CAS 735

Implementation Project
Fall 2023

McMaster University
ENGINEERING
Computing & Software

GENERAL INFORMATION

- Author: Sébastien Mosser (mossers@mcmaster.ca)
- Start date: 05/09/2023
- Delivery date: 06/12/2023
- Team size: Three students
- Weight in final grade: 40%

LEARNING OBJECTIVES

At the end of this assignment, students should:

- **Know and understand:**
  - How to identify bounded contexts and exchanged events in an architecture
  - The role of each DDD elements in a micro-service context
  - The pros and cons of microservices architectures, at the implementation level
- **Be able to:**
  - Propose an architecture based on high-level requirements.
  - Implement and deploy basic microservices
  - Assemble such microservices into a consistent architecture

WORK TO DO

CASE STUDY DESCRIPTION

Read the case study document that is available on Avenue. That will be the target of your project. Post questions in the **#customer** channel if you need clarifications.

MINIMAL VIABLE PRODUCT: MESSAGE-DRIVEN APIS

You should target to have your first version of a running system by reading week (5 weeks maximum, 2 to 3 being more reasonable). This version will target asynchronous communication between services using message-driven APIs.

- Identify what would be your minimal and viable scenario for the project.
  - You will focus on implementing **ONE SINGLE MINIMAL AND VIABLE** scenario.
- Using event sourcing, identify the chains of events (as well as the bounded contexts) necessary to support it.
- Implement the minimal version of your services using exclusively Message-driven APIs:
  - Data storage is transient (no need to use a database for now)
  - Services are released as Docker Images

    - Deployment is available using Docker Compose.
- Deliver the first version of your architecture report on Avenue (should be 3 to 4 pages max):
    - Provide the result of your event storming
    - Justify your bounded contexts and services boundaries
    - Describe and justify your asynchronous APIs
    - Identify what are the critical part of your system that need to be tested.
- Document your system in the README.md, indicating:
    - How to build the docker images
    - How to deploy your system
- Create a release on GitHub (named MVP) to freeze your MVP code.

FINAL PRODUCT: HEXAGONAL ARCHITECTURE

- You can now focus on your complete product, using any kind of API you think is relevant for your hexagons.
- Identify a list of scenarios that your product will support. Associate to reach scenario a priority in terms of business value, and an estimation of the technical effort needed to integrate it into your product
- Develop your product as a micro-service architecture:
    - Services are released as Docker Images
    - Deployment is available using Docker Compose.
    - Tests are available (unit, integration and acceptance)
- Write an executable script that will implement you business-scenario by calling your services.
- Deliver your final version of your architectrural report (**MUST** be 10 pages max):
    - Justify your bounded contexts and services boundaries fort he final product
    - Describe and justify your APIs: which communications are asynchronous? Which one are synchronous? What?
    - Describe and justify the use of any service pattern (circuit breaker, load balancer, …) in your architecture
    - Provider a graphical representation of your system, using the formalism of your choice (e.g., hexagons, UML component diagrams)
- Enrich the documentation, describing how to run the scenario script, and how to run the tests.

DELIVERY & EVALUATION

REMARK

The business requirements might be “larger” than what your system is supposed to do. For example, you are not going to re-develop the Interac system. It is part of the assignment to identify such boundaries and what you consider “external”.

DELIVERY

Each of your delivery contains two artefacts: *(i)* an implementation (as a Github repository) and *(ii)* an architecture report. The code will be automatically cloned on the master branch of your GitHub repository. The report will be delivered through Avenue (*Service Implementation* group assignment).

Only the final delivery is graded. The MVP one is a good way to receive feedback.

EVALUATION CRITERIA

It is your responsibility to ensure that your code and report are delivered in a way conform to this document. In particular, your assignment will not be graded (i.e., will obtain the grade 0/100) if:

- It is delivered too late;
- Your report not delivered as a PDF on Avenue;
- Your code does not deploy or run.

This list is not exhaustive, and the instructor reserves the right to refuse deliveries that would not meet "reasonable" quality for a McMaster's student.

The following grading scheme will be used to evaluate your work.

Dimension	Criteria	#Points
Design (/50)	Relevance of the proposed design	25
	Architecture justification	25
Implementation (/25)	Code design & quality	15
	Test quality (design & coverage)	10
Misc. (/25)	Coverage of the business requirements	15
	Report quality	10

The design dimension will only be evaluated on the grounds of what was implemented. It is useless to design 100% of the system but implement only 10%, as your architecture will be evaluated only on these 10%.

END OF ASSIGNMENT.