Updated Method and Plans: link

Additions are marked in yellow, reductions are also ~~strike-through~~.

In our context, we identified that there is an important distinction to be made between:   
**Direct Stakeholders** → as the customer who requested the game;  
**Indirect Stakeholders** → as the students in our cohort, prospects students in open-days and their parents, who will eventually play and assess our game.

This distinction is a very important one to make, as some customer requests (from the direct stakeholder) might conflict with the users’ needs (indirect stakeholders). As a matter of fact, some of the initial features thought by the team were, as we would later discover, clashing with the client’s requirement e.g. one of the early ideas for the project was (the development of) a slow-paced, low rewarding but coding-efficient turn-based strategy game, which, however, would have considerably diverged from the client’s intention of showcasing it to students and parents during open days: hence a fast paced, high-rewarding style of gameplay was agreed to be more suitable.

The team extracted the **SSON** for the project from the product brief provided [1], which was subsequently agreed with the customer to be: “Build a single-player game that involves moving fire engines between the Fire Station and the ET fortresses, avoiding ET patrols on the way, and attacking ET fortresses when the fire engines’ water cannons are within shooting range’’.   
In the **early stages of the process** the team had a brainstorming session, which served us to give initial high-level, yet clear and detailed nonetheless, direction and shape to the project and to define what features the final product would and would not have, which were then proposed and checked with the customer. The team then met the customer, who did not put forward any particularly restricting additional user requirements. Among the additional user requirements proposed, we report cross platform development, controller extension, audience-targeted development and budget and hardware constraints; the client also provided a clear prioritisation map of the aforementioned features. Following the meeting, the team discussed and modified adequately the initial ideas identifying and removing clashes when found- and created several high-level prototypes of the product. The way **User requirements** were elicited from the product brief was straightforward: the team identified and extracted the key information from the document in a systematic way i.e. find and clarify facts about the game, verify them against the customer’s requests, turn them into requirements and finally record them in the User Requirement table.

We then proceeded to deduct **Functional and Non-functional requirements** from the user requirements by following the steps in IEEE’s [2] 6.3.1.1 section: the literature provides a thorough explanation to “how the inputs to the software product should be transformed into outputs”. However, the team agreed that, considering the nature of this SEPR project, many of the steps and details are not applicable -or, in some cases, it is not advisable to do so- to our context: an instance of this is the 6.3.1.6.1 Data Base section or the five categories (Name, Mnemonic, Specification number, Version number and Source) proposed in 6.3.1.5.3 Software Interfaces. Furthermore, the submission constraints of the assessment (max 3 pages) led us to the decision of omitting some potentially polluting details additions such as the capacity section in 6.3.1.2 Performance Requirement. Throughout the whole process, the team paid great attention to “not describe any design, verification, or project management details, except for required design constraints.” Moreover, great emphasis was put into adhering with the fundamental ethical principles that apply to a computing professional's conduct: after completing the requirements tables, the team cross-checked the accuracy and faultlessness by using the ACM Code of Ethics [3]. We unanimously agreed that the result obtained was in line with the standards. The team also understood the importance of observing the basic rules of conduct presented in the BSC Code of Conduct [4] in order to guarantee a suitable environment for team-work and avoid clashes between team member throughout the course of the whole assessment. This not only helped to, but it also allowed the team to have a broader understanding of what working in a multi-faceted and diverse team is about.

For the **Use Cases creation process**, the team followed the advice presented in both textbooks “Writing Effective Use Cases”[5] and “UML Distilled”[6], but decided to not precisely and systematically stick with their guidelines, as there were many low-level ramifications and details (such as, for instance, the subdivision of use cases into Sea-level, Fish-level and Kite-level) that did not provide any more relevant information for our project. The format used, as suggested in the lecture, is text-based, as this turned out to be a straightforward, quick, reliable and unambiguous way of listing all the necessary information and did not require the use of any additional software. All use cases used by the team for the requirements elicitation can be accessed from the team’s website[7] in the “Use Cases” section.

References:

[1]University of York, Computer Science Department, “Product Brief: Kroy”, <https://vle.york.ac.uk/bbcswebdav/pid-3396020-dt-content-rid8681478_2/courses/Y2019-006404/product-brief%281%29.pdf>   
[2]IEEE Guide for Software Requirements Specifications. New York, USA: IEEE, 1984. <https://ieeexplore-ieeeorg.libproxy.york.ac.uk/stamp/stamp.jsp?tp=&arnumber=278253>.   
[3]ACM Code of Ethics, https://www.acm.org/code-of-ethics, ACM Code 2018 Task Force, June 2018. [4] BCS Code of Conduct, https://www.bcs.org/membership/become-a-member/bcs-codeof-conduct/, BCS, The Chartered Institute for IT, 2019.   
[5]Cockburn, Alistair. Writing Effective Use Cases. Boston; London: Addison-Wesley, 2001. Print. Agile Software Development Ser.   
[6]Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston: Addison-Wesley, 2004. Print.

|  |  |  |
| --- | --- | --- |
| USER REQUIREMENTS | | |
| ID | DESCRIPTION | PRIORITY |
| UR\_FIRETRUCKS\_UNIQUE\_SPEC | Each Fire Engine must have a unique spec | SHALL |
| UR\_FIRETRUCKS\_REFILL | Fire Engines need to return to the Fire Station to refill | SHALL |
| UR\_FIRETRUCK\_REPAIR | Fire Engines need to return to the Fire Station to repair | SHALL |
| UR\_ET\_UNIQUE\_SPEC | Each ET fortress must have a unique spec | SHALL |
| UR\_ET\_IMPROVEMENT | Over time the ET fortresses improve and they become harder to flood | SHALL |
| UR\_FIRETRUCK\_MIN\_START | The game should start with four Fire Engines with different specs that are all playable in the game until they are destroyed. | SHALL |
| UR\_ET\_MIN\_START | There should be at least six different ET fortresses based (possibly loosely) on real locations in York | SHALL |
| UR\_WIN\_CONDITION | The game is won when all ET fortresses have been flooded | SHALL |
| UR\_LOSS\_CONDITION | The game is lost when all Fire Engines have been destroyed | SHALL |
| UR\_ET\_DESTROYS\_STATION | After a fixed amount of time following the first attack to an ET fortress, ETs figure out where the Fire Engines are coming from and destroy the Fire Station. From that point onwards, your Fire Engines cannot be repaired or refilled | SHALL |
| UR\_MINIGAME | There should be an embedded mini-game, completely different in style from the main game, but aligned to the theme of the main game | SHOULD |
| UR\_DIFFICULTY\_LEVEL | The game ~~has different difficulty levels for different types of audiences~~ should have easy medium and hard difficulty levels. | ~~MAY~~  SHOULD |
| UR\_CONTROLLER | The game could have controller compatibility | MAY |
| UR\_HIGHSCORE | The game should have a record of high scores | MAY |
| UR\_MOBILE | The game may be cross-platform transferable | MAY |
| UR\_INSTRUCTIONS | The game should have a function at the beginning of the game to explain how it works | SHOULD |
| UR\_GAME\_TIMER | The game's length should be decided keeping in mind the target audience i.e. open days attenders, and is based on the timer that is triggered following the first attack to an ET | SHALL |
| UR\_INTUITIVE | The game should cater to different levels of ability and be unambiguous and intuitive to complete. | SHALL |
| UR\_COLOUR\_ACCESSIBILITY | The game may have a feature for different colours schemes for enhanced accessibility e.g. high contrast colours | MAY |
| UR\_PATROLS | The user shall meet patrols when moving around the map. | ~~SHALL~~  SHOULD |
| UR\_FORTRESS\_ATTACK | The user shall be attacked by the fortresses when they’re in range. | SHALL |
| UR\_FORTRESS | The game should have fortresses | SHOULD |
| UR\_PAUSE | User should be able to pause the game at any point during the game and minigame | SHOULD |
| UR\_SAVE | User should be able to save at any point in the game | SHOULD |
| UR\_POWERUPS | User should be able to pick up power ups | SHOULD |
| UR\_DRIVE | The system shall allow the user to move the fire engines around the map | SHOULD |

|  |  |  |
| --- | --- | --- |
| SYSTEM REQUIREMENTS | | |
| FUNCTIONAL REQUIREMENTS | | |
| ID | DESCRIPTION | USER REQUIREMENTS |
| SFR\_ALLOWED\_TO\_REPAIR | Health Point drop by more than 1 shall lead to Fire-engines able to repair | UR\_FIRETRUCK\_REFILL |
| SFR\_ALLOWED\_TO\_REFILL | Water Tank points dropping by 1 shall lead to Fire-Engines able to refill | UR\_FIRETRUCK\_REPAIR |
| SFR\_REFILL\_FIRETRUCK | Fire engine refills once the minigame is complete. | UR\_FIRETRUCK\_REFILL |
| SFR\_REFILL\_CONSTANT | The refill rate shall be constant | UR\_FIRETRUCK\_REFILL |
| SFR\_REPAIR\_OVER\_TIME | Fire engine repair over time | UR\_FIRETRUCK\_REFILL |
| SFR\_REPAIR\_CONSTANT | The repair rate shall be constant | UR\_FIRETRUCK\_REPAIR |
| SFR\_CANCEL\_REPAIR | The repairing can be stopped at any point during the process | UR\_FIRETRUCK\_REPAIR |
| SFR\_MOVE\_WHILE\_EMPTY | The fire engines shall be able to move even with empty water tank. | UR\_FIRETRUCK\_REFILL |
| SFR\_MOVE\_WHILE\_DAMAGED | The fire engines shall be able to move with HP < 100%. | UR\_FIRETRUCK\_REPAIR |
| SFR\_ET\_IMPROVE\_CONSTANT | The ET fortresses shall improve by a constant amount of HP and damage. | UR\_ET\_IMPROVEMENT |
| SFR\_ET\_IMPROVE\_ | The ET fortresses shall increase in HP and damage dealt over time. | UR\_ET\_IMPROVEMENT |
| SFR\_HEALTH\_BAR | The health bar of the fire engine that is being used should be visible at all times. It should be visual rather than jargon to be understandable to all audiences. | UR\_FIRETRUCKS\_REPAIR |
| SFR\_WATER\_SUPPLY\_BAR | The amount of water currently contained in the tank of the fire engine that is being used should be visible at all times. Again, similar to the health bar should be visual and avoid jargon. | UR\_FIRETRUCKS\_REFILL |
| SFR\_ET\_LOCATIONS\_NOT\_CHANGEABLE | The locations of the fortresses cannot be changed by the user | UR\_ET\_MIN\_START |
| SFR\_FIRETRUCKS\_STATS | The user will​ have four unique fire trucks during the game to switch between. | UR\_FIRETRUCKS\_MIN\_START |
| SFR\_FIRETRUCKS\_SELECTION | The user will have four trucks (lives) to complete the game | UR\_FIRETRUCKS\_MIN\_START |
| SFR\_DESTROYED\_TRUCKS | The user cannot repair trucks that have already been completely destroyed | UR\_LOSS\_CONDITION |
| SFR\_MINIGAME | The minigame should start when the truck is in the fire station, lacking water. The user cannot progress until they have won the minigame. | UR\_MINIGAME |
| SFR\_TIME\_TO\_DEFEAT\_ET | The ET fortresses should take increasingly more time to flood and defeat. The order with which the player will encounter ETs of different difficulties, however, is random i.e. it is based on the player’s movements. | UR\_ET\_IMPROVEMENT |
| SFR\_ETS\_DESTROY\_STATION | The ETs cannot be stopped from destroying the Fire Station | UR\_ET\_DESTROYS\_STATION |
| SFR\_IMPLEMENT\_PATROLS | Patrols should attack as they move around on a set path. | UR\_PATROLS |
| SFR\_FORTRESS\_AIM | The fortress projectiles must aim towards it’s target. | UR\_FORTRESS\_ ATTACK |
| SFR\_POWERUP\_FIRETRUCKS | Going over a powerup will modify the fire trucks ability. | UR\_POWERUPS |
| SFR\_POWERUP\_TYPES | There should be at least 5 different types of power ups. | UR\_POWERUPS |
| SFR\_SAVE\_MULTIPLE | There should be a way to store multiple save files. | UR\_SAVE |
| SFR\_SAVE\_MINIGAME | The game must be able to save during the minigame. | UR\_SAVE |
| SFR\_SAVE\_POWERUPS | When the game is saved the powerups must also be saved. | UR\_SAVE |

|  |  |  |  |
| --- | --- | --- | --- |
| NON-FUNCTIONAL REQUIREMENTS | | | |
| ID | DESCRIPTION | USER REQUIREMENTS | FIT CRITERIA |
| SNFR\_INSTRUCTIONS | Before the beginning of the game, the user should have the choice to read the game instructions | UR\_INSTRUCTIONS | Instructions should cover all features of the game and how they work |
| SNFR\_TARGET\_AUDIENCE | The bullets patterns should present different levels of difficulties e.g. bullets shot in a straight line, bullets shot in a circular pattern, combination of both, etc. Moreover, the movements of the fire truck should be basic and easy to learn, without hidden commands or functionalities | UR\_TARGET\_AUDIENCE | Game should be based on easy to understand rules, fast-paced and with relatively wide range of bullets’ patterns difficulties |
| SNFR\_JARGON | All user-facing messages shall be in plain English and will not use technical videogames jargon | UR\_TARGET\_AUDIENCE | N.A. |
| SNFR\_HIGHSCORES | The game should support the High Scores feature | UR\_HIGHSCORE | The game should have a local record of the top high scores |
| SNFR\_ACCESSIBILITY | The game may have a way to modify the colour pallet to enhance accessibility | UR\_COLOUR\_ACCESSIBI LITY | N.A. |
| SNFR\_MOBILE | The game (style, movement, map visualisation) should be designed with the aim of developing a mobile version | UR\_MOBILE | N.A. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CONSTRAINT REQUIREMENTS | | | | |
| ID | DESCRIPTION | RISKS | ALTERNATIVES | ENVIROMENTAL ASSUMPTIONS |
| SCR\_RUNNABLE | Game shall be runnable on every computer i.e. low-end computer | User's computer not able to support game | N.A | User's computer can run the game |
| SCR\_CONTROLLER | The game should be playable both with keyboards and controller | User does not have controller | Use keyboard instead | User possesses a keyboard |
| SCR\_NO\_BUDGET | The project's budget is 0 | Some technologies, software, libraries might have a price to be accessed and used | Ask for University's financial support or change the technology used | All technology used is free and accessible |
| SCR\_CLIENT\_MEETING | The team should not assume that the client is available every week for meeting, and time between meeting request and date of meeting might vary | Client is never available for meeting and/or client response time is delayed | We can contact the client by email to specify certain functions the game should include. | Client will be available at least once a week to ask questions about the game |
| SCR\_GROUP\_MEETINGS | The team should be able to regularly meet up to agree on design decisions and collate work done. | Group members are not able to attend. | Set up a voice chat channel to allow for all members to discuss development when they are free for a voice chat. | Each group member has a viable way to voice chat. |