Editorial Notes

Group 1

General Remarks

- Some exercise reports have been slightly adapted since the initial hand in. Usually this was done based on feedback from the teachers or based on knowledge gained later on in the semester. Whenever a report was modified we added a “Editorial Notes” section to the document where we summarised what we changed.
- The Roster domain/service may called The Assignment domain/service in earlier documents.

Submission Artefacts

Below is a list important submission artefacts and notes that might help with grading.

Artefact	Notes
TAPAS-Final/doc/workflow.png	A very detailed BPMN diagram showing how our entire system works both internally and with external systems
TAPAS-Final/doc/System Overview.jpeg	A simplified overview of our idealised system. Red lines indicate communication paths that have not yet been implemented as asynchronous. The Crawler is red as it is still a part of the Auction House.

ADRs

We weren’t sure if you wanted the ADRs all in one folder or divided into the assignment folders. We felt it was better to keep them all together but here is a table that indicates under which topic each ADR falls. Please note that the numbering of the ADRs has changed since our last submission. The numbering was misleading due to ADRs for earlier topics that we added later on. We changed the numbering to make the ordering more natural and so it’s easier to link ADRs to assignments. We know that this is not the correct way of creating ADRs.

Exercise	ADRs
2. Software Architecture	• 0001 - Microservice Architecture • 0002 - Event-driven Communication • 0003 - Separate service for the Roster • 0004 - Separate service for each Executor • 0005 - Separate service for the Executor Pool • 0014 - Use choreography for task execution workflow

Exercise	ADRs
5 - 6. Interaction	• 0006 - Separate service for auction house
7. Data Persistency and Code Sharing	• 0007 - Common library for shared code • 0008 - Executor base library • 0009 - Separation of Common and Executor base library • 0010 - Single ownership for services • 0011 - Data access
9. Semantic Hypermedia (Hypermedia I)	• 0012 - Separate service for Crawler
10 & 11. The Constrained Web of Things and Hypermedia APIs	• 0013 - Use hypermedia APIs when possible

Unimplemented functionality
As mentioned during our presentation, we did not have time to implement everything as we wanted. All of our ADRs are for an idealised architecture. Below is a list of what we mention in the ADRs but have not implemented along with some of the consequences.

- 0002-Event-driven Communication
  - We only implemented one message queue for asynchronous communication internally. This was the communication of the new/removed executor event emitted by the Executor Pool and received by the Roster and Auction House. All other internal communication is synchronous
    - Consequently, we do not get the benefits of asynchronous communication such as responsiveness, scalability, and fault tolerance. This showed during our performance and chaos testing
    - Moreover, since we had always intended to implement asynchronous communication, we did not implement much failover logic for synchronous communication. For example, if the Roster is offline when a task is created in the Task List, then the Task List will not be able to send it over to the Roster and will not try again. The task will be stuck in OPEN Status forever
- 0012 - Separate service for Crawler
  - We did not create a separate service for the Crawler. It still resides within the Auction House
    - There is no real consequence to the system's functioning at this point since we are operating at such a small scale
- 0014 - Use choreography for task execution workflow
  - This is implemented, but it could be implemented more explicitly. The Task List does not really define error conditions and the task status is not really granular enough yet to fully describe the workflow state. For example, we could add Auctioned - No Executors Found status, a state from which we could allow the user to start a new auction.

General ADR Remarks
- We only have one ADR on choreography/orchestration of workflows (0014 - Use choreography for task execution workflow). Other workflows were so simple that we did not feel like we needed an ADR about them, but they usually just involve 1-2 services and are choreographed by default.