Mid-Term Review

Process

Current ways of working

The team works in planning cycles (*Sprints*) of 1 week, where each sprint consists of a *sprint planning*, where identified tasks are time-estimated via *scrum poker*, and then prioritised in the backlog. Each sprint planning is also followed by a *retrospective*, where the team reflects on learnings from the last sprint. Every weekday the team has an asynchronous standup, where every member updates the rest of the team what he/she did yesterday, and is planning to do today.

Tools

Project Management
For task planning and prioritisation, the team uses *Trello*, a project management application. Communication is handled via *Slack* and *Google Hangouts*, two messaging platforms. Meetings synchronisation is handled via *Google Calendar*.

Software Development
For programming, all team members uses IntelliJ, some on Mac and some on Windows. Revision handling is done with Git, using command-line tools for local handling and/or the built-in tools for git in IntelliJ, and the web interface for Pull Requests.

What has worked well :)

Communication and teamwork for coding consistently working well.
Reflecting on the Sprint retrospectives we've done so far, the main thing that the team is consistently happy about is a high functional collaboration and communication. While the first sprint didn't achieve the desired deliverable outcome, the team still felt proud of its achievements in trying and putting in the effort. The great communication and teamwork is attributed to the team often working together in the same room, which eases communication, and makes problems easy to solve together. Being in the same room also naturally allows for easy pair-programming.

Although a sluggish start, the team is now up to speed with dev environment.
Much of the time has been spent struggling with setup. We intentionally devised a more complex git structure where each group member had a forked repo, requiring a reviewed pull request for each code change. This allows more flexibility in branching, and higher code quality, but keeping track of dual repos introduces complexity, especially for newcomers to git. Like many other groups, much time was also spent getting the very basics for the API to PortCDM up and running, but after now having it done the team is more confident in being able to deliver tangible stakeholder value at speed.

What hasn't worked as well :/

Struggles with setting up dev environment slowed us down significantly.
While a part of the course goal is to learn how to work with APIs, the team still felt frustrated with the vastly increased complexity of this. The team went through several different dev environments, including IDE, plugins, build management, and API management. It was very frustrating not being able to actually deliver stakeholder value, and took much longer than it should need to with proper documentation.

Struggles with team members in different cities.
One member of the team currently resides in Stockholm, doing the course on distance. This has been a communication challenge in many ways. Sometimes tasks seems to have self-explanatory context that's been generated offline, goals are not properly communicated, nor when the team should meet. Google Hangouts solves some of the problem, but not in its entirety. To solve this, the team introduced a daily requirement to simply write two lines in Slack; What did you do yesterday, and what are you doing today? The team also get more rigorous in visualising sprints, plannings, and retros in the calendar.

Product

What has worked well :)

Development of front end functionality has been a smooth process
The front end functionality developed so far has been developed to allow for additions or changes to be made easily. While the selection of data shown to the user is currently static, the logic behind it allows for us to easily add menus or other means of selecting what data to show. The demo functionality of the user interface has been developed with the future intended functionality in mind to ease the transition from a demo functionality to a finished functionality.

Feedback from the product owner is constructive and useful
Development of the product has been made easier by productive meetings with the product owner. Clear user stories are identified and the discussions are creative.

What hasn't worked well :/

Debugging and basic code setup
The connection to the existing back end and the integration of PortCDM's code libraries has been a rough process. Productive development of functionality has been hampered by extensive bug fixing. This is due to the documentation being basically non-existent. Most of the issues seem to have been fixed by now though, and therefore this is not as large a problem any more.