method in Flask needs to access the database.

Close\_db will close the currently opened database

Init\_db will use get\_db to open the current database or generate a new one, and then use the template saved in schema.sql to format the database properly.

Init\_app will be run by Flask when the server is started, and it will first run close\_db and then init\_db\_command

Init\_db\_command will call init\_db and echo to the server that the database has been initialized when it finishes.

***Classes***

The classes.py file contains the Python classes for instructors, courses, sections, and warnings.

Each of these classes has an init constructor that will populate the fields of the object; these methods can be changed to set what information the object will hold.

The warnings class also has a clear method which empties the object, and it also has a generate method which will populate the warning log with the error that has been passed to it.

***Courses***

The Courses.py file contains the methods to add, edit, and manage the courses.

The first significant method is add\_course, which will add a new course to the database and generate its sections based on the data input to the web UI.

The second significant method is edit\_course which will replace an existing course with a new one based on the data input to the web UI. This will remove the old course and all its associated sections from the database, add the new edited course to the database, and regenerate its sections.

Other methods include get\_course which fetches and returns a course’s information from the database based on the year and course id, get\_courses which returns all the courses in a given year, and get\_course\_names which returns the names of every course in a given year.

Courses.py also includes the internal method turn\_courses\_to\_objects which fetches all the courses in a year and converts them into a list of course objects which have all the parameters of a course and can be more easily used in data processing.  
***Instructors***

The Instructors.py file contains the methods to add, edit, and manage the instructors.

The first significant method is add\_instructor, which will add a new instructor to the database based on the data input to the web UI.

The second significant method is edit\_instructor which will replace an existing instructor with a new one based on the data input to the web UI. This will remove the old instructor and add the new edited instructor to the database.

Other methods include get\_instructor which fetches and returns a instructor’s information from the database based on the year and instructor id and get\_instructors which returns all the instructors in a given year.

Instructors.py also includes the internal method turn\_instructors\_to\_objects which fetches all the instructors in a year and converts them into a list of instructor objects which have all the parameters of a instructor and can be more easily used in data processing.

Another internal method filter\_instructors is included in Instructors.py, this method will generate a list of the 5 best-fit professors for a given course and year based on the instructor’s current assignments and their overall availabilities.

***Warnings***

Warnings.py contains a few methods to handle the generation and maintenance of assignment warnings.

The methods in Warnings.py includes get\_warnings and create\_warnings

Create\_warnings will use a list of objects of the instructors, courses, and sections of a given year and return a complete list of warnings.

Get\_warnings will simply return a jsonified packet of the create\_warnings list

***Sections***

Sections.py contains the methods to handle the generation and display of sections, as well as the assignment process

Create\_sections will execute every time a course is added (whether through add\_course or edit\_course) and insert the sections for the fall, spring, summer 1, and summer 2 semesters, as well as the lab courses for that course.

Get\_sections\_fal, get\_sections\_spr, get\_sections\_sum1, get\_sections\_sum2, and get\_all\_sections will return a jsonified list of sections for a course in the given semester and year (or all the sections of that year for that course for get\_all\_sections)

Assign\_course will modify a section in the database to assign an instructor to that course

Turn\_sections\_to\_objects will generate a list of section objects for a given year based on the sections in the database

***Test***

Test.py is a test method file, it contains a number of methods that are intended for debugging purposes only. If any internal methods are added for testing and are not intended for general use, they should be added to this file.

***File Locations:***

The python files are located in (app\_dir)/backed/src/sqlite

The current instance of the database is located in (app\_dir)/backed/src/instance

The HTML, CSS, and JavaScript files for the frontend are located in (app\_dir)/sched-fe/src/app

(app\_dir)/sched-fe/src/app contains sub-folders for specific pages on the frontend

***How to add new UI method to system:***

In order to add a new functionality to the UI a number of things have to be done.

First, if the functionality includes a new page, then a new .html, .css, and .ts file have to be written for that page. It is recommended to use the template files and insert your own code.

If any internal methods are needed (which they in all likelihood will be, otherwise the page would be floating and wouldn’t affect anything) then a python method has to be written, either in a new file or an existing one (see How to add new internal method to system below). Once the method is written, have the .ts file call the method prototype in handlebars.py

***How to add new internal method to system:***

If the method needs to be able to be run from the console, it is recommended to write a prototype in handlebars.py that calls the method which would itself be contained in one of the other .py files mentioned previously.

If the method is purely internal and will only be called by other methods, then simply write it within whichever file is most applicable.

If the method is for a new component of the system then you should create a new .py file, write the methods within it, and include the file in handlebars.py and whichever other files need to use it.

***Syntax Explanations:***  
***HTML***

Some useful HTML guides are:

<https://www.w3schools.com/html/>

<http://www.simplehtmlguide.com/cheatsheet.php>

***CSS***

<https://www.w3schools.com/css/>

<https://cssreference.io/>

<https://www.htmldog.com/guides/css/beginner/>

***Angular***

AngularJS is a framework for JavaScript

Some useful JavaScript guides are:

<https://www.w3schools.com/js/>

<https://javascript.info>

<https://www.geeksforgeeks.org/javascript-tutorial/>

And some useful Angular guides are:

<https://angularjs.org/>

<https://www.w3schools.com/angular/>

<https://www.codecademy.com/learn/learn-angularjs>

***Flask***

Some useful guides to Flask are:

<https://pymbook.readthedocs.io/en/latest/flask.html>

<http://flask.pocoo.org/>

***Python***

Our system can work when run as Python 2.7.X or as 3.X.

Some useful links for the Python syntax used are: <https://docs.python.org/2/reference/index.html>

And: <https://docs.python.org/3/reference/>

***SQLite***

SQLite includes a large number of queries that can be used, the most commonly used queries in our system are select, insert, and update.

Other queries and the proper syntax to use them can be found here: <https://www.sqlite.org/lang.html>

API:

Get\_instructors

POST: /get\_instructors

GET: /get\_instructors

Response:

Get\_Instructor

POST: /get\_instructor

GET: /get\_instructor

Response:

Add\_Instructor

POST: /add\_instructor

GET: /add\_instructor

Response:

Edit\_Instructor

POST: /edit\_instructor

GET: /edit\_instructor

Response:

Filter\_Instructors

POST: /filter\_instructors

GET: /filter\_instructors

Response:

Get\_Courses

POST: /get\_courses

GET: /get\_courses

Response:

Get\_Course

POST: /get\_course

GET: /get\_course

Response:

Get\_Course\_Names

POST: /get\_course\_names

GET: /get\_course\_names

Response:

Add\_Course

POST: /add\_course

GET: /add\_course

Response:

Edit\_Course

POST: /edit\_course

GET: /edit\_course

Response:

Get\_All\_Sections

POST: /get\_all\_sections

GET: /get\_all\_sections

Response:

Get\_Sections\_Spr

POST: /get\_sections\_spr

GET: /get\_sections\_spr

Response:

Get\_Sections\_Sum1

POST: /get\_sections\_sum1

GET: /get\_sections\_sum1

Response:

Get\_Sections\_Sum2

POST: /get\_sections\_sum2

GET: /get\_sections\_sum2

Response:

Get\_Section\_Fal

POST: /get\_sections\_fal

GET: /get\_sections\_fal

Response:

Assign\_Course

POST: /assign\_course

Response:

Get\_Warnings

POST: /get\_warnings

GET: /get\_warnings

Response:

Pull\_From\_CSV

GET: /pull\_from\_csv

Response:

Add\_Year\_Copy

POST: /add\_year\_copy

GET: /add\_year\_copy

Response:

Get\_Years

GET: /get\_years

Response:

Generate\_PDF

POST: /generate\_pdf

GET: /generate\_pdf

Response: