NTNU
Norwegian University of
Science and Technology

Duel of Fates

Requirements

Group 13

Hassel, William Strand

Marleau, William Andreas Krohn

Nguyen, Ken Tan van

Rigatos, Dionysios

Rosset, Adrien

Somorjai, Márk István

Primary Quality Attribute

Modifiability

Secondary Quality Attribute

Usability

COTS

Android SDK, LibGDX, Firebase

1. Introduction

1.1 Project & Phase Description

The purpose of this project is to utilize architectural techniques to develop a multiplayer Android game application. This specification document will serve as a functional requirement and quality attribute reference and documentation. With the conclusion of this document and the specification of the bar, the application should be, a well-defined architecture will be designed to satisfy these requirements.

1.2 Description of the game concept

The application will be a 1-versus-1 duel card game inspired by the popular game *Slay the Spire's* [1] combat system, which served as a reference for both gameplay and artwork. You can choose to be a brave knight, a wise mage or a devious skeleton - each one with its benefits and drawbacks. In the arena, you can choose to save up your mana and perform a chain of attacks in your turn, or consistently make moves to keep your opponent on their toes. Depending on your class, you have certain hidden aces in your sleeve that can turn the tables of the battle.

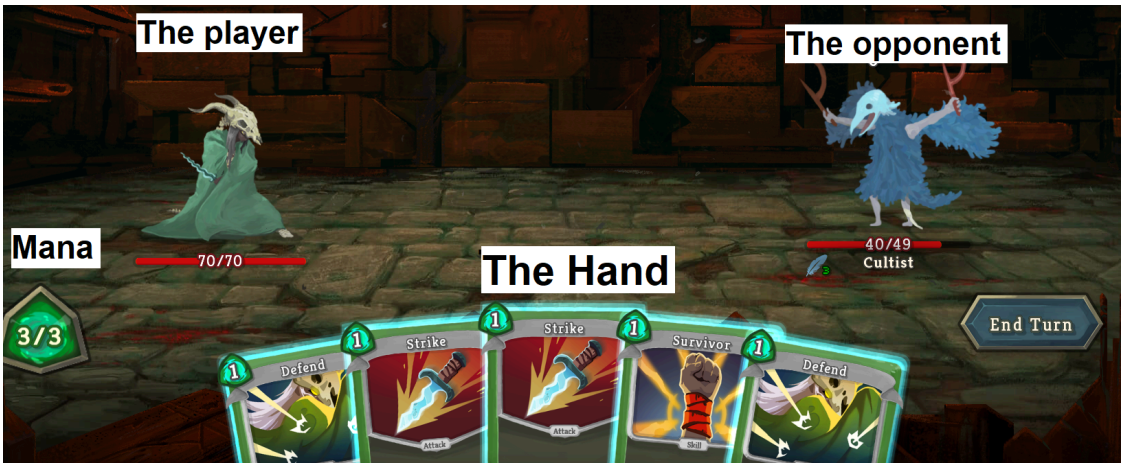

Figure 1 - Screenshot of the interface of "Slay the Spire" from Mega Crit Game

The arena is brutal - either one or no one will make it out alive. This is why it is important to keep track of creeping lasting effects the opponent has cast on you. When it's a player's turn, they can choose to play any of the cards on their hand they can afford with their current level mana. Multiple cards can be played in a single turn, thus making the management of mana particularly important. The effects of the cards are immediately cast on the opponent. As soon as the round concludes, the opposing side has their passive mana refill and gets a chance to take a shot. The game ends when either player runs out of health.

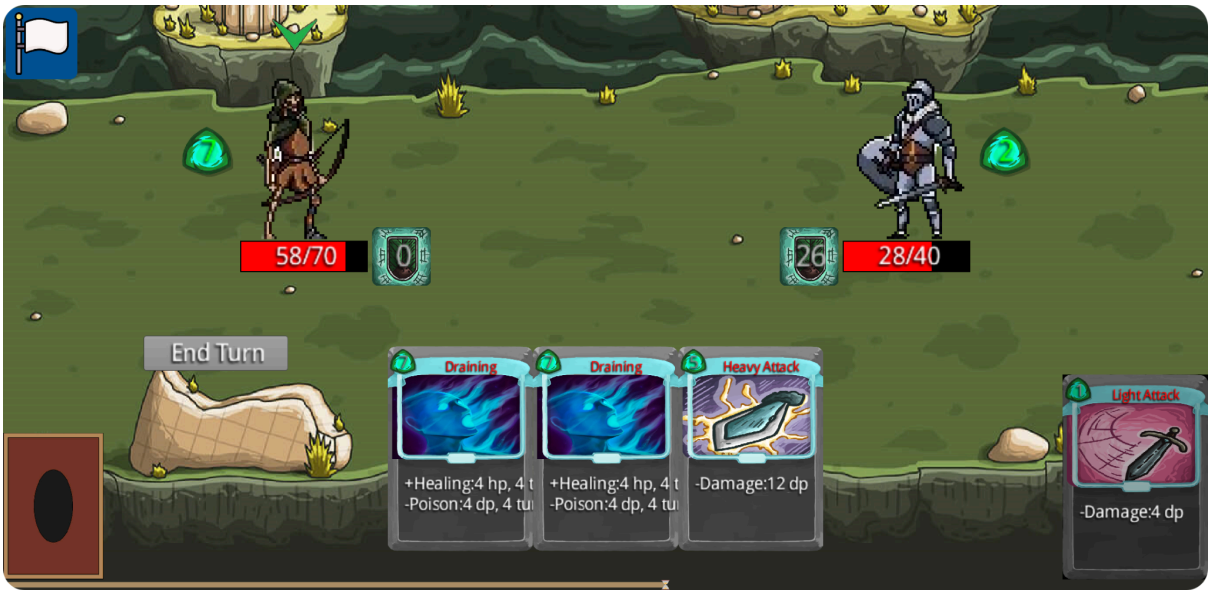

Figure 2 - Screenshot from a battle in Duel of Fates

2. Functional Requirements

Priorities

The functional requirements are classified by three priorities: High, Medium, and Low.

1. High priority means that the requirement is essential to the application's functionality.
2. Medium priority means that the feature would notably improve the application.
3. Low priority means that the application serves its purpose fine without the requirement's inclusion, but it would be nice to have.

ID	Requirement	Priority
FR1	The user must be able to navigate the application with touch gestures.	High

FR2	The user must be able to join a game match online with another user.	High
FR3	The user should be able to choose the player class they wish to play with before a game match starts.	Medium
FR4	A match should occur between two players, each player taking turns in an alternating order.	High
FR4.1	A player’s turn should automatically end after a defined amount of time.	Medium
FR4.2	A player should be able to end their turn manually.	Medium
FR5	A player should have health, armor and mana, as well as cards in their hand.	High
FR6	The user should be able to view their and the opponent’s available resources (health, armor, mana) when in a match.	High
FR7	The game should deal cards to each player from their decks at the beginning of the game.	High
FR8	A player should only be dealt cards from their chosen class’s deck.	High
FR9	During their turn, a player should be able to play all of the cards in their hand.	High
FR9.1	When a card is played during a game, the system should remove the card from the player's hand and add it to a “played card” pile.	High
FR9.2	When a card is played during a game, the system should modify the player’s health, defense and/or mana according to the card’s effects.	Medium
FR9.3	When a card is played during a game, the system should modify the opponent’s health, defense and/or mana according to the card’s effects.	High
FR9.4	A player should only be able to play a card if it doesn’t reduce their mana below 0.	High

FR10	The system should fill up the upcoming player's hand to the maximum allowed amount of cards from their deck at the beginning of their turn.	High
FR11	If the player's deck runs out of cards, the system should re-shuffle the player's played card pile and repurpose it as a drawing pile.	Medium
FR12	The system should refill the player's mana by a certain amount of points at the beginning of their turn.	High
FR13	If the player has no more health, they should lose the game.	High
FR14	The user should be able to turn on/off sound and/or music.	Low
FR15	The user should be able to forfeit the game.	Medium
FR16	The user should be able to view their game history and the outcome.	Medium
FR17	The user should be able to initiate a tutorial game as many times as they wish, from the main menu.	Medium
FR18	The user should be able to enter and exit a lobby created by other users, or create and delete their own lobby.	High

Table 1: Functional Requirements

3. Quality Requirements

Our primary software quality attribute is modifiability, as required by the project description. We have chosen usability as our secondary attribute, as we believe it is an essential quality of game applications.

3.1 Modifiability

M1 Add new Effect

ID	M1
Source	Developer
Stimulus	Add a new card effects to the game
Artifact	Game Model Module
Environment	Design Time
Response	New effects are usable for addition to cards and applicable in-game.
Response Measure	Within 30 minutes, without affecting any existing entities.

M2 Modify Effect

ID	M2
Source	Developer
Stimulus	Change the attributes of an existing effect in the game
Artifact	Game Model Module
Environment	Design Time
Response	The modified effect is applicable as before, with the updated attributes.
Response Measure	Within 10 minutes, without affecting any existing entities or gameplay logic.

M3 Add new Card

ID	M3
Source	Developer
Stimulus	Add a new Card to the game
Artifact	Game Model Module
Environment	Design Time
Response	New card is playable in-game.
Response Measure	Within 30 minutes, without affecting any existing entities.

M4 Modify Card

ID	M4
Source	Developer
Stimulus	Change the properties of an existing card in the game
Artifact	Game logic module
Environment	Design Time
Response	The modified card is playable in-game.
Response Measure	Within 15 minutes, without affecting any existing entities or gameplay logic.

M5 Add new Deck

ID	M5
Source	Developer
Stimulus	Add a new Deck of Cards to the game
Artifact	Game Model Module
Environment	Design Time
Response	New deck is available in-game.
Response Measure	Within 45 minutes, without affecting any existing entities.

M6 Modify Deck

ID	M6
Source	Developer
Stimulus	Change the content of an existing Deck
Artifact	Game logic module
Environment	Design Time
Response	The modified deck is available in-game.
Response Measure	Within 20 minutes, without affecting any existing entities or gameplay logic.

M7 Add new PlayerClass

ID	M7

Source	Developer
Stimulus	Add a new Player Class to the game
Artifact	Game Model Module
Environment	Design Time
Response	New PlayerClass is available for selection in-game.
Response Measure	Within 1 hour, without affecting any existing entities.

M8 Modify PlayerClass

ID	M8
Source	Developer
Stimulus	Change the attributes of an existing PlayerClass
Artifact	Game logic module
Environment	Design Time
Response	The modified PlayerClass is available in-game.
Response Measure	Within 30 minutes, without affecting any existing entities or gameplay logic.

M9 Modify Game Views

ID	M9
Source	Developer
Stimulus	Modify the appearance of a game entity

Artifact	Views Module
Environment	Design Time
Response	The modified view appears in-game.
Response Measure	In 1 hour, without affecting other parts of the system.

3.2 Usability

U1 Learn the Game

ID	U1
Source	User
Stimulus	Play the tutorial for the first time
Artifact	System
Environment	Runtime
Response	User understands the game controls, mechanics and is able to play the game as intended after a short tutorial.
Response Measure	Within 10 minutes the tutorial is concluded and the user has an understanding of the game controls and mechanics.

U2 Enter a Game

ID	U2
Source	User
Stimulus	Enter a game
Artifact	User Interface

Environment	Runtime
Response	User navigates through the menu/lobbies and enters a game match
Response Measure	Within 2 minutes the user should be able to be in a game match

U3 Modify Game Settings

ID	U3
Source	User
Stimulus	Modify a game setting (e.g. Toggling the music ON/OFF)
Artifact	User Interface
Environment	Runtime
Response	The user finds the game setting and is able to make a change.
Response Measure	Within 10 seconds, the user has toggled the setting they wish and the change has been applied.

4. COTS Components and Technical Constraints

4.1 Android Device

The game will be developed for Android devices. Android is an operating system mainly used for smartphones and tablets. These types of devices can have many different characteristics, but the most constraining one is the Android OS version. Android OS is backwards compatible which means we only need to define a minimum Android version that we wish to support and every later version will support our application. Our development will support a minimum Android SDK version of 24 (Android 7.0), which according to APILevels [2], allows for compatibility with approximately 97% of Android devices.

4.2 LibGDX

LibGDX is a powerful, open-source game development framework that provides a comprehensive set of APIs to facilitate the development of cross-platform games including Android. Despite its appeal, it imposes a few constraints on how the application is developed. One of the main things to be noted is the necessity for careful manual disposal of assets to avoid memory leaks. Additionally, its stack-based screen model may be restrictive in multithreading contexts. Finally, it has a logarithmic learning curve which comes with the cost of flexibility in more sophisticated scenarios in graphical processing.

4.3 Firebase

Firebase is a platform currently maintained by Google that offers platforms and backend as a service (PaaS, BaaS) for application development. Firebase is extremely flexible, offering scalability and real-time synchronization at very low or no cost. It additionally supports a plethora of services useful for development, such as authentication and real-time NoSQL databases - not restricting you to the Google ecosystem. According to [3], Firebase imposes constraints on the structuring of data with support only for JSON trees for its real-time database. Firebase will be used for user authentication, synchronization of gameplay and as a data store for user data.

5. Changes

Dates	Changes	Comment
March 3, 2024	Initial Document	N/A
March 24, 2024	Quality Attributes, Frontpage	Adjusted based on examiner feedback
March 25, 2024	Functional Requirements	Adjusted to match revisions/scope
April 16, 2024	Introduction and COTS	Updated for clarity and completeness

6. Contributions

Member	Part of the document	Approximate Hours
Adrien Rosset	Introduction and COTS	4h
Márk Somorjai	Functional and Quality Requirements	3.5h
Dionysios Rigatos	Introduction, Functional Requirements, Quality Requirements and COTS	3h
William Marleau Ken Nguyen William Hassel Dionysios Rigatos Márk Somorjai	Revision and finalization of the document	30 min

7. References

[1] Slay the Spire by Mega Crit Game on Steam, https://store.steampowered.com/app/646570/Slay_the_Spire/

[2] APILevels, https://apilevels.com/

[3] Firebase documentation for Android, Read and Write Data https://firebase.google.com/docs/database/android/read-and-write