# Coursework Resit 2021-2022 - Agile Cloud Automation (CO4217/CO7217/CO7517)

## Disclaimer on Plagiarism and Collusion

This is an **individual assessment**. Plagiarism, including collusion, across groups is penalized. By submitting your solution, you are stating that you are aware of the consequences, as summarized in the Declaration of Academic Integrity that you signed already, and that the solution provided for the worksheet is the result of **your sole work**.

## Part 1: NoSQL

In this part, the goal is develop a rapid prototype that processes data from a data set and that stores it in a MongoDB instance on the Cloud using MongoDB ATLAS. The application will then obtain some business intelligence from the data extracted using both Groovy collection operations and the Java MongoDB API.

This part directly draws on the contents practised in the programming exercises of the three first weeks.

### Exercise 1 (30 marks) - max 250 words

Choose a JSON dataset that is of your interest. For example you can choose one dataset from the lists below but feel free to find an interesting JSON data set:

- awesome JSON datasets
- Webster English dictionary

Other datasets, which may be present in other formats (CSV, TSV, etc):

- Interesting datasets
- Data world datasets: find data sets related to your country (e.g. Covid-19)

Define a query that will require at least one of each of the following type of operations:

1. **data selection** (typical `SELECT * FROM ...` in SQL),
2. **data projection** (typical `SELECT COL1, COL2, COL3 FROM ...` in SQL),
3. **data filtering** (typical `condition` in `SELECT * FROM ... WHERE condition`),
4. **data combination** (like sums) **and/or data grouping** (like `GROUP BY` in SQL).

Implement a cloud solution using MongoDB Java Driver from Groovy, test it, and explain the code in a self-contained way, interpreting results.

### Exercise 2 (20 marks) - max. 500 words

Critically discuss of the implications of using NoSQL technology for developing cloud systems regarding **one** of the following aspects:

1. Evolution/maintenance
2. Scalability
3. Consistency

Define and use criteria to assess advantages/disadvantages of the application of NoSQL technology in a particular cloud system.

See rubric for further details.

# Part 2: Low-code software development

In this second part, the goal is to develop a low-code development platform for a domain-specific language using the technologies discussed in the module.

This miniproject directly draws on the contents practised in the programming exercises of the second part of the module.

Below you will find a brief description of what is expected in each exercise, together with the expected duration for each section and the mark breakdown. The **accompanying rubric** details the marking criteria for each exercise adding detail on what is expected and how each exercise is going to be marked.

## Exercise 3: Problem Domain (20 marks, max. 500 words)

Find a problem domain related to cloud technologies or cloud-inpired problems where automation of tasks can bring benefits, usually by generating code from concise scripts written using a DSL. The target code can be used for documenting a system (for example, by producing graphical representations of the system), for implementing (parts of) a system or for testing (parts of) a system, or for simulating a system.

Choose an appropriate DSL for that domain from the resources provided below (feel free to do research in a problem domain of your choice).

In the essay, explain:

- The aim of the DSL: the chosen problem domain and the scope of project (the part of the problem domain to be considered). Including an enumeration of key concepts and their intrinsic and extrinsic properties.
- The objectives of the DSL: the tasks to be automated;
- Some examples illustrating examples and potential benefits of such automation with a domain-specific language.

A list of example DSLs that can be used for inspiration or as example can be found in worksheet 2 used in the module. The examples illustrate different levels of academic achievement. The rubric contains further information on how the worksheet will be marked.

## Exercise 4: DSL Design (15 marks, max. 250 words)

Design of your DSL and provide:

- A grammar of your DSL using Xtext notation, within the enclosing Xtext project.
- Examples used in exercises 3 encoded using your DSL.

## Exercise 5: Code Automation (15 marks, max. 250 words)

Develop a model compiler for automating the tasks presented in Exercise 3 using Groovy, following the code patterns shown in the lectures and programming exercises. In the absence of an implementation, explain how

you would implement it by referring to design patterns and by using examples of meaningful code snippets.

# Submission

I will assume that team members contribute evenly. If this is not accurate, please send me an email, **cc'ing all team members in that email**, stating the individual contributions of each team member.

## Zipped file

The contents of the zipped file should be:

- The Gradle project(s) including all software artifacts involved in the project, after executing `./gradlew clean`. The projects should be executable.
- Document (markdown, Word or PDF) explaining where the solutions for each question can be found. For essay-type questions, write your text in this document. For other questions, indicate how to execute your code and where examples can be found.

## Rubric and marking criteria

A more detailed description of what is expected in each exercise, accompanied by the corresponding mark and grade band can be found in the rubric.

# Additional academic resources

- Study guide on organising time.