CPSC 447 23Sep: Final Project Submission

Submit a single project report (M4) for the whole team as a *PDF* on Gradescope by Wed Dec 6, 6pm.

Tag a release of your project with tag version `v1.0-final` by the same deadline. Your release must contain the final implementation and a thumbnail screenshot of your vis (`thumbnail.png`).

Submit individual peer evaluation forms for each team member on Qualtrics.

Reminder: Face to Face Grading

Your team will sign up for a face-to-face grading slot with the TA assigned to your group (roughly 5-10 minutes per group), through a scheduled Zoom session. The TA will have already read your report and watched the video version of the demo. You should be prepared to answer any questions they have for clarifying the behavior of your system through a short demo, be ready to share your screen with the submitted version running live. The TAs may ask you to talk through any details of the software architecture or implementation. All members of your team should attend the face to face demo if at all possible. Exact time slots for signups will be announced before the due date.

Implementation Requirements

Finish developing your visualization project by using your git team repository (don't forget, we'll be considering your commits when grading your project).

Turn your project proposal into an interactive web-based visualization with D3 that is ready to be deployed and that can be consumed by your intended audience.

- **Refine the features you started earlier and implement the rest of the features you outlined in your proposal, paying close attention to usability and information density.**
- You must have at least three distinct views.
- At least one view should be innovative. Your innovative view should be cleared by your TA through M2, office hours, or M3. As a reminder, it should be something beyond these standard chart types: bar chart, scatterplot, line chart, simple area chart, pie chart, donut chart, choropleth, dot density map, circle-pack (bubble) chart, tiled heatmap. You could design and implement a unique view that is appropriate to support one or more of your intended tasks. You could also augment a simple standard chart type as listed above with significant additional visual encoding or interaction functionality.

- At least two views should be bidirectionally linked, and those should be simultaneously visible side by side onscreeen.
- At least one view must support mouse-based interaction (eg. hover, click, drag, tooltip) in addition to the bidirectional linking interaction.
- You must have at least one UI widget (eg. dropdown, radio button, range slider, calendar).
- Most projects will be a good fit for dashboard style (where all or nearly all views are visible at once), but you are also allowed to use a scrollytelling style.
- Your layout should use screen space wisely. Layouts that are too sparse force the user to scroll unnecessarily and fall short with supporting bidirectional linking. Layouts that are too dense are cluttered.
- 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 app.
- Apply what you have learned in the *color* foundation lectures and choose appropriate color schemes.
- 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 (*optionally preprocessed*) datasets in your git repo or access live data through an API.
- Remember, 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. Your web application should be stand-alone and have an `index.html` file as entry point. You can either store static (*optionally preprocessed*) datasets in your git repo or access live data through an API.
- Your code must be readable, reasonably structured, and conform to a consistent style. You should follow the structure and code style of the assignments and tutorials.
- Your git commits must be reasonably sized and include informative commit messages.
- Review the project description document ensure you are meeting all requirements.

Thumbnail

We would like to publicize the top projects! Please create a single representative screenshot of your project (size: **1280 x 720 px**) named **`thumbnail.png`** and add it to the root directory of your repository.

Project Final Report Requirements

In addition to the code implementation pushed to the provided git repository, you must submit a final report as PDF.

Your final report should be a standalone document that fully describes your project.

When submitting your project proposal as a single PDF, please include **exactly** the following sections in this order:

1. Basic Info
2. Overview
3. Data and Data Preprocessing
4. Tasks
5. Visualizations
6. Usage Scenarios
7. Reflections
8. Work Breakdown and Schedule
9. Credits

Your report should be at least 5 pages of text (~2500 words), without screenshots; there is no hard limit on length because we encourage you to include as many screenshots as you need to illustrate your project clearly and to write as much as you need to explain your design choices.

You should build on your project proposal report, but make sure to carefully update everything. In particular, remove/reword any future tense when discussing any part of what you built as the project is finished with the completion of this milestone.

1. Basic Info
    - Project title
    - A list of all team members and student numbers
2. Overview
    - *Teaser image* (screenshot) of your visualization.
    - Refine your overview to be a concise 250 words.
    - A few sentences that describe what problem your visualization is tackling and how. You don't necessarily need to create a project that is purely focused on exploratory data analysis. You may choose to tell an interactive data-driven story intended to be consumed by the general public. The theme of your visualization can draw from any topic, including current affairs, history, natural disasters, and research findings from the sciences and humanities. State the intended audience. Be

brief and clear.

3. Data and Data Preprocessing

    - You are free to choose any publicly-available dataset(s).

    - In your report, briefly describe the dataset and the attributes that you will visualize. Provide a link to your data sources. Include the cardinality of the entire dataset (total number of items), the types of the attributes, and the cardinality for categorical attributes or the range for quantitative/ordered attributes.

    - When these numbers are large, you can be approximate with terms like "dozens", "hundreds", "thousands": for example, 200K instead of 196,873; hundreds instead of 376. You should differentiate attributes that come with the dataset from new attributes that you might derive for your visualizations. You may find it helpful to include a table, rather than simply relying on text.

    - If your dataset has many more than 20 attributes and you plan to visualize them all, then you may provide a high level descriptor of groups of attributes, for example say the dataset contains *demographic attributes* instead of describing every single variable.

    - Do you need to preprocess or clean up the original data? Do need to join multiple datasets? How do you plan to implement the processing pipeline? (e.g. in a separate tool, such as OpenRefine or Trifacta Data Wrangler; R scripts that are executed once; on the fly in JavaScript).

4. Tasks

    - You must have at least three distinct tasks (in both domain and abstract language) and together they must pertain to at least five distinct attributes. The tasks must have sufficient overall complexity that visualizing multiple attributes using multiple views is necessary.

    - Present your tasks in two concise lists. First create a list of (at least three) domain-language tasks where each task is closely tied to the description of data above. Then abstract those tasks into (at least three) abstract-language tasks, as discussed in class.

5. Visualizations

    - Describe the visualization interface that you have built. What views are there and what do they allow users to do?

        - For each view, describe your visual encoding choices and include the rationale for your design choices.

        - You should analyze the visual encoding in detail, in terms of marks and channels, **only** for your

most complex view. For the other views, if they have already been analyzed in any of the async lecture slides, you can describe the visual encoding very concisely - just explain how your data abstraction is mapped to the elements of the view.

        - How can users interact with your project within each view, and how are views linked?

    - **Provide rationale for your design choices**, for both visual encoding and interaction. Why are they appropriate, in terms of the principles covered in this course, for your chosen data and tasks?

        - Your rationale should link together your design, tasks, and data and discuss why they are appropriate together.

        - You may find it helpful to include comparisons to other possible design choices and describe how your design choice was more effective than the others.

        - See the F3 Debrief on Piazza for examples of rationale.

    - Screenshots illustrating all aspects of your design are mandatory.

6. Usage Scenario

    - Include a usage scenario walking through how your current visualization can be used during an interactive session, illustrated with screenshots of your system in action. It may be different than what you originally envisioned in your proposal, that's fine.

        - Usage scenarios are typically written in a narrative style and include the specific context of usage, tasks associated with that usage context, and a hypothetical walkthrough of how the user would accomplish those tasks with your visualization. The purpose of the usage scenario is to get you to think about how someone else might use the web application you're going to design, and to think about those needs before you start programming. It should also help you assess the plausibility of your identified tasks and the suitability of your chosen datasets to address those tasks. If you are using a Kaggle dataset, you may use their "Overview (inspiration)" to create your usage scenario, or you may come up with your own inspiration.

7. Reflection

    - Describe how your project has developed from your initial proposal, through your first submission, to your final product.

    - How have your visualization goals changed?

    - How have your technical goals changed?

- How realistic was your original proposal in terms of what is technically possible in D3?
- Was there anything you wanted to implement that you ultimately couldn't figure out how to do? If so, then what workarounds did you employ, or did you abandon your original idea?
- If you were to make the project again from scratch (or any other interactive visualization), what would you do differently?

8. Project Management & Team Assessment
    - Include a detailed table of all work completed for the project, split into components like in previous milestones.
    - Have each piece of the work be a different row in your table. Indicate which team member(s) worked on it, the date you estimated it to be complete, how many hours you estimated it to take, the date it was actually completed, and how many hours it took. You must include all of these columns in your table. You may add new pieces of work to this table if they came up later in development, it does not have to be identical to the estimated breakdown.
    - You must include a summary somewhere that indicates the total number of hours used overall as well as the total number of hours per team member.
9. Credits
    - Indicate any sources of inspiration, including any specific D3 code blocks that you consulted or built upon. Explain what changes you made and their magnitude (e.g. unchanged vs. minor tweaks vs. major functionality additions) for any code that you built upon.

Video

You must make a video that mirrors what you would show in a live demo. Instructions and tips for making the video are available in video-demo.pdf. Include a link to your video in the report.

Peer Review

Peer Review: You will be asked to assess the ratio of work done by each member of the team (including yourself). The common case is that everybody is doing roughly equal shares of the work. Sometimes some people are doing a little more or a little less. We particularly need to know if some people are doing much more work or much less work. Please provide additional comments in the free-form text field if you indicate any situation other than the work being equally split. Even if you have mutually agreed to a non-equal work breakdown as of an intermediate milestone (for example, a teammate is ill and plans to do more later), you

should still report the actual work breakdown and explain the situation in the comments. We also welcome any comments regarding your experience with the course, overall.

The peer review will take place through a Qualtrics form that you fill out personally. At each milestone, report on the cumulative ratio across the entire term, not just the work since the previous milestone.

The link to the M4 peer review form is: https://ubc.ca1.qualtrics.com/jfe/form/SV_cCsjVIhSsRkQyfc