Continuity of the Timeline Project

As CMPT371 comes to a close, Team 2 passes the torch to the next group of software developers to continue developing the project.

Continuous Integration

Some immediate actions should be taken so the build can continue to operate autonomously. The following github secrets should be added/changed for the new owners:

> **`DEV_ENDPOINT:`** The HTTP(S) endpoint to reach using a POST request in order to notify the host that an updated docker image has been uploaded. The software we previously used to handle Docker Swarm deploys is called Portianer. This environment tracks the develop branch.
>
> **`TEST_ENDPOINT:`** Same as the `DEV_ENDPOINT`, however for the test environment. This environment tracks the master branch.
>
> **`DOCKER_PASSWORD:`** The docker authentication key used to access the docker registry. You should update `.github/workflows/docker.yml` in order to point the `docker push` command to your personal (or organizations) registry.
>
> **`SENTRY_AUTH_TOKEN:`** Your sentry account authentication token. Used for uploading source maps.
>
> **`SENTRY_DSN:`** The Sentry DSN for your project. This is where errors will be reported.
>
> **`SONAR_TOKEN:`** Your Sonarcloud Token.

The following configurations should be adjusted:

> **`.github/workflows/docker.yml:`** As mentioned above, the docker hub location from the `docker push` command should be changed to your personal or organizations registry.
>
> **`Sonar-project.properties:`** Adjust sonar.organization and sonar.projectKey to match your organization.

Please note that Electron builds are done automatically, by creating a release on Github using their own release UI at:
https://github.com/UniversityOfSaskatchewanCMPT371/term-project-fall2019-team-2/releases
The Electron files will become available as artifacts on the particular release you created.

Branches are purposefully protected against commits from unauthorized users. These can be adjusted in Settings → Branches → Edit.

Refer to `documents/builds/README.md` for additional details about how Docker Swarm was used, and the stacks that were created to do rolling deploys.

Development

Ideally, one should have a thorough understanding of our design document before beginning development.

The following software must be installed to build and develop the software:

- NodeJS v10+
- Python 3.5+ (to develop selenium smoke tests)
- Dependencies installed via `npm install`

In order to build the project and run the development web server, run: `npm start.`

To run electron as a developer use `npm run electron`.

Building for docker is equally as easy, simply run `docker build -t my-tag .` and then run it with `docker run my-tag`

There is a wealth of details in the repository's README.md which includes how to create a production build, how to run tests, and more.

Testing

For coverage reports, refer to Sonarcloud or generate them locally. It's assumed you have read the relevant documentation and as such are aware of which manual tests have been created.

Selenium is used to run smoke tests, the build pipeline runs a script (which can be found in `scripts/setup-builder.sh`) which sets up the environment inside a docker container and then runs our smoke tests. The smoke tests themselves are well documented in `scripts/smoke-test.py`.

List of Known Bugs

- First interval isn't drawn in interval data
- Newline characters are read during parsing as invalid rather than just being thrown away
- Labels don't disappear when scrolling occasionally

- Lollipop heads are rendered halfway off the top of the screen when they're the maximal values
- Tooltips display internal data that's used to sort the columns
- Times without hours get GMT 6AM appended to their timestamp even if no valid hours were given
- The app does not scale properly when the window is sized differently
- When using the left and right arrows to select a column in the drop down menus it causes infinite scrolling
- Cannot upload .csv files made in Excel in Firefox
- Zooming also pans the data

Current Defect Estimate

After the bug party on November 27th, the team found 35 bugs. Of those, 10 were overlapping in both teams. This gives, a current defect estimate of having discovered roughly 29% of the bugs in the system.

What to Improve

Moving forward, plenty of things could be added to the project to improve the quality. For example there are two classes that have both been written for future extensibility, but that aren't used currently. These are the filter and the predicate classes; which together would allow the user to filter the data by specifying constraints.

First and foremost the greatest effort would be put into improving the robustness of the system as a whole. Additionally, the UI could be improved to utilize screen real-estate better (by making menus slide in and out contextually). The second Y column should be moved to be on the right hand side of the graph in order to improve the readability of the visualization.

If the developer is hungry for more problems to solve we are happy to provide our list of issues on git containing our current known problems:
https://github.com/UniversityOfSaskatchewanCMPT371/term-project-fall2019-team-2/issues