Sprint Retrospective, Iteration # Second Week

User Story #	Task #	Task Assigned To	Estimated Effort per Task *(in hours)*	Actual Effort per Task *(in hours)*		Done *(yes / no)*	Notes
As a user, I shall be able to review the trainings and competitions I can pick	*#12*	*Viet Luong*	*50*	*40*		*Yes*	
As a user, I should be notified when I receive an application to an activity, and when a decision has been reached on my application.	*#29*	Viet Luong	`50	60		Yes	
	#30	Rithik Appachi Senthilkumar	50	40		Yes	
As a user, I should be able to request to join and create a competition, and be accepted or rejected by the owner	#20	Maarten van der Weide	110	120		Yes	See problem addendum.
	#30	Yongcheng Huang	110	110		Yes	
	#23 #14 #16 #13 #10						
As a user, I should be able to view	#44	Viet Luong	60	70		Yes	
		Rithik Appachi Senthilkumar					

the messages I receive.							
As a user, I should be able to have basic functionalities for interacting with the concept of boats.	#19	Stefan Creasta	100	110		Yes	
		Vlad					
As a user, I should be able to interact with the system without it rising any issues	#46	Stefan Creasta	120	130		Yes	
	#40	Vlad Iftimescu					

Project: Rowing Scenario
Group: 34B

Main Problems Encountered

Problem 1

Description: We needed to figure out a way to structure, manage, store and retrieve messages as and when required. There were many approaches for the same - we could either keep a record of every message, or just show the message once and not bother with it henceforth.

Reaction: We decided to create a Message class in order to structure a message appropriately - including the sender and receiver netIds, the message content in String format, the activity Id of the activity involved in said message, and a Position. We store the messages in the message repository, and when a user wants to view their messages, they call the /notifications endpoint and every message sent to our user is retrieved and shown. We decided on this model, since it is especially useful when a user has applied to multiple activities and wants to track their applications on all of them simultaneously, and similarly for activity owners to track all applicants for their activity.

Problem 2

Description:
While working on the activity microservice we realized that it really is the hub for inter-microservice communication, thus we needed a solid way to do this.
After figuring out how to do this, we all individually implemented parts of the communication chain, since people work on separate microservices. This led to there being miscommunications about 2 sides of the same communication, and resulting in incompatibility.

Reaction:

In response to this we held a physical meeting where we discussed the communication requirements and made a definite diagram of how exactly each communication should work, what arguments it takes and what should be returned by the other microservice.

Adjustments for the next Sprint Plan

Motivate any adjustments that will be made for the next Sprint Plan.

In the future we should hold (preferably physical) meetings where we discuss key design elements we want to implement, this helps iron out any miscommunications that could lead to non-functional code down the line.
Moreover, since a significant portion of the core logic was implemented this week, the next week’s focus was to complete the remaining tasks, refactor to introduce relevant design patterns that improve our code quality, and testing our implementations.