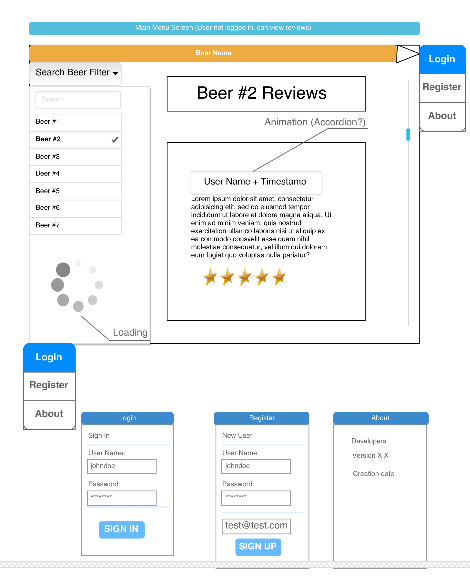
Team Collaboration Journal

The following document consists of our (**Sean Ryall, Jordan Parisee, Trent Mackeil, Andrew Jung**) project development timeline, records, and post-mortem.

Timeline of work completed:

**3/29:**

* Project proposal completed and approved (Beer review website (CRUD functionality))
* Wireframe completed in class:

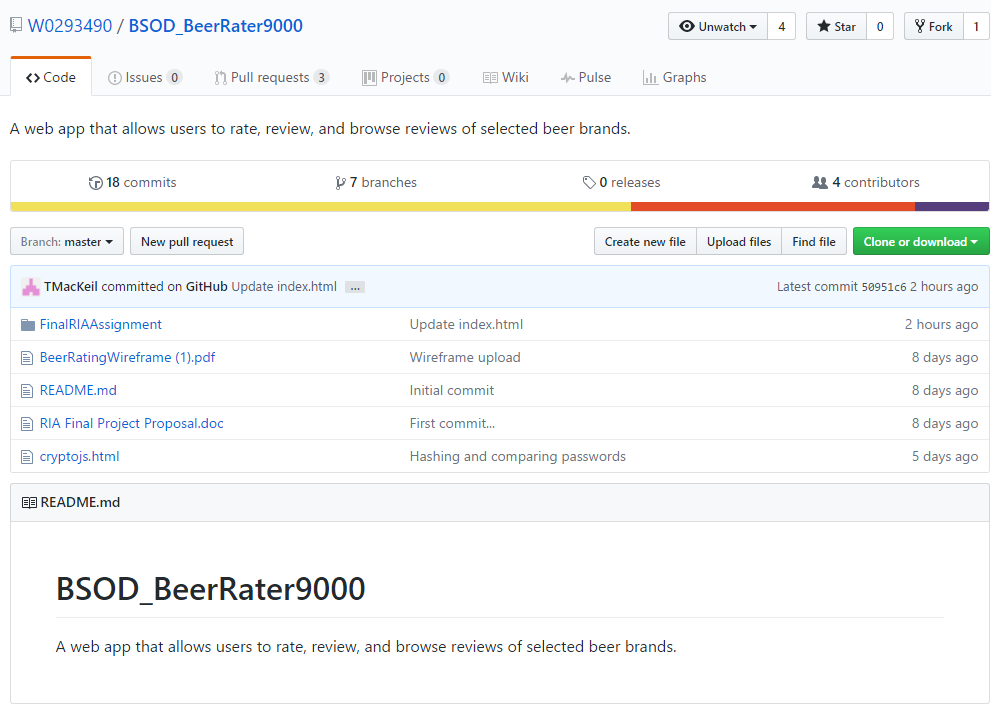


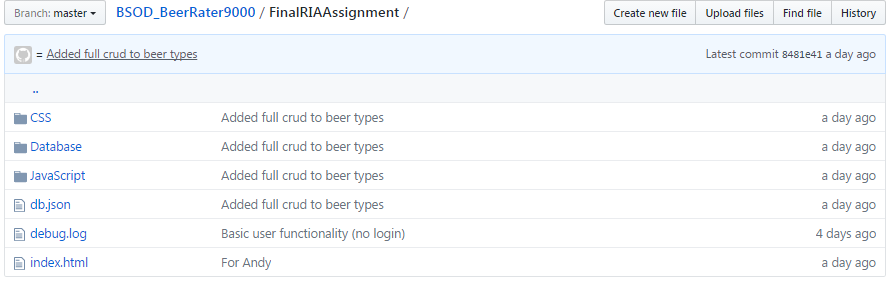
Partial wireframe.

* Created public repository for project: (https://www.github.com/W0293490/BSOD\_BeerRater9000)

**3/30:**

* Base project completed, sub-directories created (main, css, db, js)
* Base project pushed into GitHub, development branches created.
* Talked about design and how functionality wanted to appear (forms in page v.s pop-up, rating design, user capabilities, etc).

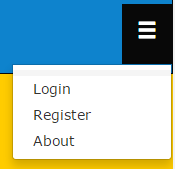




GitHub repository structure.

**3/31:**

* Created db.json format (first iteration) for users.
* Hard-coded creating users to have proper functionality
* Current passwords at this state are in plaintext, researching ways to implement hashing (and types: md5 v.s sha512, etc)
* Menus in navigation bar are complete, hamburger menu with dropdowns



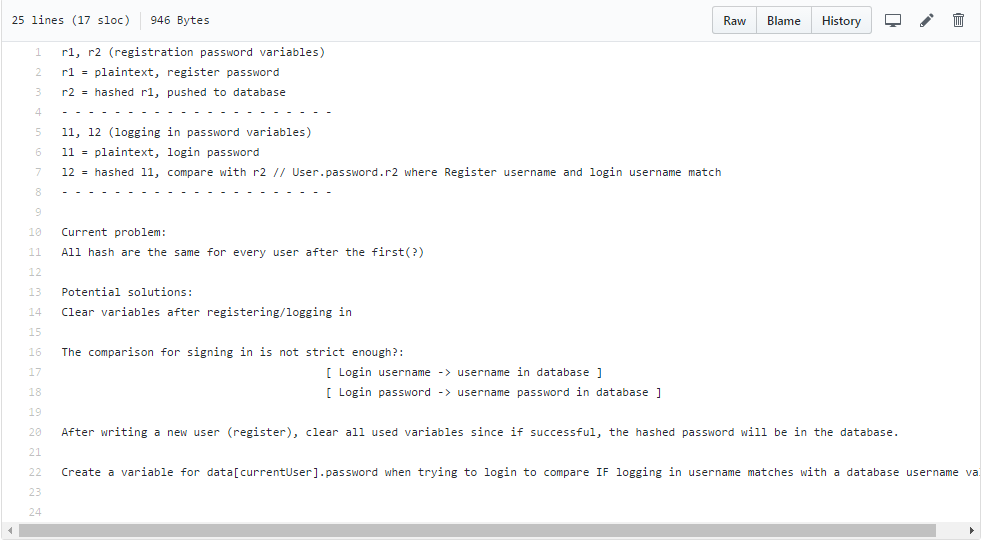
Navigation bar, opens up divs in page with/out ‘hidden’

**4/1:**

* Started the styling of the web page (continuous process)
* Hashing methods found (CryptoJS), implemented through a prototype page for testing: cryptojs.html
* After implementing the crypto method, we encountered a structural problem where each new registering user had the same hash.
* Originally used md5, but switched to SHA512 after fixing issues (simple change of function name).
* A fix for the crypto was implemented in the JavaScript file, along with a text file explaining how the method works as well as what the possible problem/solution was/is.



Prototype for hashing methods to be used in final product.



Hashing/CryptoJS help file, solution was to clear password variables after each POST.

**4/2:**

* Some styling was done to the project (again).
* Ability to create/update/delete reviews (in a jQuery dialog).
* No real review/rating functionality.

**4/3:**

* Encountered a problem where our database was not sufficient for the task at hand (structurally)
* Pursued help with instructor for ideas (change db.json structure or use new technology (mongo/express/nosql/etc)).
* Decided on restructuring our current db.json database, and called it a day.

**4/4:**

* Styled the website, added color to plain elements, effects, etc.
* Review CRUD functionality partially completed.

**4/5:**

* Implemented review ‘stars’.
* Implemented ‘add review’ functionality to front and back end.

As a team, we collaborated and discussed several milestones of the project to be completed, in terms of importance. First we created a base project and folder structure to hold our documentation and website files. The first push of the project contained a barebone design of the website with Bootstrap outlining containers and grids, as well as, elements where lists and CRUD functionality would eventually exist. This base project allowed us to each pull from the master and create our own branch to break up the development into segments. The biggest concern and roadblock for our project was structuring our database properly to fit our needs. We met this roadblock by rewriting our database to be less relational and “simplified”. From there, it was a matter of creating the CRUD functions, admin permissions, searching through lists, and styling.

For our project retrospective, the main realization was choosing a very simple backend (json-server), and in return we received a product that is not highly scalable or modular. In the future, a standard robust relational database would be necessary for the back end. For future products, hashing is great security for passwords but also be complemented with salting (https://www.en.wikipedia.org/wiki/Salt\_(cryptography)). We found an example of hashing and salting passwords using bcrypt but used a simpler implementation instead (CryptoJS). Furthermore, testing the CRUD functionality by manually inputting data and checking the output was a waste of time and very slow. In the future, we could use libraries to run unit tests (such as Mocha <https://mochajs.org/>) or use a “faker” library (<https://github.com/marak/Faker.js/>) to generate data and test our database in that fashion. There could be further implementation with reviewing beer(s) by calling multiple APIs and aggregating data but that was not in our scope for this project (but would be cool!). Time was managed well in our production, only one “main/big” roadblock to solve, which in return led to a relatively easy streamline of coding and producing.