Assessment 1: Greenfield Development - Requirements

Cohort 1 Team 2

Team Members:

Trace Chinelle
Vidhi Chohan
Apollo Cowan
Siyuan Liu
Aryaman Marathe
Charlie Mason

Requirements A

Our requirements elicitation process began by analysing the product brief and forming a list of questions to ask at the client meeting to clarify the scope and requirements of the project. After the client meeting we had a group discussion to pitch our own ideas and to identify and fill in any gaps in our initial plan.

After we had defined the basic outline of our requirements, we researched the most effective way to explore our requirements comprehensively. Jacobson recommends a use case driven approach to requirements elicitation [1]. This method allowed us to understand and represent all of our requirements from a user perspective. Cockburn provides a number of different ways to write use cases [2] so we followed his guidance to write a series of use cases that are both comprehensive but not unnecessarily complex. We have included two use cases in this document to demonstrate the format and method that we used. A full list of all use cases can be found on our website.

Our final step in our requirements process was to create a series of requirements tables that we could reference throughout the project. We created three tables for this purpose - User Requirements, Functional Requirements, and Non-Functional Requirements. We formed the lists of requirements by identifying the key points in our use cases and breaking them down into straightforward, testable requirements.

In the User Requirements table, we provided a short description of each requirement and its priority in the project. We also included a unique ID for each requirement which we could reference in the other tables and throughout the project for traceability. We used a simple descriptive naming convention for all of the User Requirement IDs to prevent any unnecessary confusion.

In the Functional Requirements table, we provided a brief description and unique ID, following a similar descriptive naming convention to the User Requirements, for each requirement. We also included the ID of the User Requirement that each Functional Requirement linked back to.

In the Non-Functional Requirements table we provided a brief description and unique ID for each requirement along with the User Requirement ID it links to, similar to the Functional Requirements table. Since the Non-Functional Requirements are less quantifiable than the Functional Requirements we opted to also include a fit criteria for each requirement. This gave us a clear testable way of knowing whether each requirement had been met.

Requirements B

User Requirements Table		
ID	**Description**	**Priority**
UR_CHOOSE_ PLACE	Users shall be able to choose where they want to place a building/ area	Shall
UR_RESTRICTED_ARE A	Buildings shall not be placed in restricted areas.	Shall
UR_UX	The system shall offer the user a pleasant and fun user experience.	Shall
UR_PAUSE	The system shall allow users to pause the game.	Shall
UR_THREE_ EVENTS	The user shall experience at least 3 events during the course of the game (either positive, negative or neutral).	Shall
UR_PLACE_ ACTIVITY	Users shall be able to place 2 buildings/ areas for recreational activities.	Shall
UR_PLACE_ FOOD	Users shall be able to place at least one food hall for students.	Shall
UR_PLACE_ SLEEP	Users shall be able to place at least one sleeping area for students.	Shall
UR_PLACE_ STUDY	Users shall be able to place at least one study area for students.	Shall
UR_REACT_ TO_EVENT	Users shall be able to react to the events appropriately.	Shall
UR_DURATION	The system shall last 5 minutes.	Shall
UR_SS_SCORE	The system shall update the student satisfaction score as users make changes and react to events.	Shall

Functional Requirements Table		
ID	**Description**	**UR ID**
FR_SNAP_GRID	The buildings/areas should snap to grid when being placed.	UR_CHOOSE_ PLACE
FR_PLACE_ CONFIRMATION	The system shall provide confirmation facilities when users are placing a building/ area down.	UR_CHOOSE_ PLACE
FR_BUILD_ TIME	The system shall take a certain amount of time to “build” a new building/area, users must wait until it is done to use it.	UR_CHOOSE_ PLACE
FR_ERROR	The system shall provide an error message on the screen when players try to place a building/area on a restricted area of the map.	UR_RESTRICT ED_AREA
FR_UX_DESIGN	The game design will be easy to use and visually appealing to look at.	UR_UX
FR_PAUSE	The system will allow the user to pause the game. It will save their progress and stop the countdown/timer.	UR_PAUSE

Requirements B

FR_POS_EVENT	The system shall include at least one positive event, this should increase the satisfaction score.	UR_THREE_EVENTS
FR_NEG_EVENT	The system shall include at least one negative event, this should decrease the satisfaction score.	UR_THREE_EVENTS
FR_NTRL_EVENT	The system shall include at least one neutral event, this should not affect the satisfaction score.	UR_THREE_EVENTS
FR_PLACE_ACTIVITY	The system shall allow the user to place at least 2 recreational activities, whether the user places buildings/ areas as long as they are placed within the map boundaries and not in a restricted area.	UR_PLACE_ACTIVITY
FR_PLACE_FOOD	The system shall allow the user to place at least 1 area/ building to eat in, as long as it is placed within the map boundaries and not in a restricted area.	UR_PLACE_FOOD
FR_PLACE_SLEEP	The system shall allow the user to place at least 1 area/ building for living/sleeping, as long as it is placed within the map boundaries and not in a restricted area.	UR_PLACE_SLEEP
FR_PLACE_STUDY	The system shall allow the user to place at least 1 area/ building to study in, as long as it is placed within the map boundaries and not in a restricted area.	UR_PLACE_STUDY
FR_REACT	The system shall allow the user to react to all 3 types of events appropriately, giving them a choice between multiple options	UR_REACT_TO_EVENT
FR_DURATION	The game shall last 5 minutes.	UR_DURATION
FR_SS_SCORE	The game shall update the satisfaction score each time the user makes changes to the campus and reacts to events.	UR_SS_SCORE
FR_DISPLAY_SCORE	The system shall display the satisfaction score so the user can see how their actions affected the simulator.	UR_SS_SCORE

Non-Functional Requirements Table			
ID	**Description**	**UR ID**	**Fit Criteria**
NFR_ACCESSIBLE	The system shall be operable by users with impaired vision and hearing.	UR_UX	90% of users will be able to play the game without their impairments hindering them.
NFR_AVAILABILITY	The system shall be available to users without crashing at any time.	UR_UX	The game shall load within 10 seconds and not crash 90% of the time.
NFR_MAINTAINABILITY	The engineers shall be able to diagnose and repair bugs in the system remotely.	UR_UX	The bugs should be fixed within 2 weeks and the updated version should be released.
NFR_SCALABILITY	The system shall support multiple individual players at one time.	UR_UX	The system should not crash when there are >20 different users playing the game at once.
NFR_OPERABILITY	The system shall be operable by users that have read all of the information bubbles.	UR_UX	The system shall be operable by 99% of users after reading the information bubbles.

Requirements B

NFR_USABILITY	All information shall be clear and not use technical jargon.	UR_UX	99% of users should be able to understand messages/ information after reading it >3 times.
NFR_AUDITABILITY	The system shall maintain a record of the users' progress at the time of pausing the game.	UR_PAUSE	The system shall keep the current progress of users at the time of pausing.

Use Case - Place Building

Primary actor: Player

Scope: Game **Level:** User Goal

Stakeholders: Player: Wants to add a new building to the map.

Precondition: Game is not paused **Trigger:** Player decides to place a building.

Minimal Guarantees: No building is placed and Player's funds remain unchanged.

Success Guarantees: New building is visible on map after short build time; position of new building has effect on the score; building counter goes up; position of new building has effect on score.

Main Success Scenario:

1. *Player:* select building type from list.
2. *Player:* choose location for building on map grid.
3. *Player:* pay for building.
4. *Building Counter:* increase by one.
5. *Map:* display building in a construction state in the chosen location for a predetermined building time.
6. *Map:* display building in finished state in the chosen location.
7. *Score:* calculate score.

Extensions:

1a. Player does not have sufficient funds: Do not allow Player to select the building type

2a. The chosen location is invalid: Prompt Player to choose a new location or cancel.

5a. Game ends during the building time: Do not do successive points.

Secondary Actors: None

Use Case - Pause

Primary actor: Player

Scope: Game **Level:** User Goal

Stakeholders: Player: Wants to pause the timer.
Timer: Wants to stop counting down while the game is paused.

Precondition: Game is not paused **Trigger:** Player decides to pause game.

Minimal Guarantees: Game remains unpaused.

Success Guarantees: Timer stops counting down; displays unpause button; Player is not able to interact with any elements apart from the unpause button.

Main Success Scenario:

1. *Player:* interact with the pause button.
2. *Timer:* stop counting down, remember its current time.
3. *Map:* display unpause button, do not allow player to interact with any other elements.

Extensions: None

Secondary Actors: Timer

References

[1] I. Jacobson *Object-oriented software engineering : a use case driven approach.* Wokingham, Eng. ; Reading, Mass.: ACM Press ; Addison-Wesley Pub., 1992
[2] A. Cockburn *Writing Effective Use Cases.* Boston; London: Addison-Wesley, 2001