Anything added will be highlighted in yellow like so: EXAMPLE
Anything removed will be crossed out and highlighted in yellow like so: ~~EXAMPLE~~

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 The game shall become harder over time.	SHALL
UR_FIRETRUCK_MIN_START	There should be at least four Fire Engines	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	MAY
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_TARGET_AUDIENCE	The game should cater to different levels of ability	SHALL
UR_COLOUR_ACCESSIBILITY	The game may have a feature for different colours schemes for enhanced accessibility e.g. high contrast colours	MAY
UR_DRIVE	The system shall allow the user to move the fire engines around the map	SHALL
UR_PATROL	The game should have ET patrols	SHOULD
UR_FUN	The game should be fun to play	SHOULD
UR_FORTRESS	The game should have fortresses	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_OVER_TIME	Fire engine refills over time	UR_FIRETRUCK_REPAIR
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_CANCEL_REFILL	The refilling can be stopped at any point during the process. i.e. Leaving the station.	UR_FIRETRUCK_REFILL
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 choose the type of fire truck at the beginning of the game	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 be a platform-based game inspired by SuperMario and Flappy Bird	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_ARROWKEYS	The fire engines should be able to move using the arrow keys on the keyboard	UR_DRIVE
SFR_BUILDINGS	Fire engine must not be able to go through buildings	UR_DRIVE

SFR_RIVERS	Fire trucks must not be able to go rivers.	UR_DRIVE
SFR_ENDSCREEN	Game displays a win/lose screen	UR_WIN CONDITION UR_LOSE_CONDITION
SFR_FORTRESS_DESTROY	The fire engines must be able to destroy all the fortresses if they are in range.	UR_ET_MIN_START UR_FORTRESS
SFR_FORTRESS_ATTACK	Fortresses should attack the fire trucks.	UR_FORTRESS UR_FUN
SFR_PATROL_DAMAGE	ET patrols must be able to damage fire trucks.	UR_PATROL
SFR_PATROL_HEALTH	ET patrols cannot be damaged or destroyed.	UR_PATROL
SFR_PATROL_DIFFICULTY	The number of ET patrols will continue as the game goes on	UR_PATROL
SFR_PATROL_FIRESTATION	ET Patrols should be able to destroy the fire station.	UR_ET_DESTROYS_STATION UR_PATROL UR_FUN

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~~ Instructions should be easy to understand.	UR_TARGET_AUDIENCE	~~N.A.~~ All user-facing messages shall be in plain English and will not use technical videogames jargon
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_ACCESSIBILITY	~~N.A.~~ There should be a way to modify the colour scheme in the for people who may be colour-blind.

SNFR_MOBILE	The game (style, movement, map visualisation) should be designed with the aim of developing a mobile version	UR_MOBILE	~~N.A.~~ The game should use an engine which allows you to easily transfer from pc to mobile.
SNFR_TIME	The game should be playable within a reasonable amount of time	UR_FUN	You should be able to finish the game in under 5 minutes.
SNFR_SIMPLE	The game should be simple and easy to understand	UR_FUN	The game should use arrow keys for the controls and the water cannons should be automatic.
SNFR_FORTRESS	To make the game fun, it will have the goal of destroying fortresses	UR_FUN	You are able to destroy all the fortresses in the game.

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.