CIS 192 Project Proposal:

Group Members: Andrew Ong

Project Idea: A Dynamic Calendar

My project proposal is a combination of a to-do-list and a standard calendar. I am implementing a web application where you can create items on a to-do-list as well as schedule fixed appointments.

* Sign In / Account Creation:
  + The web application will have a login and signup page upon visiting the site. Once the user has an account and enters the correct username/password, the user will be given access to their list of appointments and the ability to create more.
  + On the Front-End this is handled by simple bootstrap forms
  + On the Back-End this is handled using a Flask API
* Calendar Page:
  + The web application features two lists. One for creating/modifying/deleting to-do-list items and one for creating/modifying/deleting static events.
  + The application queries the database for existing static and dynamic events and then sorts them by start date. It dynamically inserts dynamic events into open time-slots between “working” hours, currently 9am-9pm.
  + Jinja is used to format the html data from python objects and a bootstrap grid is used to better visualize the information.
  + Creating/modifying/deleting events are all done using jquery and ajax requests in javascript files. The appropriate triggers (clicks/doubleclicks/etc.) signal the javascript to send ajax requests to the backend server, which modifies the database to complete the request.
  + Note that the web page is not dynamically updated. I may consider adding this for some portion of the requests, but as event sorting is completed on the backend, the web page needs to be reloaded for any new event to appear. This would be feasible if the sorting was implemented on the front-end, which would have performance boosts, but as this is a python class, sorting of the events was done on the backend.

How I will stasify the requirements:

* Magic Methods: I use object oriented programming to create dynamic/static appointments. I implement magic methods in order to sort them, as they have different fields to sort on.
* Modules: I use flask, pymongo, bson, datetime, os, requests, and functools.

Flask is used to create the server and api endpoints.

Pymongo is used to connect user data to a mongodb database hosted on mongolab

Bson is used convert integers into ObjectIDs for mongodb querying.

os is used to import environmental variables (the mongodb uri )

* Decorator: I use flask decorators for endpoints and the functools @total\_ordering decorator to sort my events.