4CCS1ISE Team Coursework, 2018-19

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 28 January, 2019 and will be due to be submitted on 29 March, 2019. You will undertake this coursework in groups of 3–6 students.

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. Design;
3. Refactoring and Test planning;
4. Final report.

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, in week commencing 4 February, 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, in week commencing 18 February, 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, in week commencing 4 March, 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 need to organize yourselves in groups. Groups must have at least 3 (three) and at most 6 (six) members. The deadline for making groups is Friday 25 January, 2019. You need to do this on KEATS.

After the deadline, teams with less than 3 students will be merged randomly or their members will be assigned individually to other groups.

Teams should choose a project leader, a team secretary, and allocate further specific tasks to all members. The team secretary should keep a record of what meetings the team has held, who attended, what was discussed, and what actions were agreed.

Teams can use a UML tool of their choice to draw the diagrams required. Teams are recommended to use *Github* to manage their models and documentation artefacts.

1.2 Submission details

The deadline for submission is March 29th, 4pm. There should be one submission per group, to be made on KEATS by the project leader or team secretary.

2 The fashion inventory management improvement 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.

WEARTHIS, a fashion retailer are improving their systems and processes for inventory management.

They want to start using the native QR code detection of their customers' phones to have faster and cheaper services at their stores and also promote and improve their on-line sales.

At each of the stores, their employees will attach QR codes to some of the products and update the store inventory.

When customers scan a QR code, they will be able to check what is the inventory status for that item in the store they are in; and

- add the item to a virtual shopping bag;
- order other sizes of the item from other stores or from the online shop.

At the end, the customers should be able to checkout and pay for their purchase through a connection to an online payment service.

Please also consider the appendix provided with this coursework specification, which shows the APIs provided by various existing systems already in use by WEARTHIS.

3 Coursework requirements

At the end of this coursework, your team will need to submit a joint report. This final report should have at least 10 (ten) and at most 15 (fifteen) pages, and be submitted in PDF format. It should include information about the following topics (you will need to agree a meaningful structure for the report in your group):

- Team management and process:
  - Number of your group and the list of team members;
  - Describe team members' roles and their contributions to the project;
  - List of meetings and who attended;
  - List of software tools used and comments about the group experience with them.
- Requirements analysis;
- Use case diagram;
- Use cases descriptions;
- State machine diagram(s);
- Class diagram;
- Sequence diagrams;
- Testing plan.

When submitting the report, label the file by the group number—for example, **`Team33ISE.pdf`**.

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.

3.1 First stage: 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, but without operations;
- Express some behaviour within the system as a state machine. For example: the states of the shopping cart or the states of an item in the inventory.

3.2 Second stage: Design

- Write detailed use case descriptions;
- Draw a detailed class diagram now including operations;
- Draw sequence diagrams for each of the use cases.

3.3 Third stage: Refactoring and Testing planning

- Refactor your class diagram to include at least one design pattern;
- Create a testing plan for your system.

4 Mark scheme

The coursework is worth 15% of the overall mark for 4CCS1ISE. This breaks down as shown below.

- Team management and process: 10 Marks
- Requirements analysis: 15 Marks
- Use case diagram: 10 Marks
- Use cases descriptions: 10 Marks
- State machine diagram(s): 10 Marks
- Class diagram: 15 Marks
- Sequence diagrams: 15 Marks
- Testing plan: 15 Marks

5 Appendix

Below, we describe the functionality provided by the following systems that already exist and are in use by WearThis. Your application is required to build on these systems and make use of them as appropriate. In particular, there is an inventory system for managing information about availability of stock in various stores, an order system for issuing orders and triggering their fulfilment, as well as a payment service that can be used to validate and execute payments from customers.

5.1 Inventory System API

The inventory system manages information about stock availability in different stores. Stock is managed by product codes. These codes can be obtained by store employees through a separate interface, enabling them to provide them to your system when they create new QR codes.

The key operations for the inventory system are:

- **`getAvailability(productCode: ProductCode, storeCode: StoreCode, size: int) : int`** – returns the number of items of the specified product and size available in stock at the specified store.
- **`findClosestStores(addressIdentifier: String):Set<StoreCode>`** – returns a set of store codes for stores in the vicinity of the given address identifier. If the address identifier cannot be understood by the system or if there are no stores in a 50 mile radius, an empty set is returned.
- **`getStoreDescription(store: StoreCode): String`** – obtains a human-readable, HTML-formatted description of the specified store

5.2 Order System API

The order system is used to prepare and fulfil on-line orders. The key operations are:

- **`createNewOrder(customerDetails: CustomerDetails):Order`** – creates a new order that can subsequently be filled with specific order elements using the other operations. **`CustomerDetails`** is a simple class that takes information such as name, address, etc.
- **`addItem(order: Order, product: ProductCode, size: int)`** – adds the specified item to the specified order
- **`getOrderTotal(order: Order):Money`** – calculates the total price of the given order in a format that can be passed to the payment service.
- **`submitOrder(order: Order, paymentConfirmation: PaymentConfirmation) : boolean`** – triggers the delivery of the specified order given a secure payment confirmation code (which can be obtained from the payment service). Returns true if the order was successfully moved to fulfilment state. After this, you can no longer add items to the order.

5.3 Payment Service API

The payment service is an external API that handles credit card payments. It has one key operation:

- **`processPayment(creditCardDetails: CardDetails, amount: Money) : PaymentConfirmation`** – processes the specified payment and returns a payment confirmation if successful (or null if the payment was declined for any reason). **`CardDetails`** is a class with field for the usual card information.