**1-1. Poker Effort Estimation**

For our planning poker session, we chose to create tasks for the following functional requirement:

**The app must process event data from the muon detector to determine the number of events per minute over a specific time interval.**

We generated six tasks for this functional requirement and individually rated the difficulty of each task from one to ten. During our planning session, we generally had consensus for our ratings, with some variance between us.

For example, the task, “Create an addEvent method to store processed MuonEvents to an array”, was considered to be relatively easy by all of us. However, one person rated the difficulty slightly higher and suggested that this method may need to check if the event data is valid (eg. if the timestamps are in an order that makes sense). We discussed some other issues that may occur, such as what should happen if a maximum number of events had been reached. These additional factors helped clarify the difficulty of the task.

The task of “calculating the difference in minutes between two timestamps” was more divisive. One of our group members believed the task would be quite difficult (7) but another believed it would be easy (3), since Java likely contained some built-in functionality to help with the task. We learned that this was indeed the case and adjusted our rankings accordingly.

We all agreed that the task of “calculating events per minute over a specific time interval” would be difficult. During our discussion, we contemplated how the number of events per minute could be displayed “live” on our app screen while collecting data, since that would involve frequently saving new timestamps for the calculation. We also wondered if events per minute should only be calculated over some fixed time interval (eg. the last 10 seconds of recording) or use the whole time spent recording.

Overall, the planning poker session was useful for evaluating our list of tasks. It raised new questions about our implementation and let us generate ideas for test cases and our code structure.