Requirements

Group 12

Team 12

James Leney
Daniel Agar
Charles Thompson
Leyi Ye
Vanessa Ndiangang
Matt Smith

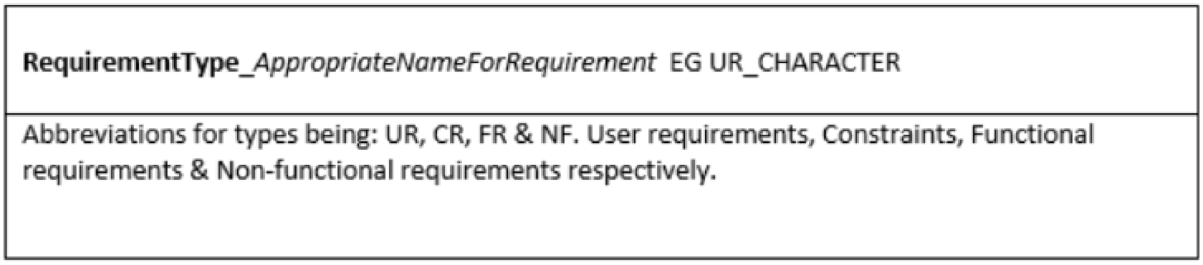
Initially, the user requirements were elicited from the project brief. Each person reviewed the brief individually and attended a meeting soon after to discuss thoughts on the user requirements. These were initially very simple and focused on the game's core mechanics. From this, gaps in our understanding of the product were identified and a list of topics/questions to ask the customer to refine our understanding of the requirements was formed.

In our first meeting, the team used a use-case approach [1] where we discussed how the user would move through the game, what they would see, what they would encounter, and how they would interact with the game as potential use cases. We noted our thoughts on requirements based on the use cases we explored and clarifying questions we wished to ask the customer.

The customer meeting involved two types of interview questions, a notion taken from Sommerville chapter 4 [2]. We sought answers to our list of questions and we aimed to gain an understanding of the customer's vision of the game by asking more general, open-ended questions such as “How does each action affect the final score and do some affect it more than others?” as opposed to a question with a discrete answer such as “Should movement be done with the WASD keys or the mouse?”. From our questions, we understood the user requirements and assigned them a priority based on how critical achieving them was to producing the game.

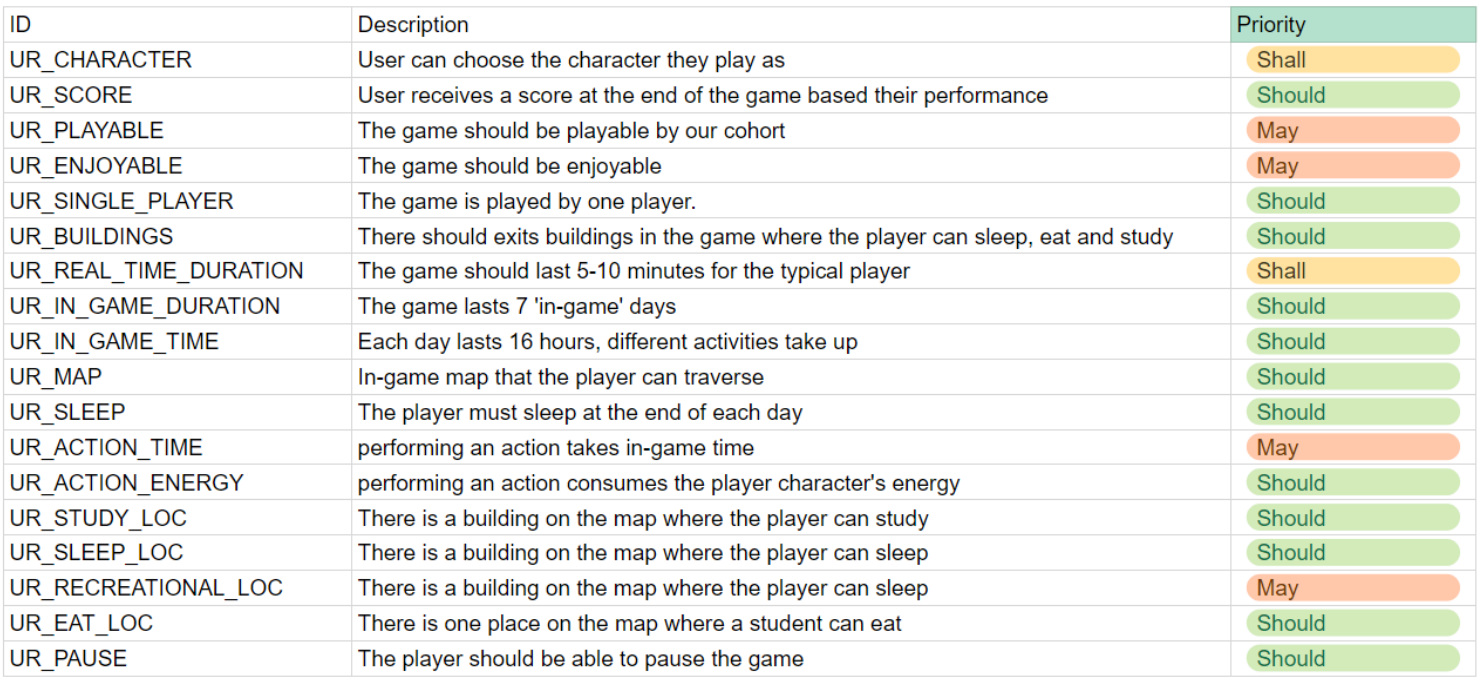
From this, we derived further user requirements and could then move forward with the system requirements. We further categorised our system requirements into functional and non-functional requirements [2]. Most of our functional requirements map directly to how the customer wants certain features implemented. We then chose to split the requirements into user and system. User requirements were high-level requirements of what the customer wished to achieve from the product and system requirements were the low-level requirements required to achieve each user requirement. Non-functional requirements refer to constraints imposed on the system by the customer such as play-through time and the platform on which the product is meant to run. The constraints table includes any limitations external to the product and more specifically within the team.

We sorted our requirements into tables and established a naming convention as described in the figure below. This enabled us to reference the concerned user requirements next to our non-functional/functional requirements.



RequirementType_*AppropriateNameForRequirement* EG UR_CHARACTER
Abbreviations for types being: UR, CR, FR & NF. User requirements, Constraints, Functional requirements & Non-functional requirements respectively.

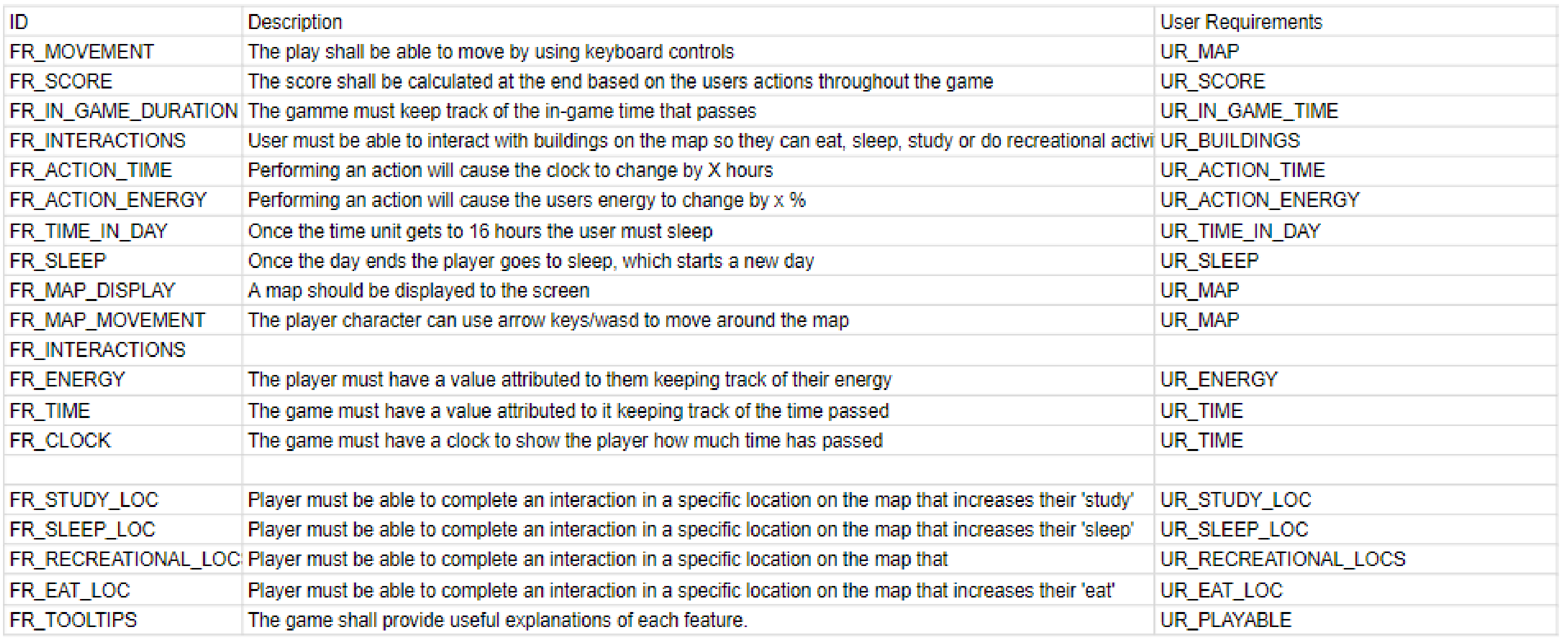
User requirements



ID	Description	Priority
UR_CHARACTER	User can choose the character they play as	Shall
UR_SCORE	User receives a score at the end of the game based their performance	Should
UR_PLAYABLE	The game should be playable by our cohort	May
UR_ENJOYABLE	The game should be enjoyable	May
UR_SINGLE_PLAYER	The game is played by one player.	Should
UR_BUILDINGS	There should exits buildings in the game where the player can sleep, eat and study	Should
UR_REAL_TIME_DURATION	The game should last 5-10 minutes for the typical player	Shall
UR_IN_GAME_DURATION	The game lasts 7 'in-game' days	Should
UR_IN_GAME_TIME	Each day lasts 16 hours, different activities take up	Should
UR_MAP	In-game map that the player can traverse	Should
UR_SLEEP	The player must sleep at the end of each day	Should
UR_ACTION_TIME	performing an action takes in-game time	May
UR_ACTION_ENERGY	performing an action consumes the player character's energy	Should
UR_STUDY_LOC	There is a building on the map where the player can study	Should
UR_SLEEP_LOC	There is a building on the map where the player can sleep	Should
UR_RECREATIONAL_LOC	There is a building on the map where the player can sleep	May
UR_EAT_LOC	There is one place on the map where a student can eat	Should
UR_PAUSE	The player should be able to pause the game	Should

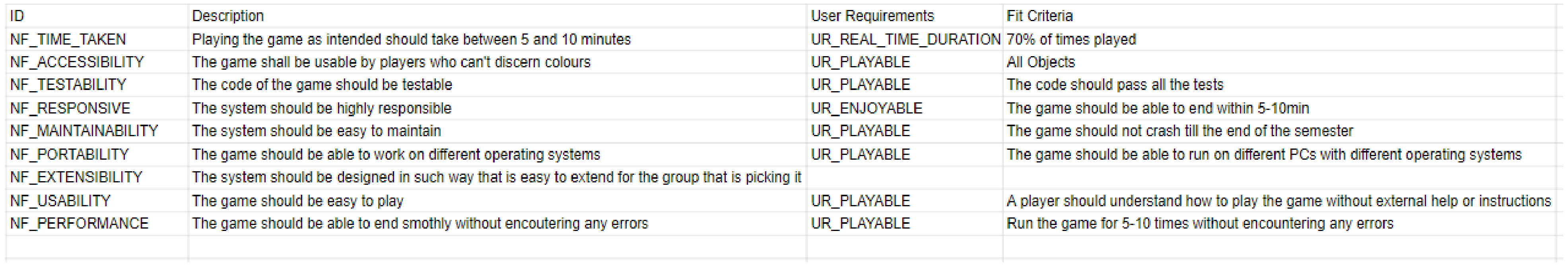
System requirements

Functional requirements



ID	Description	User Requirements
FR_MOVEMENT	The play shall be able to move by using keyboard controls	UR_MAP
FR_SCORE	The score shall be calculated at the end based on the users actions throughout the game	UR_SCORE
FR_IN_GAME_DURATION	The gamme must keep track of the in-game time that passes	UR_IN_GAME_TIME
FR_INTERACTIONS	User must be able to interact with buildings on the map so they can eat, sleep, study or do recreational activi	UR_BUILDINGS
FR_ACTION_TIME	Performing an action will cause the clock to change by X hours	UR_ACTION_TIME
FR_ACTION_ENERGY	Performing an action will cause the users energy to change by x %	UR_ACTION_ENERGY
FR_TIME_IN_DAY	Once the time unit gets to 16 hours the user must sleep	UR_TIME_IN_DAY
FR_SLEEP	Once the day ends the player goes to sleep, which starts a new day	UR_SLEEP
FR_MAP_DISPLAY	A map should be displayed to the screen	UR_MAP
FR_MAP_MOVEMENT	The player character can use arrow keys/wasd to move around the map	UR_MAP
FR_INTERACTIONS		
FR_ENERGY	The player must have a value attributed to them keeping track of their energy	UR_ENERGY
FR_TIME	The game must have a value attributed to it keeping track of the time passed	UR_TIME
FR_CLOCK	The game must have a clock to show the player how much time has passed	UR_TIME
FR_STUDY_LOC	Player must be able to complete an interaction in a specific location on the map that increases their 'study'	UR_STUDY_LOC
FR_SLEEP_LOC	Player must be able to complete an interaction in a specific location on the map that increases their 'sleep'	UR_SLEEP_LOC
FR_RECREATIONAL_LOC	Player must be able to complete an interaction in a specific location on the map that	UR_RECREATIONAL_LOCS
FR_EAT_LOC	Player must be able to complete an interaction in a specific location on the map that increases their 'eat'	UR_EAT_LOC
FR_TOOLTIPS	The game shall provide useful explanations of each feature.	UR_PLAYABLE

Functional requirements



ID	Description	User Requirements	Fit Criteria
NF_TIME_TAKEN	Playing the game as intended should take between 5 and 10 minutes	UR_REAL_TIME_DURATION	70% of times played
NF_ACCESSIBILITY	The game shall be usable by players who can't discern colours	UR_PLAYABLE	All Objects
NF_TESTABILITY	The code of the game should be testable	UR_PLAYABLE	The code should pass all the tests
NF_RESPONSIVE	The system should be highly responsible	UR_ENJOYABLE	The game should be able to end within 5-10min
NF_MAINTAINABILITY	The system should be easy to maintain	UR_PLAYABLE	The game should not crash till the end of the semester
NF_PORTABILITY	The game should be able to work on different operating systems	UR_PLAYABLE	The game should be able to run on different PCs with different operating systems
NF_EXTENSIBILITY	The system should be designed in such way that is easy to extend for the group that is picking it		
NF_USABILITY	The game should be easy to play	UR_PLAYABLE	A player should understand how to play the game without external help or instructions
NF_PERFORMANCE	The game should be able to end smothly without encoutering any errors	UR_PLAYABLE	Run the game for 5-10 times without encountering any errors

Constraints

ID	Description
CR_PLATFORM	Desktop game
CR_TIME	Development activity is limited to 6 weeks
CR_EXPERIENCE	The experience of our team is indirect in relation to the product and limited.
CR_STAFF	Maximum of 6 people working on all deliverables
CR_QUALITY	Always have at least 2 team members on programming and is to be reviewed by the other 4 members of the team periodically
CR_BUDGET	There is no budget from the team or the customer
CR_SCOPE	Develop 4 actions the player can do and different UIs to interact with based on game states
CR_RESOURCES	LIBgdx, IntelliJ, Notion, GoogleDocs and various assets
CR_RISK	Limited by the risks determined and listed in the risk register

References

[1] I. Jacobson, M. Christerson, P. Jonsson and G. Overgaard, *Object-Oriented Software Engineering : a Use Case Driven Approach.* New York: ACM Press, 1992.
[2] I. Sommerville, *Software Engineering*. 10th ed. Essex: Pearsons Education Ltd, 2016.