Inf2-SEPP 2020-21

Resit coursework

Developing a Festival Event Booking System

1 Introduction

This resit coursework is to be tackled **individually**.

Similar to your coursework from the semester, it is split into two parts: Software Engineering and Professional Issues. For the Software Engineering part, the aim is to develop a small scale booking system for events in an arts festival, passing through the typical stages of software development, as well as reflecting on some open questions including around team work and the software development process. For the Professional Issues part, you will be asked to write an essay around a question which is related to the same case study. Both parts will get a mark out of 100, and then total is calculated as follows: the Software Engineering part is worth 75% of its mark, and the Professional Issues part is worth 25% of its mark.

You must achieve at least 40% overall in order to pass the course.

2 System Description

Arts festivals are very popular in the UK, and a much beloved way to spend the long summer days for audiences of any age. Edinburgh itself hosts 3 very famous festivals: Fringe, the world's largest arts festival, the Edinburgh International Festival, and the Edinburgh Military Tattoo, as well as some smaller scale festivals. Last year, the COVID-19 situation led to most of these festivals being cancelled. Things are improving this year and festivals are being scheduled again, but Government restrictions mean that many of their shows must be provided online, while for others the size of the audience must be considerably reduced to respect social distancing. Moreover, queuing at a box office is oftentimes replaced by the need to book events online.

Foreseeing the increased reliance on technology by festival organisers in the years to come, your software development company is, amongst other things, developing an online booking system for arts festival events. This system will be advertised and sold to businesses which organise festivals. You are part of the team tasked to develop this system, and you are required to work independently on part of it.

An event at the festival is the performance of a show at a certain venue and time. Using this system, customers can browse information on all the events for a certain date and then filter this information based on one or more of: the show's genre (comedy, variety, theatre, music, childrens'), how the show can be watched (online scheduled, or in person), and suitability of the show (general audience, adults only). Assume the filtering action is performed once all wanted filters are chosen. Events are presented using information about the show they belong to: the same as in filters, plus title, description, performer, duration, price; as well as the event's time(s) on the selected date. We assume all the events for a show have the same price, and there can only be one event for the show at the same time. If customers identify an event of interest, they can book it if it has tickets left. To this end, they must provide the required number of tickets (tickets are all of the same type), their personal details (name, address, email address, phone number), and their method of payment (debit or credit card). The system should complain in case the customer required more tickets than left available. After correct provision of all required information, the system forwards customers to an external payment system - which it notifies of the value needed paid - where they can pay. On this system notifying ours of a successful payment, customers are issued with a confirmation containing a booking number. This booking number is automatically generated by the system to consist of the concatenation, in order, of: the user's initial (i.e. first letter of first name), first 5 letters of their family name padded with '*'s if the name is shorter, current date, a '-' sign, current time (e.g. CAlexa2021-07-23-13:54:55.130719, LYu***2021-07-23-13:56:51.995877). If the payment system instead notifies of unsuccessful payment, customers are notified and will need to restart the booking. Bookings have a status of 'pending' for the customers ever since the start of the booking process and until they successfully pay, after which it becomes 'active'. You can assume that customers will always ultimately pay, even if initially the payment system declines their payment.

Within maximum 14 days of a booking being made, a booking can be cancelled either directly by the customer by accessing it with the email address and booking number, or by calling the box office who can do this on the customer's behalf. This changes the status of the booking from 'active' to 'cancelled'. Refunds are not considered in this version of the system, so you do not need to worry about them. Bookings cannot be edited at any point.

The system can also be used by the festival organisers to record new venues (name, address, capacity), performers (only by name, can be either represent an individual or a group) and their shows. Moreover, organisers can schedule each event for the same show, as follows:

- Always providing a date and time
- For in person shows, once the above information is provided the system can inform

them of available venues (i.e. which do not have other shows at the same time) and their maximum number of attendees given by the formula capacity / 3, to consider social distancing. The organiser can then choose a venue, which schedules the event for that show and gives back a confirmation.

- For online shows, providing the date, time and duration leads directly to the system scheduling the event for that show and giving back a confirmation. Online events have a default maximum of 100 tickets.
- For both in person and online shows, several events but of different shows can be scheduled at the same time.

Assume that the festival organisers have a way to log into the system, but do not develop this in your solution. There are no registration and login functionalities available for customers.

The system can also be used by the same festival organisers to organise several festivals.

3 Software Engineering Tasks (worth 75%; the marks for individual tasks below are out of 100)

3.1 Draw use case diagram (6 marks)

Identify use cases and actors for the festival event booking system, and represent their associations (both for primary and supporting actors) in a UML use case diagram. Your use cases should include **Book event** and **Schedule event** and closely follow the system description. For any assumptions you may make, write a separate justification.

You may either draw the use case diagram by hand and include a high-quality scan of your diagram in your report or use a software tool such as draw.io.

Keep the use cases high-level. They should be about the main interactions between actors external to the system and the system itself. They should not be concerned with all the details of user interface interactions: you are not doing design at this stage.

Please do not use any other UML use case notation than shown in class.

3.2 Describe use cases (6 marks)

For the **Book event** and **Schedule event** use cases, produce a description using the template below.

Use case name: Often a short verb phrase.

Primary actor: The actor with the goal the use case is trying to satisfy. Is usually but not always the initiator of the use case.

Supporting actors: Others the system communicates with while carrying out the use case.

Summary: A one or two sentence description of the use case. Make clear the goal that the primary actor wishes to achieve. Try not to just reiterate the steps in the main success scenario.

Precondition: What the system should ensure is true before the system allows the use case to begin. Useful for telling the programmers what conditions they don't have to check for in their code.

Trigger: The event that gets the use case started.

Guarantee: What the system will ensure at the end of the use case in the event the use case execution is a success. Sometimes it can be useful to distinguish **success guarantees**, **failure guarantees** and **minimal guarantees** (guarantees for what holds at the end of all scenarios).

Main Success Scenario: The steps the primary actor and system would go through to achieve the goal. Focus on the most common and simplest course of events. If there are alternatives, record them in the Extensions field or in a separate use case.

Extensions: Variations on the Main Success Scenario. Start each with an identifier related to the step at which it starts and a statement of the condition for when the extension applies. Then list the alternate steps and if and when you return to main scenario. Ensure it is clear whether a variation results in the use-case goal being achieved.

3.3 Stakeholders-actors (2 marks)

Using the system description:

1. Give an example of a stakeholder of the system who is not a use case actor. (1 mark)
2. Give an example of a use case actor that is not a stakeholder of the system. Hint: you should have such an actor in your solutions on use cases above. (1 mark)

Briefly justify your examples.

3.4 Describe requirements (3 marks)

1. Write 2 non-functional requirements which are relevant for the festival event booking system, using the format: "The system should . . . ". Mention the type/category/'ility' each of them falls under. Attempt to make at least one of these non-functional requirements measurable (quantifiable). (2 marks)

2. Using the same format, for each of the non-functional requirements you identified above, write one functional requirement that helps to fulfill it. (1 mark)

For the following tasks, assume that the system is single-threaded. Inputs and outputs are to be treated at an abstract level: your purpose is to simulate actions through the passing of inputs as method parameters and the retrieval of outputs from what methods return. Do not think about a user interface, not even the console, nor the reading or writing of information.

3.5 UML class model (12 marks)

Construct a UML class model for the system. Include all classes and attributes which you can extract from the system description. You must have a class named **Booking**. Only include operations when you consider them important for the execution of the scenarios of the **Book event** and **Schedule event** use cases, which you have described in section 3.2. Do not include simple operations such as attribute getters and setters.

The operation descriptions must include types of any parameters and the type of the return value. Some methods may take or return collections of objects of a given class `T`; represent these as of type `Collection<T>` rather than specifying a concrete collection type, unless you already have the exact type in your diagram as a separate class. For any enum types, use the notation from Fig. 1.

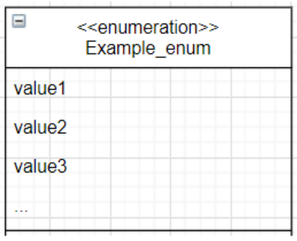

Figure 1: An enum type in class diagrams

Associations must include multiplicities at each end, but don't include these for enums. Leave navigation arrowheads off associations, as at this abstract level of design, this information is not needed.

You might find it useful to express the class model using more than one UML class diagram. For example, one diagram might show just class names and the associations and other dependencies such as generalisation dependencies between classes. Other diagrams might omit the associations, but show for each class its attributes and operations.

You are allowed to use a class acting as a controller for other classes: please name it *Festival*. Also, assume that the customer can send direct messages to event objects.

Remember to think about producing good object-oriented design.

3.6 High-level description of the UML class model (5 marks)

Expand upon the details of the design you chose. Focus on explaining where you have made design decisions. What alternatives did you consider and why did you make the choices you made? How have you attempted to deliver good object-oriented design? Make sure you refer to design principles and/or design patterns and/or using a certain architecture.

3.7 UML sequence diagram (7 marks)

Construct a UML sequence diagram for the **Book event** use case, making sure you closely follow the scenarios of that use case from section 3.2 and your class diagram. Show message names, but there is no need to include a representation of any message arguments. As needed, use the UML syntax for showing optional, alternative and iterative behaviour.

3.8 Implementation (24 marks)

1. Write your implementation of the code in Java, to cover the **Book event** and **Schedule event** use cases as you described them in section 3.2. For simplicity, in the implementation you are allowed to replace any interface classes from your class model with normal classes, but please do not change the class model. Moreover, please consider that the payment system will always return true when the payment is for an even value, and false when the payment is for an odd value. You are expected to use Java JDK 15 and IntelliJ IDEA. (15 marks)

2. Make sure you use good coding practices. Comment on their use here. (4 marks)

3. Add class-level, method-level and field-level Javadoc comments to your classes. For your methods you only need to include `@param` and `@return` tags as well as a summary of the role of the method. (3 marks)

4. While you are allowed to assume that the UI has already performed some input validation before calling the methods of your implementation, include assertions in 2 of your methods as a means of catching faults and invalid states (you may want to include more to help you develop and test your system, but this is not expected and will not receive extra marks). (2 marks)

Important! Your implementation should closely follow the structure and interaction presented in your design diagrams. These should ideally be all consistent (apart from the interface classes, see point 1 above), but if you run out of time it is also acceptable to explain specifically what you would do to make them so. Also, your implementation will only be run through its tests, thus please do not put effort into a main() method and running it from the console.

3.9 System-level tests (12 marks)

Use JUnit 5 to create system-level tests that run the scenarios of each of the **Book event** and **Schedule event** use cases. Include these tests within a `SystemTests` test class.

Hint: For the system tests to be able to retrieve events which to book, as we are not implementing the actions of browsing or filtering, you can have the operation of scheduling an event return that event. You can also think of other solutions.

Important! Your tests will provide one of the main means for assessing your implementation, therefore try to cover all of the different scenarios of the above use cases with them, and make sure they pass before submitting. Moreover, make sure the results of running the tests are easily understandable.

Briefly describe your test results and their coverage of the code being tested with screenshots.

3.10 Unit tests (5 marks)

Use JUnit 5 to write unit tests for the **Booking** class from your implementation, within a file named `TestBooking.java`, and using assertions and annotations. For simplicity, these unit tests should only concern methods which do not use objects of other classes from the system (Hint: you should have at least one such method in the Booking class as additional to any getters and setters).

Important! Your tests will provide one of the main means for assessing your implementation, therefore for each method of the class being tested try to cover a good mix of classes of inputs within your unit tests: frequent correct inputs, but also unusual/incorrect inputs and edge cases. Make sure your tests pass before submitting. Moreover, make sure the results of running the tests are easily understandable.

Briefly describe your test results and their coverage of the code being tested with screenshots.

3.11 The software development (18 marks)

For this section you should consider the assignment in terms of the type of system that is being developed, and the software process used to develop this system.

1. For the festival event booking system that is being developed, and disregarding your previous tasks, would software project engineering or software product engineering be a better choice? Please justify your answer, with 2 reasons for choosing it. (3 marks)
2. Of the two types of software development processes that we have studied in this course, which of them have we been using in this assignment? Why do you think this is the case? Refer to specific Software Engineering activities which you have undertaken, as well as to your role of developing part of the code. Provide 3 reasons. (4 marks)

3. Would the other type of software development process have been better in this context, or not, or are there arguments for both answers? Justify your reply with 2 reasons. (3 marks)

4. Focusing on the implementation and testing activities:

   - Are there any agile practices which you would have liked to use had you worked on this coursework together with a team? If so, state 2 of them and explain why. (4 marks)
   - Are there any software tools which you would have liked to use to facilitate your team's software development work? If so, state 2 of them and explain why. Here, we are looking more than for videoconferencing, file sharing or chat systems, to something that is specifically used by software development teams. (4 marks)

4 Professional Issues Tasks (worth 25%)

"What professional and ethical concerns might be particularly important to keep in mind while developing this booking system?"

Write a short essay on this topic. You can assume the booking system is being worked on beyond the limited functionality expected for this assignment. Relate your points to the topics covered by this course, but don't feel that you need to cover every topic. It is better to flesh out your arguments than to make a lot of assertions with little support.

The essay should be a maximum of 500 words (about one side of A4), and the best submissions will not be far under this.

This article gives good advice on how to structure a short essay:
`https://www.wikihow.com/Write-a-Short-Essay`

Remember that reading the first chapter of "A rulebook for arguments" should help here too.

Good essays will:

- Clearly choose a main position
- Consider the terms used and define them where necessary
- Justify arguments with reference to course materials and/or other sources
- Anticipate and address counterarguments

How well you do these things will be the core criteria for marking.

5 Good Scholarly Practice

Please remember the University requirement as regards all assessed work. Details about this can be found at:

`http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`

Please note that we may run a plagiarism checker on your solutions.

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on an online repository like GitHub then you must set access permissions appropriately (permitting access only to yourself).

6 Submission

6.1 Working code and tests

Produce a .ZIP file of your working code and tests. To do this in IntelliJ, go to
`File - Export- Project to ZIP file`.
Rename this file **code.zip**.

6.2 Report

Please submit a PDF (not a Word or OpenOffice document) of your report part of this coursework. The document should be named **report.pdf**.

How to Submit

Ensure you are logged into MyEd. Access the Learn page for the Inf2-SEPP course and go to “August Resit” – “August Assessment” – “Submit your work here”.

Submission is a two-step process: upload the file, and then submit. This will submit the assignment and receipt will appear at the top of the screen meaning the submission has been successful. The unique id number which acts as proof of the submission will also be emailed to you. **Please check your email to ensure you have received confirmation of your submission**.

If you submit several times, only the last submission will be checked by the markers.

If you do have a problem submitting your assignment try these troubleshooting steps:

- If it will not upload, try logging out of Learn / MyEd completely and closing your browser. If possible, try using a different browser.
- If you do not receive the expected confirmation of submission, try submitting again.

- If you cannot resubmit, contact the Informatics Teaching Office: ito@inf.ed.ac.uk, attaching your assignment, and if possible a screenshot of any error message which you may have.
- If you have a technical problem, contact the IS helpline (is.helpline@ed.ac.uk). Note the course name, type of computer, browser and connection you are using, and where possible take a screenshot of any error message you have.
- Always allow yourself time to contact helpline / the ITO if you have a problem submitting your assignment.

7 Deadline

The deadline for this resit coursework is on the **20th of August at 16:00 BST**. This is a very strict deadline, with no extensions granted by the school.

Cristina Alexandru and James Garforth, 2021.