# 4CCS1ISE Team Coursework, 2020-21

## 1 Introduction

This document describes the requirements for the group coursework for 4CCS1ISE, Introduction to Software Engineering. The coursework will be made available in the week commencing 25 January, 2021 and will be due to be submitted in two parts, the first on 24 February, 2021, 10:00 am and the second on 24 March, 2021, 10:00 am. You will undertake this coursework, ideally, in groups of 4–6 students which will be randomly assigned.

Please read this document very carefully. It explains both the process and the requirements for the coursework in detail and you need to follow the instructions contained in this document carefully. Note that this is a formal part of your overall assessment for 4CCS1ISE. As such, you need to make sure that you abide by the College's rules and regulations around formal assessment. In particular, we will undertake plagiarism cross-checks between submissions from different groups; while it is OK to discuss your work with other students, it is not acceptable to collaborate with or copy from students in other groups.

The coursework focuses on the design of a software system and will follow the overall structure of the software development life cycle (SDLC). In line with the SDLC, the coursework tasks can be, approximately, divided into 4 stages:

1. Requirements analysis and initial specification;
2. Structural design;
3. Refactoring and behavioural design;
4. Test planning.

There are three small-group tutorials (SGTs) for 4CCS1ISE. These have been designed to help you make good progress with your coursework, so please ensure you attend and actively participate. In particular,

1. The first SGT will focus on issues of requirements analysis. In order for this to be successful, your group needs to have met at least once before the SGT and undertaken the requirements-analysis part of the coursework.
2. The second SGT will focus primarily on the software design stage. To make the most of this SGT, please ensure your group has met one or more times after the first SGT and has produced a first design for the coursework.
3. The final SGT will focus on test planning and on report writing. To prepare for this SGT, please make sure your group has met at least once or twice to discuss these issues in relation to the coursework and prepare a first rough outline of the final report.

### 1.1 Group formation and management

You will be assigned randomly to your group. Groups will have, ideally, at least 4 (four) and at most 6 (six) members.

Teams should choose a project leader, take turns as team secretary, and allocate further specific tasks to all members.

The team leader will need to be nominated via KEATS and will be responsible to submit the report. Any member will be able to submit the appendix for the group, but only the nominated team leader will be able to submit the report.

The team secretary of each meeting should keep a record of when the meeting was held, who attended, what was discussed, and what actions were agreed.

Teams should use the Papyrus UML tool to create the diagrams required. Teams are encouraged to use *Github* to manage their models and documentation artefacts.

If significant problems arise in your group, it is your responsibility to let the module leader know as soon as possible and to register it on the reports. All team members will receive the same mark as the group, so distribute work evenly. If group members fail to contribute, become unresponsive or abandon the work etc., it does NOT mean that the assignment scope will change.

### 1.2 Submission details

The coursework is divided in two parts and has two deadlines.

- Part 1 - February 24th, 10:00am
- Part 2 - March 24th, 10:00am

There should be one submission per group on KEATS in each part of:

- the report (in PDF) by the team leader; and
- an appendix file with your compressed Papyrus project folder (.zip) by any member of the group.

## 2 The Academic Misconduct Management System

*This section describes the system you will have to design for this coursework.* ***The description provided is fairly high level on purpose.*** *As a first task, you will need to make sure that you have derived a set of well-defined requirements from this description.*

ATOM University, wants to improve its systems and processes to control academic misconduct in assessments by its students.

ATOM University is formed by different academic departments distributed among different faculties. At the department level, there is an academic misconduct team and a professional services team that execute the process for dealing with academic misconduct. Some departments have some automation to control such process, but others don't. There is also a central office (Student Conduct Office - SCO) which defines this process and that keeps records of all the misconduct from all departments. Atom decided that it will offer a single and standardized system to all the departments but want to make it as flexible as possible given the dynamic nature of the misconduct process.

Each department offer modules for students enrolled at specific programmes. Each module has a code, a title, a module leader, lecturers, number of credits and if it is mandatory or optional depending on the programme that the student is enrolled.

The Academic Misconduct Process defined by SCO and that should be implemented is:

### 2.1 Misconduct process

1 – Module leader or lecturer identify a suspected case of misconduct in an assessment of a module (plagiarism, collusion, 3rd party involvement in the authorship of work etc.)

2 – Module leader or lecturer report the case providing:

Module code, academic year, term

Student number, name and email

Type of misconduct

Title of assessment affected, % of marks that affects the total mark of the module

Anonymized evidence

3 – A member of the Academic misconduct team, check programme that the student is enrolled in and decides if there is enough evidence for the process to continue or if it should be forwarded to the "home" department, i.e. the one that offers the programme. If the case is forwarded, the "home" department Academic misconduct team should be notified.

3.1 – It is verified if there are previous offences at the college level - SCO

3.1.1 – If there are NO previous offences and it is 3rd party involvement in the authorship of work

a. An investigatory interview is scheduled with the module leader and a member of the misconduct team with at least 7 days' notice.

b. An invitation is sent to the student.

c. The student confirms their attendance.

d. A summary of the interview is recorded by the misconduct team.

e. An email with the summary of the interview is sent to the student.

f. The case is referred to SCO (See 3.1.3)

3.1.2 – If there are NO previous offences and it is any other case (plagiarism, collusion etc.)

a. A Local Academic Misconduct Procedure (LAMP) interview is scheduled with the module leader and a member of the misconduct team with at least 7 days' notice.

b. An invitation is sent to the student.

c. The student confirms their attendance.

d. A summary of the interview is recorded by the misconduct team with its result (no case, poor academic practice, misconduct or referral to SCO) and notes if necessary.

e. An email with the outcome of the misconduct is sent to the student.

e.1 In the case of misconduct, a form to be signed is sent to the student with the outcome (result and notes).

e.1.a. The student has 5 days to complete the form.

e.1.b. The misconduct team checks the completed form and ask for corrections from the student or notify the Professional Services team.

e.1.c. The PS team sends a copy of the signed form to the student, module leader and SCO.

e.2 In the case of referral to SCO, do the same as 3.1.3

e.3 In the case of no case, the case is dropped.

e.4 In the case of poor academic practice, that is registered but no action is taken.

f. The case is closed.

3.1.3 – If there are previous offences

a. An email is sent to module leader asking for a description of the case.

b. The referral form is completed by the Misconduct team.

c. The case is sent to SCA

d. A hearing is scheduled by SCA

e. An outcome of the hearing is received from SCA via email and registered by the PS team.

f. The case is closed.

3.2 – The case is dropped.

All the actions taken during the process should be logged with time stamps and the user who performed them to guarantee security and compliance.

Please also consider the appendix provided with this coursework specification, which shows an API provided by existing systems already in use by ATOM.

You DO NOT need to consider user administration nor access control (authentication and authorization) because ATOM already has a standard solution to these requirements.

## 3 Coursework requirements

At the end of each part of this coursework, your team will need to submit a joint report and an appendix compressed file with the Papyrus project with the diagrams.

Each report should have at least 5 (five) and at most 7 (seven) pages, excluding the cover page, and be submitted in PDF format. All the images should be readable as if the report were print (zoom 100%).

The cover page should contain:

- Name of the group
- Group members and student numbers
- Table of contents

The report should include information about the following topics (you will need to agree a meaningful structure for each report in your group):

Part 1 Report

- Team management and process:
    - Describe team members’ roles and their contributions to the project;
    - List of meetings and who attended.
- Requirements analysis;
- Use case diagram;
- Use cases descriptions;
- Class diagram;

Part 2 Report

- Team management and process:
    - Describe team members’ roles and their contributions to the project;
    - List of meetings and who attended;
- Refactored class diagram;
- State machine diagram;
- Sequence diagrams;
- Testing plan.

When submitting the reports and appendixes, label the files by the group number and part—for example, **`Team33ISE_part1.pdf and Team33ISE_part1.zip`**.

Below, there is more information of what to include in the report for each stage of development. Remember that every diagram you include needs to come with a textual explanation and a discussion of key decisions you took to come up with the diagram.

You will also have to submit the source code for your diagrams as an appendix as a Papyrus project. All the diagrams should be on the report as images and their source code submission as an appendix is mandatory.

### 3.1 Part 1: Requirements analysis and initial specification

- Carry out requirements elicitation and document the system functionalities as use cases in a use case diagram. Point out any ambiguous or incomplete requirements;
- Draw an initial class diagram of the logical system data including classes, attributes and associations;
- Write use case descriptions.

### 3.2 Part 2: Refactoring and Testing planning

- Express some behaviour within the system as a state machine. For example: the states of a shopping cart or the states of an item in an inventory.
- Refactor your class diagram to include at least one design pattern;
- Draw sequence diagrams for EACH of the use cases.
- Create a testing plan for your system.

## 4 Mark scheme

Part 1 and part 2 of the coursework each are worth 15% of the overall mark for 4CCS1ISE. This breaks down as shown below:

Part 1 – 15%

- Team management and process: 10 Marks

- Requirements analysis: 25 Marks
- Use case diagram: 20 Marks
- Use cases descriptions: 20 Marks
- Class diagram: 25 Marks
- Diagrams source code (.zip): not marked but mandatory.

Part 2 – 15%

- Team management and process: 10 Marks
- State machine diagram(s): 20 Marks
- Refactored class diagram: 30 Marks
- Sequence diagrams: 20 Marks
- Testing plan: 20 Marks
- Diagrams source code (.zip): not marked but mandatory.

## 5 Appendix

Below, we describe the functionality provided by the following system that already exist and are in use by ATOM. Your application is required to build on this system and make use of it as appropriate.

The Student Conduct Office has an API for managing Academic misconduct cases providing previous offences, referral of cases and consultation of status of cases.

### 5.1 SCO API

The key operations are:

- **`getPreviousOffences(studentNumber: int) : int`** – returns the number of previous offences of the specified student.
- **`referCase(case: MisconductCase): int`** – returns the SCO code for the case referred.
- **`getCaseStatus(caseCode:int): String`** – obtains a human-readable, HTML-formatted status of the specified case.