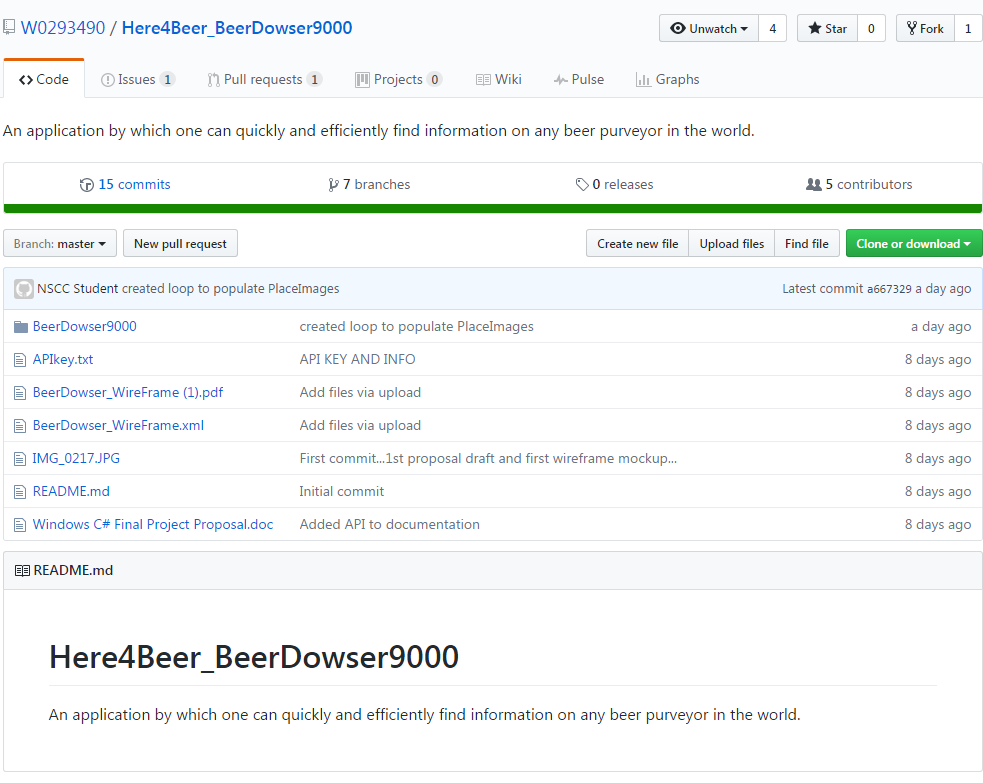
Team Collaboration Journal

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:

**3/28:**

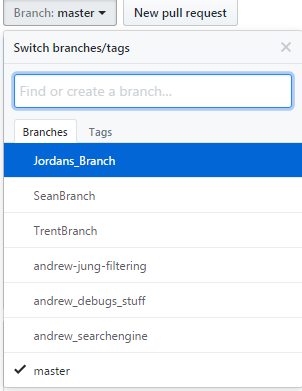
* Project proposal completed and approved (Beer Mapping API, ‘locator’ application)
* Wireframe in Microsoft Visio completed during class/night.
* Created public repository for project: (<https://github.com/W0293490/Here4Beer_BeerDowser9000>)



Overall repository for the project, holds all related documents as well.

**3/29:**

* Base project completed, main page xaml with grids for search box, filters, list view, and main information section
* Base project pushed into GitHub, development branches created for each member.



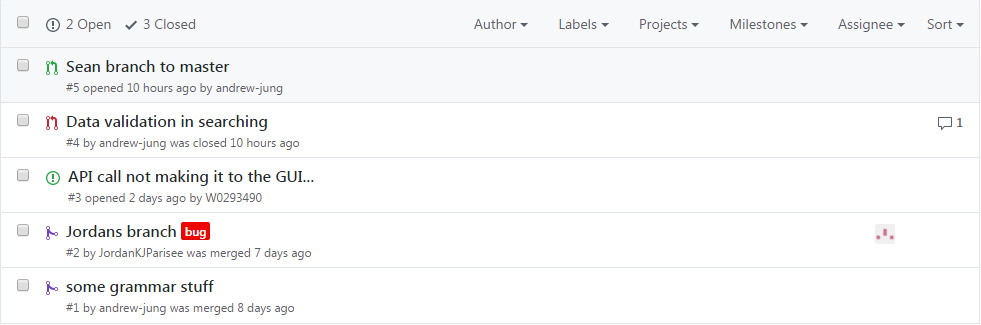
Branches for the project

**3/30:**

* Group meeting after Windows C#, during the break before Special Topics (~1 hr)
* Group meeting consisting of breaking down the project (UI, necessary methods, calling and reading from API, JSON methods, etc).

**3/31:**

* Group meeting in the morning, attempt to create JSON objects and call API to no avail, two different attempts (NewtonSoft JSON library, and NamedayDemo example)
* About page xaml created and filled.
* About page filled with custom animation and sound clip.



Pull requests and issues

**4/1-2:**

* Completed JSON methods, ability to retrieve information from BeerMapping API and filter information completed.
* MainPageData and Repository hold main code and functionality for this.
* Images unable to be displayed through search functionality (with ImageModel, but exists through hard-coding values)
* Search through user interface incomplete, but querying and filtering available through hard-coded values.
* No data validation coded at this point (blank searches, invalid returns, exceptions thrown in specific cases).



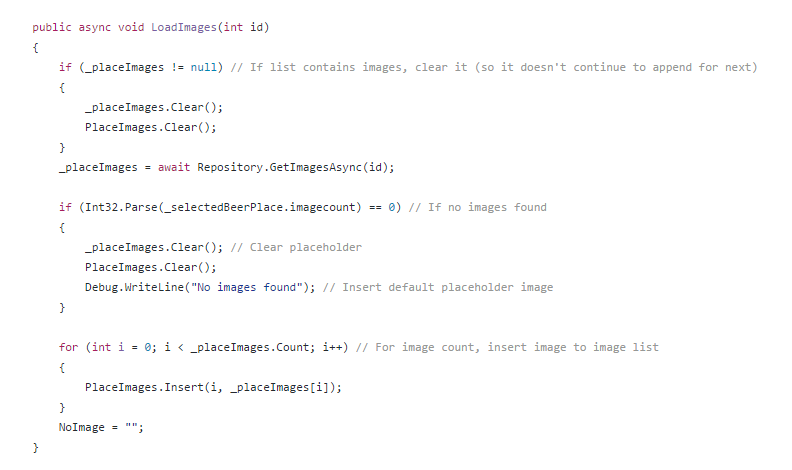
Code snippet: API query and JSON data retrieval (Repository.cs)

**4/3:**

* Search function through user interface and listing results complete.
* Filtering list through search box complete.
* Image retrieval through another call from original call complete.
* Images append to current list after every search, creating a longer list of images of previous locations browsed.
* When there is no image, a blank box is shown rather than a placeholder or message.
* Data validation still non-existent.

**4/4:**

* Images shown and removed properly from list.
* Images have a placeholder textbox when no image is found.
* Image now clickable to navigate to bigger picture, and saves search when pressing back and ability to navigate again afterwards.
* Data validation on all forms for user interface present.
* Merging of separate development branches.
* Completion of final project code.



Code snippet: data validation (MainPageData.cs)

As a team, we collaborated and discussed several parts of the project to be completed, in terms of importance. First we created a base project with the necessary interface elements and pushed the project to GitHub. 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 undertaking and most time consuming was creating the ability to properly call the API and return JSON data to be filtered through our application. We solved this with the help of Ronan’s Nameday Demo code that emulated a similar process. Afterwards, we made sure the images were being shown properly, contained placeholders, and was viewer-friendly. From there, the next milestone was the ability to query the data through the interface and return visible results and the ability to navigate and filter these results. Afterwards we focused on building proper data validation as to avoid errors and exceptions being thrown.

What we could have done better is optimizing our solutions and methods from the start. We did not validate data and only did so during the final phases, this also hindered the development cycle when testing expected inputs and outputs. We also did not include testing in our project (not in our scope), which could have helped us with the aforementioned problems. What we did well was keeping the pace of our project consistent, where we did not have to cram to finish and make constant major changes. We maintained a good stream of communication within school and outside of school to report on the status of our assigned parts. If one part lagged behind, we quickly assisted each other and ‘rotated’ through the project to have a clear perspective; we partook in pair programming to make sure each of us understood what was going into the code and to “rubber duck” our code to bounce ideas off each other and debug code.