**Group # 5 - LIVE MULTIPLAYER GAME**

**Mohammad Asim Iqbal BSCS14002**

**Mahad Saleem BSCS14008**

**M. Murtaza Azam Khan BSCS14027**

**Omer Zaman BSCS14046**

**Software Engineering**

**ASSIGNMENT 2 – REQUIREMENTS ENGINEERING**

**Introduction**

This software system will be an Online Multiplayer Battle game for fun loving and thrill seeking people. This system will be designed to maximize the joy and fun for the users in their leisure time. It will allow users to flee the usual thrust of life and have a few moments of joy.

More specifically, this system is designed to allow users to experience real-time multiplayer war game. The idea of this game is to engage two players in a battle, connected through internet, with the objective of destroying opponent’s military base. Each player will be allowed to build a deck of six cards to play with. There will be multiple cards, each having its own unique attributes. The player has to defend his base as well as attack opponent’s using the cards in the deck.

*Incorporate the feedback that you have been given in the previous assignment.*Already complete.

*Identify which of the following techniques do you think are useful for requirements elicitation in your project. Make sure that you justify your answer. More than one techniques may be applicable in your project.*

*■ Interviews*

*■ Scenarios*

*■ Use Cases*

*■ Ethnography*

For this project, scenarios would be the best approach towards requirements elicitation. Each scenario description will reflect various transitions in the game starting with the game state at the beginning of the scenario and ending with the game state at the end of the scenario, which a user will reach after a sequence of actions from the starting state. Following is an example of a scenario that can occur in the game:

**Scenario for user engaged in a battle in the game**

**Initial assumption:** The user has an android device running android 4.2 or above. Minimum processor speed should be 1 GHz with 512 MB memory and 100 MB. The user must also have a google id associated with their device so that they can download the game from google play store and save their game progress. An internet connection is necessary to play the game. Before starting a battle, user will have selected a deck of six cards that they will use in their battle.

**Normal:** A player wants to use a card to deploy troops to attack the enemy tower. To do so, the player will have to accumulate energy equal to the card’s cost. It is also necessary the desired card is part of the deck and is one of the three cards visible to the player. There are two ways in which the player may deploy the card. They can either tap on a card to select it and then tap on map at the position where they want to deploy the troop or they can drag the card towards the position where they want to deploy the troop. Once a card has been deployed, the troop will move towards the enemy tower and may or may not engage with the opponent’s troops it encounters in its path, depending on the troop’s characteristics.

**What can go wrong:** The opponent deploys their troop in the player’s path. The player should deploy more troops to ensure that the enemy troops are destroyed and the troops deployed by the player initially reach the enemy’s tower. The player can also use the decoy strategy to lure the enemy troops out of the path of the first troops deployed by the player.

The player does not have enough energy to deploy that troop. In this case, either the player will have to wait until they have sufficient energy or they will have to use another card.

They player does not have the desired card in their deck. In this situation, the only option the player has is to use the cards that are part of their battle deck.

**Other activities:** Opponent may continue to deploy troops while the user waits to deploy theirs. The battle timer will continue to expire. Energy will continue to increase until maximum energy is reached.

**System state on completion:** The intended troop will have reached the enemy tower and will attack it until the player or the tower loses all its hit points.

*Game developer: pick the relevant “lenses” as discussed in the class and identify requirements of your game.*

**The Lens of Emotion:** I would want the player to enjoy the game and arouse his curiosity along with an element of surprise. That would ensure that the game’s concept is fresh even if the user continually plays it. It would also maintain the level of interest. The game should be able to pacify the user’s enthusiasm and instill a level of satisfaction. A good experience will leave a lasting impression and hook the user to the game. Putting this in perspective, we will be adding new characters (playable) every three weeks. The user would unlock them without knowing which one, this would bridge the element of surprise and curiosity. A good experience will be had due to the infinite possibilities that would be in play during the battle sequence. The user would be playing under different circumstances each time due to a great number of character combinations being possible and the nuance in player tactics. This would keep it fresh and maintain the zeal. Also, this would serve as an incentive for the player to return to the game every now and then.

**The Lens of Surprise:** The game will be updated every 3 weeks with new characters. These characters will enable the users to come up with new strategies. The users have no control with respect to which character they unlock. They will be randomly awarded to them after they gain a handsome amount of experience or collectibles in the game. It is a multiplayer game and each time the system selects (applying certain filters) two players to play against each other, the users will be oblivious to the type of characters the opponent is playing with. It would unroll only during the battle sequence hence the user would have to adapt to a new situation every time and muster up a new game plan. Hence, the game provides various instances of surprise.

**The Lens of Fun:** There are many aspects of the game which can be deemed as fun. Playing with your friends online, engaging in a battle for strategic supremacy. Trying different combinations of characters and playing against various other ones is interesting and fun. Allowing players to socialize and send emojis could make the game more entertaining. Adding multiple landscapes (or arenas) would also improve the recreational prospect of the game.

**The Lens of Curiosity:** The players will be categorized according to the experience points they have gained and the number of characters (cards) they have unlocked. Players with more experience points will be able to unlock stronger characters or newer ones. This will provide new users with an objective to climb the hierarchy. The user will have to question his strategy or wonder about the new characters that will be introduced. How will the user improvise under strenuous circumstances? The game will constantly provide incentives to the users by offering them new cards in exchange for the game’s currency. This will motivate the new users to continue on their plan. With the introduction of new landscapes which would affect the gameplay, new questions could be invented for the players.

**The Lens of Endogenous Value:** Things valuable to the players are the characters (cards), the experience points, in-game subpar and premium currency. These can be made more important to the user by offering upgrades for the characters, by rewarding in game currencies on every level up. Allowing players to buy new cards using the in-game currencies. This will motivate the player to level up and accumulate the currencies in order to buy new stuff and for upgrades.

**The Lens of Problem Solving:** The game primarily focuses on this aspect. The whole battle sequence is tailored by a set of actions that need to be taken after careful consideration. In order to be tactful, the player must solve the problem of navigating through the opponent’s defense and taking down the base camp. There is no hard and fast rule to cracking this code. Each battle may require a different strategy and skill. There is another problem the user needs to solve, which is coming up with an effective combination of cards. The user needs to develop his deck in accordance to the strategy he intends to follow.

*Requirements Grouping*

* *Categorize your system requirements into functional and non-functional requirements.*
* *For each of the non-functional requirements, also identify its type (Figure 4.3 Types of non-functional requirements)*
* *For each non-functional requirements, define metrics that can be used to measure it (Figure 4.5 Metrics for specifying non-functional requirements)*

The following are the **functional requirements**:

1. Starting with the main menu, the user should be able to navigate between five screens:
   1. Decks: The user must have the ability to configure their battle decks, upgrade their cards, view locked, unlocked and upcoming cards and unlock new cards if enough resources are present. Each new player will be given six randomly selected cards out of twelve to start their game. They will unlock new cards as they make progress. Once they unlock cards, they will have the option to configure their decks. Each deck has exactly six cards.
   2. Store: On this screen, the user should have the ability to purchase items using in-game currency or real currency.
   3. Main screen: This is where the players will be able to see their own statistics by clicking on the “Stats” button and start a new game by clicking on the “Play” button. This screen will also contain the button for the “Settings” screen.
   4. Settings: This screen will allow the user to toggle music and sound effects on or off, also view the help, and support documents.
2. The user should be able to use their existing google play accounts to login and out of the game. The account will keep track of their progress, which will be used for matchmaking.
3. Battles will begin with each player having five out of ten energy units.
4. The battle will have a three-minute timer. During the first two minutes of a battle, each player will get one energy resource every 3 seconds. After the first two minutes have expired, the players will get one energy resource every one and a half seconds.
5. After the three-minute timer has expired, the game will switch to sudden death mode. In this mode, both players’ towers will lose equal number of health points per second and the player to lose their tower first loses the game.
6. The battlefield shall be divided into 36 rows and 18 columns. Each player will have 18 rows and 18 columns as their “side”. Each player will have three guarding posts protecting their base. The base shall be located at the near end of the battlefield. The health points of the base will be determined by the player rank.
7. Each player will have access to three out of the six cards in their deck.
8. Player should be able to deploy troops either on their side of the base or anywhere on the map (depending on the troop characteristics).
9. Troops include tanks, helicopter, f-16, commando, guerrilla, sniper, nuke, drone, paratrooper, EMP, Humvee and sentry. Each of these troops will have their unique characteristics and varying energy costs. Troops shall also be categorized into ranged, melee, ground only, ground + air, structures only, single combat and splash damage troops.
10. Players should have the option to quit a game and return to the main menu.
11. Whoever destroys their opponent’s base first shall be declared the winner of the game.
12. Players shall be rewarded with currency, experience points and rank points for each victory.
13. Players shall lose rank points for each loss.
14. Players should be prompted if they or their opponent disconnect/reconnect to the game.

The following are the **non-functional requirements**:

**PRODUCT REQUIREMENTS**

**Security requirements:**

1. Users must secure login using Google Play.
2. All the user data shall be kept on a private server and no user will be able to access it.
3. User shall not be able to play the game without logging in the system.

**Usability requirements:**

1. The system will be available for all users older than 7 years of age.
2. The system will be easy to use. User interface will be kept simple.
3. The system will provide help/tutorial to the user where needed.
4. The system will be available 24/7 and downtime shall be kept no more than 10 mins for maintenance.
5. The system shall be self‐explanatory and intuitive.

**Efficiency requirements:**

**Performance requirements:**

1. The system restart cycle must execute completely in less than 60 seconds
2. The system shall take no more than 80% of processor capacity at full load.
3. The system shall generate notification within one minute of generation.
4. The system shall be able to handle at least 1000 concurrent users over the network.
5. The network latency shall be kept minimal.
6. The network server shall be kept operational 24/7. It must not have disconnections.
7. The system should be able to recover gracefully from a fault.

**Storage requirements:**

1. The system shall not take more than 150 MBs of disk storage.
2. Main memory (RAM) shall be least used, i.e, no more than 1 GB.
3. The system shall generate low storage notification if it is accessing more storage than defined threshold.

**Dependability requirements:**

1. The system shall be kept reliable and must deliver authentic data.
2. The system shall be kept fault tolerant and must recover from a fault gracefully.
3. The system shall be kept intelligent and should be able resist against security breaches.

**ORGANIZATIONAL REQUIREMENTS**

**Environmental requirements:**

1. The system must be available for only Android devices.
2. The system shall be able to run on any Android device by any manufacturer.
3. There must be a private server for the system.

**Operational requirements:**

1. Users must authenticate into the system using their Gmail IDs.
2. There must be a system administrator which can manage the user data.
3. Regular updates shall be released.

**Development requirements:**

1. The system shall be built using Unity 3D game Engine.
2. C# programming language will be used to develop the game.
3. The system shall be thoroughly tested using the software testing techniques.

**EXTERNAL REQUIREMENTS**

**Regulatory requirements:**

1. Data shall be kept in encrypted form.
2. Advertising laws shall be kept intact.
3. This software system shall not be used for gambling.

**Ethical requirements:**

1. User data will be kept private, no data will be available for public.

**Legislative requirements:**

**Accounting requirements:**

1. User credentials used for in-app purchases shall be required and will be kept private.

**Safety/Security requirements:**

1. The system shall not be available for ages less than seven years.

**Metrics for specifying Non-functional requirements**

**PRODUCT REQUIREMENTS**

**Security requirements:**

These requirements cannot be measured.

**Usability requirements:**

Depending upon the user feedback, we can check if the game is easy to play or not.

**Efficiency requirements:**

**Performance requirements:**

Latency, FPS, Loading time, event response time will allow us to measure the performance of our system.

**Storage requirements:**

The size of game must not exceed the specified threshold.

**Dependability requirements:**

In case of a failure, mean time to failure will be calculated and minimized. Rate of failure will also be monitored.

**ORGANIZATIONAL REQUIREMENTS**

**Environmental requirements:**

Number of devices on which the system can run will be monitored.

**Operational requirements:**

The uptime of the system shall be noted.

**Development requirements:**

These requirements cannot be measured.

**EXTERNAL REQUIREMENTS**

**Regulatory requirements:**

Data traffic and analysis can be used to keep intact the regulatory requirements.

**Ethical requirements:**

These requirements cannot be measured.

**Legislative requirements:**

These requirements cannot be measured.

*Requirements Validation*

*Consistency Checks*

*Identify conflicts in your requirements, if any. Document the conflicts and the mechanism you used to fix them.*

There are no conflicts in the requirements we have documented so far.

*Realism Checks*

*Can you actually build all the requirements using the current technology?*

Yes. We will be using Unity gaming engine for game development, Blender for humanoid creation. We will use FL Studio 12 for background music and SFX. Some of the assets have been bought from online asset stores.

*If there are any requirements that you cannot build, how can you possible modify them to make them realistic?*

Currently there are no requirements that cannot be achieved.

Verifiability Checks

Which of the following techniques you think could be used to check verifiability of your system requirements and why:

For this game, prototyping is a suitable technique for requirements verifiability. Game is a complex software in which various problems emerge while development. Because of that, a prototype can be a best solution to test the requirements, if the requirements are not conflicting and can be implemented. A small prototype of the game can highlight various properties of requirements, whether the requirements are right and the right system can be built using these.