Wong Ray Down a One Lin Street:
Raymond Wu (PM), Tina Wong, Ray Onishi, Jason Lin
SoftDev2 pd07
P05 -- Fin
2019-05-12

Actavius

Description

The goal of Actavius is to combine all of the services that a student uses to keep track of their college application activities, such that it is accessible all on one website. In addition to exploring different colleges and looking up details about universities, the user will be able to keep track of where they are on a timeline with the help of a To-do list and a calendar that will be populated with deadlines.

Basic Features:

- Sign up, authentication functionality
- For each college, display detailed requirements/checklist
    - Provide ability to edit user's answer to the supplementary questions
- Display financial aid information and links to FAFSA and CSS
- Calculate EFC based on user information
- Search for colleges (unfiltered, by name only)
- Display deadlines for each college (with the earliest deadlines on the top of the page)
- Allow user to add their own deadlines

Possible Extra Features:

1. If there is an available API, allow the user to input scores or other preferences in their profile and give the user suggestions for colleges based on that information
2. Send emails if deadlines are approaching
3. Peer review of supplements
4. Filtered search (by major, etc.)
5. "Gamifying" the college application app

Overview of Components:

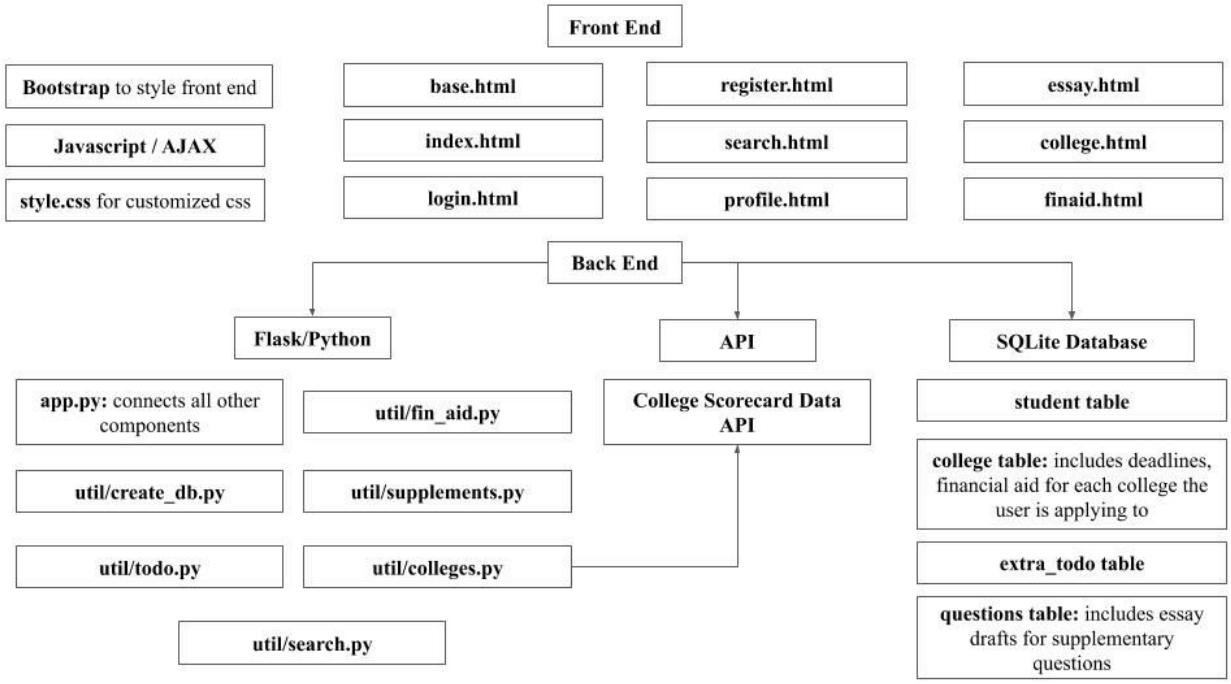
Back-end

- APIs:
  - College Scorecard Data API
    - https://collegescorecard.ed.gov/data/documentation/
    - API key: iGVF13OyKrZsilrXKgaWpf0pfJP2wbaOD919nb6H
  - TAP Fall Headcount (CSV)
    - https://catalog.data.gov/dataset/tuition-assistance-program-tap-fall-headcount-by-college-sector-group-and-level-of-study-2/resource/cde22997-05a2-4663-81c9-73931abc3da1?inner_span=True
  - College AI API
    - https://api.collegeai.com/v1/docs/data-types/reason-ids
    - API key request submitted
- SQLite database, stores data about:
  - Student account credentials
  - Student preferences/interests
  - College details
  - Supplemental questions for colleges
  - User-defined/customized to-do entries
- HTML templating (Jinja2)
- Bootstrap for Front End Framework
  - Reasoning: It is clean and easy to use, and provides simplified and easily customizable forms. Foundation is still in its Beta version, whereas Bootstrap has legacy in that it has long been supported by a massive community of developers.
- Register/log in/log out capability
- Flask with Python (app.py connects all of the routes together)
- JavaScript/AJAX calls

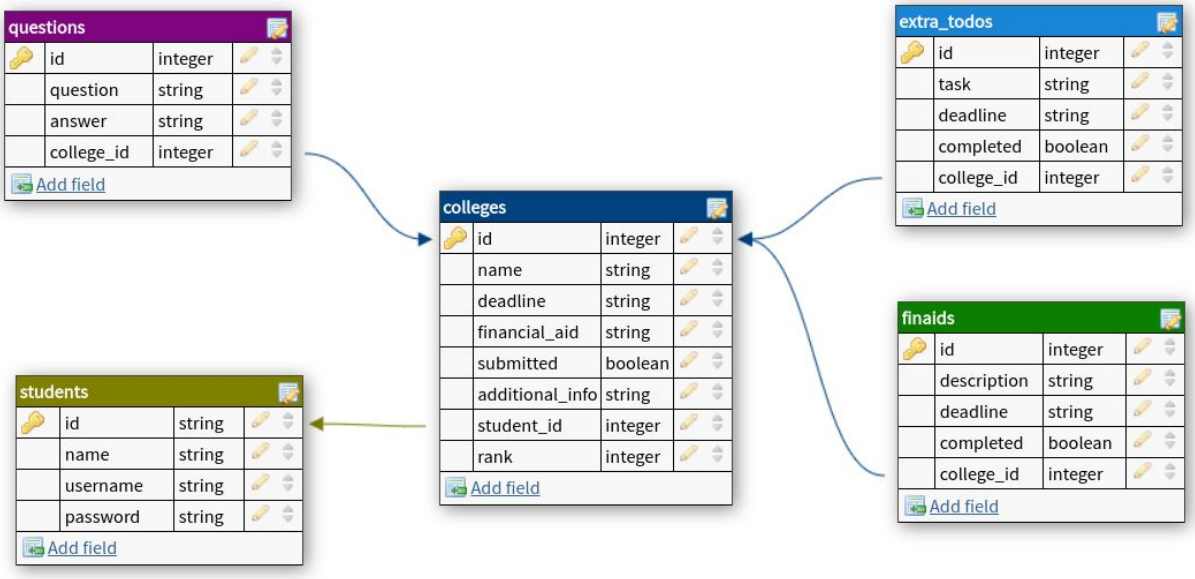
Front-end

- Landing page/user dashboard
  - If user is not logged in, show splash page with log in/register buttons
  - If logged in, show calendar and to-do list (switch views w/ button)
- Login page
- Registration page
- Search page (AJAX to show results w/o refresh)
- Profile page
  - User can customize their list of colleges to apply/to save or “favorite”
  - Financial aid settings
- Financial aid page
  - Information about only financial aid for each college
  - EFC calculator (either modal or link)
- College page
  - Add to “My Colleges”, add deadline(s) date
  - See full details about college
- Drafting page
  - Users can draft essays and save to work on later

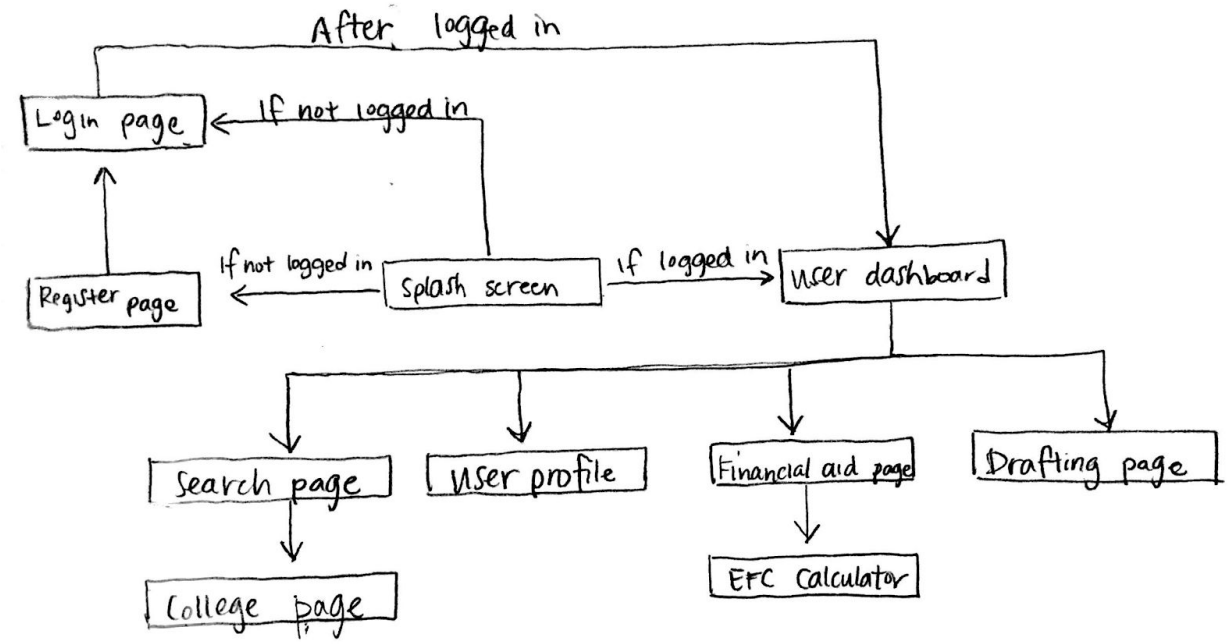
Component map:



Database schema:



Front end site map:



Timeline & Breakdown of core tasks:

Tentative project due date: June 3

Other important dates to take into account:

- AP Physics C Mechanics: May 13 (Jason, Ray)
- AP Macro exam: May 15 (Tina)
- AP Micro: May 17 (Tina)
- AP Physics C: May 22 (Jason, Ray)
- AP Calc BC: May 24 (Ray, Tina)

Task delegation: `Jason` `Ray` `Tina` `Raymond`

DUE M 2019-05-13

0. Complete API research, gathering API keys

DUE T 2019-05-14

1. Write `create_db.py` file in `/util` to create/set up database `IF NOT EXISTS`
2. Front-end: Splash page, log in page, registration page
3. Hooking up registrations to database
4. Log in capability

DUE W 2019-05-15

5. Write `supplements.py` file in `/util` to allow adding drafts to database
6. Front-end: Drafting page
7. Hooking up drafting page saves to database

DUE F 2019-05-17

8. Write `colleges.py` file in `/util` to use API(s) to get college information
9. Front-end: College page
10. Hooking up college page to display info returned from API call(s)

DUE M 2019-05-20

11. Write `search.py` file in `/util` to search database for search query, return results in a JSON
12. Using JavaScript AJAX call to show search results without refreshing page

DUE T 2019-05-21

13. Add function in `/util/colleges.py` to be able to save/remove colleges
14. Front-end: User profile page
15. Hook up adding/removing colleges on front-end with database

DUE F 2019-05-24

16. Allow user input for deadline on College page after adding to saved list
17. Add `todo.py` file in `/util` to use API(s)/user input for deadlines
18. Front-end: Logged in user dashboard page
19. Hooking up calendar/to-do view on user dashboard to database
20. Implementing view switching w/ button: view by college vs. view by date

DUE R 2019-05-30

21. Add function in `/util/colleges.py` to only return financial aid information
22. Show financial information for each saved college on page
23. Add financial aid settings section to user profile page
24. Add `fin_aid.py` file in `/util` to set user's financial aid settings
25. Add link to EFC calculator on financial aid page (modal or separate route)

`IF EXTRA TIME:`

26. Add e-mail to registration form, connect to database backend
27. Send e-mails when deadline approaches