Requirements

Group 29
Shard Software

James Burnell
Hector Woods
Jensen Bradshaw
Ben Faulkner
Adam Leuty
Jiahao Shang

2a.

We started eliciting requirements by having an interview with the customer. This interview was a mix of open and closed, with the team having a predefined set of questions to ask the customer and then afterwards allowing any team members to ask new questions that may have arisen from how the customer answered the initial questions. We went into the interview willing to listen to the customer to make sure that the system meets the customer's vision. During the open section of the interview, we made sure our questions would prompt the customer rather than expecting the customer to tell us everything they wanted from the get go, to make it easier for the customer to talk about their expectations.

After the interview we had a team meeting to go over any notes we had taken and concentrate them into a first draft of requirements. We then took time to think about the system from a user perspective to help refine certain requirements and split our draft into both user requirements (UR) and system requirements (SR). User requirements consider what a user would need to be able to do using the technology and are generally more surface-level, written from an end-user point of view. System requirements refer to functionality that the system needs in order to fulfil the user requirements, and are written from a more technical point of view. Having these two groups aids in the design of the architecture of the program - the system requirements are more critical to the final product, and the user requirements are built on top of them.

We have also distinguished between functional (requirements that describe what a system should do) and non-functional (requirements that describe how a system works) requirements. Non-functional requirements represent goals that we should orient our system towards whereas functional requirements specify exactly how the system should behave.

We have also noted any environmental assumptions associated with each requirement. In addition we have included any risks associated with implementing specific requirements.

2b.

User Requirements

ID	Functional or Non-Functional (F or NF)	Requirement	Implementation Notes /Environmental Assumptions	Alternatives/Risks
UR1	F	The player should be able to control a boat. Mouse and Keyboard controls must be available.	Assume the user has access to a keyboard and mouse.	An alternative style for the game, spaceships instead of boats, was discussed amongst the team but in the end was decided not to fit the stakeholder's view.
UR2	F	The player should be able to encounter other boats.	Colleges will generate ships nearby that follow the player.	See UR1.
UR3	F	The player should be able to choose a college. There must be at least three colleges in the game.	The player can choose a college from an abstract overview of the map.	Player can choose colleges from a list, however this is arguably less visually appealing.
UR4	F	The player should be able to accumulate points which must be accumulated by the passage of time, and plunder which must be accumulated by destroying colleges.	Self explanatory. The game will give the player a certain number of points each second, and reward plunder when a college is destroyed.	Points/plunder awarded for destroying ships, however implementing ship combat is not one of the requirements. Alternative methods of acquiring plunder e.g. a treasure chest, however the stakeholder didn't mention this as something they wanted to see
UR5	F	The player should be able to complete an objective that is not immediately achievable.	We decided that our objective will be for the player to capture all opposing colleges.	Could consider alternative objectives such as achieving a certain amount of plunder, however we decided this objective best meets the stakeholder's requirements, as another requirement was that it be possible to capture opposing colleges.
UR6	F	The staff should be able to run the game on any desktop OS.	Assume the user is running Linux, MAC or Windows. Assume the user can run OpenGL and thus use libgdx.	Could also support a mobile OS such as Android but this was not specifically required by the stakeholder. Nevertheless, designing with cross-platform support in mind is good practice.
UR7	F	There must be background music and sound effects.	Assume the user can run OpenAl (audio library associated with OpenGL)	Could support alternative libraries, but since Libgdx supports OpenGl, OpenAl is a good choice.
UR8	NF - Game design issue rather than system architecture. Furthermore, playtime	The player should be able to play for 5-10 minutes.	Assume the use of the product is just the uses listed in the product brief such as university open days where players will likely drop in and out during gameplay.	Provide difficulty levels that vary how difficult the game is and thus how long it takes to complete. However this is outside the scope of the brief which is merely designing a game that the player can play for 5-10 minutes.

ID	Functional or Non-Functional (F or NF)	Requirement	Implementation Notes /Environmental Assumptions	Alternatives/Risks
	depends on the user.			
UR9	NF - Art design issue rather than anything to do with the system architecture.	The player should be able to understand the visuals from a few meters away.	Design art so that entities are easy to distinguish from a distance. Assume that 'by a few metres away' we are speaking of the context of visitors to an open day walking past a copy of the game running.	Provide a zoom/scaling option so that when not being played the camera can be 'zoomed in' to give a clearer view of the textures from a distance. However this may cause feature creep and requires someone to zoom in after each session.
UR10	F	The player should be able to access a tutorial to explain the controls.	Assume that controls are consistent between devices, and that all devices are using a standard keyboard layout. Implemented through an overlay which appears when the game starts that details the game's controls.Can be closed with the escape key.	Provide a more in depth tutorial with interactivity from the user. However this risks feature creep.
UR11	NF - Art design issue rather than system architecture.	The player should be able to identify the game without relying on colours.	Use textures that can easily be differentiated from one another.	Feature a colourblind mode that uses colours which are easier for colourblind users to distinguish. However this may be out of scope for our project and thus risks feature creep.
UR12	NF - Level design issue rather than system architecture.	The player should be able to play on a single level.	We will design the game to feature only one level (though this level is randomly generated as specified in requirement UR13)	If the level is poorly designed or visually unappealing this risks the user becoming bored with the game and quitting early.
UR13	F	The user should have variety in gameplay by some form of randomisation.	We will implement this by randomly generating the level at the start of each session.	We are risking feature creep in doing so, however we feel this is justifiable as we want to have an interesting level that meets our stakeholder's requirement for the gameplay having some form of randomisation. Furthermore a randomised level adds some replay value to our game.

System Requirements

ID	Functional or Non-Functional	Requirement	Implementation Notes /Environmental Assumptions	Alternatives/Risks
SR1	F	The system's inputs should have little delay.	Assuming that the game will be run on relatively modern hardware. Write all control methods in an efficient manner to reduce input delay.	Risk of over-engineering control methods for efficiency when simpler methods might suffice.
SR2	F	The system should have AI, complex enough to mimic a ship.	Very simple AI where the ship moves towards the player and circles them, maintaining a set distance.	Could implement simple pathfinding methods such as A*, however this risks overengineering and feature-creep. Furthermore as most of the map is navigable (there is much more water than land) there is not much need for complex pathing.
SR3	F	The system should provide a generated list of colleges.	A list of colleges is visible on the College display detailed in requirement UR3	Provide a list of colleges as a textual representation on the screen, however this is less visually appealing and may take up too much screen space.
SR4	F	The system should be able to display an objective.	Objective described in UR5. The game should display the colleges remaining as text on one side of the screen	One risk is that the objective may be unclear or that the user might not notice the display.
SR5	F	The system should be OS independent.	Write code in an OS independent manner, use libraries that have cross platform support. As we are using java this shouldn't be much of an issue.	Writing OS-independent code could limit some of our design decisions, since we may not be able to use a useful library that is only available on one operating system. However Java as a language is designed with cross-platform compatibility in mind.
SR6	F	The system should have mute-able audio.	Audio should be mutable from a button on the screen	Could provide a volume slider, however this risks feature creep and additionally could take up too much screen space.