5COM1053/61 CS/IT Development Exercise (CSITDE)

2021-22

Mega Independent Cinema Experience - MICE

Instructions:

Read the project scope in Section 1 below, followed by the tasks you need to complete in Section 2. Some supporting materials are provided in Section 3. The marking criteria that you will be assessed against is listed in Section 4. A final checklist to help you work through the tasks is at the end of this document (Section 5).

1. Project scope

First, read the supporting case study in the attached document ‘**MICE integrative case study**’.

**Your role:** Suppose you currently work as a Requirements Engineer for a software development company called DoITSystems (in the case study). At this stage of the development project, your manager has asked you to elicit and document the system’s requirements as per the tasks below.

2. Tasks and Outcomes

This coursework, worth 12% of the module overall, is broken down into **three parts**:

* **Part 1: User Stories (40 marks)**
* **Part 2: Volere Template Shells (36 marks)**
* **Part 3: Requirements Analysis (24 marks)**

Part 1: User Stories

Using the grid in Part 1 of the ‘GP1 templates’ document, elicit requirements via **4 user stories**. Of these user stories, **x3 should be functional requirements (FR), and x1 should be a non-functional requirement (NFR)**.

For the “As a…” part of the user story template, choose **primary** stakeholders that you identify from the case study (see Section 1) and potentially other standard good design practice you may find from research as necessary. You can choose the same stakeholder for multiple requirements, though aim to include at least two distinct stakeholders in your answer. Your choices should be key requirements (non-trivial).

For ***each*** user story, you should also document that story’s supporting **acceptance criteria** to ensure each requirement (via the user story) is testable. *Tip: if the acceptance criteria for a user story is difficult to write, check your user story is singular/broken down and not an ‘epic’ (see lecture notes).*

The user stories, and acceptance criteria, templates will give you experience of how requirements can be elicited, and documented, in Agile methodologies (often they are written on post-it notes!).

Part 2: Volere Template Shells

Choose ONE of your four requirements (via the user stories) in Part 1 and describe this requirement in more detail by completing the blank Volere requirements shell in Part 2 of ‘GP1 templates’.

You are expected to justify where you have **derived** this requirement from. Whether this is from a numbered section of the case study (please be specific) or other research, use the “*supporting materials*” section of the Volere shell to indicate this.

Note that a specification of this kind could be used with a range of stakeholders, some of whom may be unfamiliar with the Volere Template, so you should use the “*comments*” section of the Volere shell as a key for any numbered ratings/codes so that its meaning is clear. The Volere template will give you experience of how requirements can be elicited, and documented, into a more formal specification, typical for some plan-driven methodologies.

Part 3: Requirements Analysis

Analyse your functional requirements (x3 of the user stories) by completing a ‘pairwise’ comparison, deciding which is the ‘winning requirement’ using the following Likert scale devised by Saaty (2008).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Equal | Slightly | Moderate Importance | Moderate plus | Strong Importance | Strong plus | Very strong Importance | Very strong Plus | Extreme |

Look at each pair of requirements in turn (see ‘pairwise’ grid below) and decide which requirement is more likely to be ***simpler to implement*** than the other being compared. The requirement in each ‘pair’ which is easier to implement (and more likely to be implemented successfully) wins.

It may be helpful to think about CRUD (create, retrieve, update, delete), e.g. a requirement that describes a simple search (retrieve) should be easier to implement than a complicated update function. So, in this case the simple search requirement would ‘win’, and we would then decide ‘how much’ this requirement is easier by mapping to one of the numbers in the Likert scale: if the ‘win’ is marginal, lower numbers on the Likert will be used; whereas, if the requirement is far simpler than the other to implement, higher numbers will be used.

Repeat this process until all functional requirements in the specification have been compared with each other, and try to ensure each pair of requirements are compared **in isolation** from the other requirements each time.

Track your decisions by completing the grid in in Part 3 of ‘GP1 templates’. This grid should be filled in with each requirement having the same identification number (#) as the user story it relates to. The result of each comparison, e.g. winners and losers, should also have a reason given as to why (in your opinion) the requirement ‘wins’. Each ‘reason’ need be no more than few sentences to show your thinking.

**Example** **completed grid** using‘GP1 templates’ (s*uppose our user stories were describing requirements that might implement #1 = simple update | #2 = complicated update | #3 = simple search)*:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Req. #1 : #2 | Req. #1 | is | 3 | times more (\*) than | Req. #2 |
| Req. #1 : #3 | Req. #3 | is | 5 | times more (\*) than | Req. #1 |
| Req. #2 : #3 | Req. #3 | is | 6 | times more (\*) than | Req. #2 |

(\*) more likely to be simpler to implement and therefore the “winner”

**Example reason statements based on the grid above:**

* Requirement **#1** is **moderately** more likely to be successfully implemented as requirement **#2**, because whilst they are both ‘update’ features to implement, **#2** appears to have more complicated processing steps required for implementation whereas **#1** is simpler.   
  *[N.B. In your statements, you should include specific details (e.g. wording of the requirement if they relate to bookings, reports, etc.), not just generic CRUD phrasing]*

The ‘pairwise’ comparison will help you see how implementation depends on clearly written and understood requirements. You may find it useful to reuse, or adapt, this technique again when making implementation decisions, e.g. alongside requirements prioritisation techniques such as Agile’s ‘Planning Poker’.

3. Supporting Materials and References

* Canvas lecture and tutorial notes for weeks 10-14 in particular, alongside guided reading
* Saaty, T. L. (2008) ‘Decision making with the analytic hierarchy process’. *International Journal of Services Sciences*. 1 (1). pp. 83-98.
* VolereTemplate\_v14 on Canvas for more details on the template and requirement types

4. Marking Criteria

**The assignment must be completed in your assigned teams (of 4– 5) only. Individual submissions or those with incomplete teams will not be accepted without the prior consent of the module leader.**

**Notes on marking criteria:** ***Higher marks*** typically demonstrate singular, high priority requirements relevant to the case study. Templates are correct and complete, and follows conventions, such as user-friendly and non-technical language in requirements. ***Lower*** ***marks*** typically indicate requirements are not singular and testable (e.g. multiple requirements documented as one). The requirements may also be unsuitable, or out of scope, to the project described in the case study. The templates may be completed incorrectly, or sparsely, not following examples provided in class.

You will be assessed against the following criteria:

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Actual**  **Mark** | **Total Max**  **Mark Possible** |
| **Part 1: User Stories** | | |
| User stories with a clear primary stakeholder, relevant task, and justification |  | 20 |
| Acceptance criteria across the user stories demonstrate tests |  | 20 |
| **Total Part 1** |  | **40** |
| **Part 2: Volere Template Shells** | | |
| Choice of requirement maps to a user story in Part 1 |  | 6 |
| Requirement demonstrated is clear and complete |  | 10 |
| Acknowledgement of sources and assumptions for decisions made |  | 10 |
| Quality of the fit criteria |  | 10 |
| **Total Part 2** |  | **36** |
| **Part 3: Requirements Analysis** | | |
| Three pairwise comparisons as instructed in a grid |  | 6 |
| Reasons for each ‘winning’ requirement give clear implementation considerations |  | 18 |
| **Total Part 3** |  | **24** |
| **FINAL TOTAL** |  | **100** |

**4.1 Marking Scheme – Team Contribution**

As a team you should consider and submit the “**GP1 Roles and Contributions Form**” posted with the assignment. This asks you to give a percentage “contribution” that each of you has made to GP1.

**For example:**

*Example (a):* if there are five team members and you consider that everybody has contributed equally, then everyone should be allocated 20%. The total must of course be 100%.

*Example (b):* if there are four members and you all agree that one person has contributed twice as much as everybody else, then he/she gets 40% and the other three get 20% each.

If you all agree then you should complete a single team form in your submission.

If you cannot agree, then each team member who disagrees should complete the form individually instead.

**4.2. Marking – Individual**

Individual student's GP1 mark will be calculated on the assignment mark. No individual team member can score higher than the mark awarded for the assignment. Where a team member's contribution is significantly lower than that of other team members, they will be awarded only a proportion of the assignment mark.

|  |  |
| --- | --- |
| **Contribution difference from the norm** | **% of assignment mark** |
| <= 5% difference | 90% |
| > 5% AND <10% difference | 70% |
| >=10% AND <15% difference | 50% |
| >=15% difference | 0% |

**Example 1:**

**Assignment mark = 60**, 4 team members each assessed as contributing 25% (4 \* 25 = 100%)

Norm is 100/4 = **25%**. All members contributing the norm

Each team member gets 60

**Example 2:**

**Assignment mark 60**, member1; **50%**, member2: **20%**, member3: **15%**, member4: **12%**, member5: **8%**

Norm is 100/5 = **20%**

member1 gets 60 [No student can get more than the assignment mark]

member2 gets 60 [meets norm of 20%]

member3 gets 60 \* 0.90 = 54 [difference from norm <= 5%]

member4 gets 60 \* 0.70 = 42 [difference from norm between 5% and 10%]

member5 gets 60 \* 0.50 = 30 [difference from norm > 10%]

N.B. If member5 was on 5%, they would get 0%

The raw final mark out of 100 will be **scaled to be worth 12%** for the module overall. This is calculated by scaling the raw mark out of 100% to its equivalent mark out of 12%. For example, a raw total mark of 50 out of 100 means you accumulate 6% out of the possible 12% for the module, because 50 x 0.12 = 6.

5. Assignment Checklist

*This is a summary of the tasks for you to work through as a checklist. Full details are given elsewhere.*

|  |  |  |
| --- | --- | --- |
| **Task** | **Item** | **Done** |
| 1 | Read all **documentation**, ideally before starting any work. |  |
| 2 | Use the discussion on Canvas, and opportunities in class, **to ask questions**. |  |
| 3 | Read (and re-read) the **case study**, making annotations to help you. |  |
| 4 | Copy the ‘GP1 templates.docx’ and fill in each (contributing) team member’s student registration number (SRN). |  |
| 5 | Document **four user stories** (3 FRs, 1 NFR) for the proposed system. |  |
| 6 | Choose **one** of the four user story requirements and document in a **Volere** shell. |  |
| 7 | Document **‘pairwise’ comparisons** of the **functional** requirements (user stories) with reasons for the results of this analysis. |  |
| 8 | Complete the **Assignment Briefing Sheet (ABS)** as a team. |  |
| 9 | Complete the **Roles and Contribution Form (R&C)** as a team. |  |
| 10 | By the date/time shown on canvas for this assessment, **submit x3 files**: 1) the ABS 2) the R&C form 2) your work (tasks) as a completed copy of ‘GP1 templates.docx’. |  |