CPSC 436V: Project Work-in-Progress (WIP)

Updated: 4 March 2020

Submission (Publish Release)

Tag a release of your project with tag version `v0.1-wip` by **Mar 25, 11:59pm**. Tagging a release allows you to continue the development, even before Milestone 2 is graded. See instructions for creating releases on Github. You release must contain your first prototype implementation and a `README.md` with your writeup.

In addition, each team member must fill out this form with a project self-assessment by Mar 25, 11:59pm: https://forms.gle/t6oTyYPumZuDjuFLA

Implementation

We created a git repository for your team on github.students.cs.ubc.ca. Develop your project by using this repository (don't forget, we'll be considering your commits when grading your final submission in Milestone 3).

Turn your project proposal into an interactive web-based visualization with D3.

- Make sure that your code is well-organized and easy to read. Use separate files for visualization components, use functions to promote code reuse. Agree upon a JavaScript style guide (e.g., Airbnb or Google) that every team member will follow. Don't work with a messy repo and then try to clean it up before the final milestone. Comment your code early on!
- We don't allow custom backends (Node.js, Python, etc) and database systems, such as Postgres or MySQL, although they could facilitate more powerful applications. In this course project, you should put your efforts into the front-end development (JavaScript, D3, HTML, CSS). Your web application should be stand-alone and have an `index.html` file as entry point, similar to the programming exercises. You can either store static (*maybe preprocessed*) datasets in your git repo or access live data through an API.
- Implement the features you outlined in your proposal. You don't need to have all details fully implemented but the general user interface and all views (visualization components) must exist. Don't worry about missing tooltips or small usability issues.
- Your interface should be as self-documenting as possible, with appropriate labels for panes, axes, and widgets, a legend documenting the meaning of visual encodings, and a meaningful title and description for the project.

- Keep your usage scenario (from your proposal) in mind when designing your interface. Make sure that your visualizations's interface and functionality either (a) allows a user to answer your proposed question or (b) successfully communicates a message or series of messages (narrative visualization).
- It can be easy to get sucked into a rabbit hole when trying to implement a stubborn feature. While it is important to build skills troubleshooting, we do not want one feature to prevent you from completing the full first prototype. If you're stuggling with a particularly tough problem, some ideas are: save it for later, or discuss with your teammates, or ask a TA for help!

Write-up

In addition to the implementation, you must document the functionality of your visualization project in a markdown document `README.md` in the root directory of your git repository.

- Your writeup should include the rationale for your design choices, focusing on the interaction aspects and connecting these choices to a data abstraction (including a characterization of the raw data types and their scale/cardinality, and of any derived data that you decided to compute) and the task abstraction. You should also concisely describe your visual encoding choices.
- Talk about how your vision has changed since your proposal
    - How have your visualization goals changed?
    - Does your visualization enable the tasks you set out to facilitate or successfully communicate the story you want to tell?
- Add at least one screenshot to your document that illustrates your current prototype. Make sure that all of the views you have implemented so far are documented in screenshots, you may need more than one.
- Add a link to the original data source.
- Briefly describe your current data preprocessing pipeline, if there is one.

Project Management & Team Assessment

- Status update:

Update your work breakdown and schedule with your current status. For each chunk of work that you have started so far, add actual numbers of how long it took to do in reality (put these next to your original estimates so you can see them side by side). Do this for both estimates, the number of hours and the date completed. Also add to your work breakdown any new chunks of work that you ended up needing to do that you had left out of your previous plan.

- Contributions Breakdown:

Briefly describe which team member worked on which tasks and their responsiblities. Did everyone contribute equally?

- Team Process:

Assess your team's performance so far according to the table below.

	Weak	**Satisfactory**	**Good**	**Excellent**	**What are specific actions you want to take to address issues, if there are any?**
Team has a clear vision of the problem(s)					
Team is properly organized to complete task and cooperates well					
Team managed time wisely					
Team acquired needed knowledge base					
Efforts communicated well within group					