Requirements and Analysis Document for KahIt

Working Group: Anas Alkoutli
Oussama Anadani
Mats Cedervall
Johan Ek
Jakob Ewerstrand

11 oktober 2019

1 Introduction

KahIt is a fun and entertaining quiz-app for everyone that can elevate a party to the next level. *KahIt* is an app designed to make parties more fun by making people play and interact with each other. With this app the user will be able to play with friends by either hosting a game or joining someone else's game.

Give some background and explain the purpose of this application. Describe the functionality of the application. Describe the stakeholders of the project, highlight who will benefit from/use this particular application.

1.1 Definitions, acronyms and abbreviations

todo

Create a word list to avoid confusion and give a definition of every abbreviation you use in the document.

2 Requirements

2.1 User Stories

Story identifier: sFQuestionl
Story name: The first question
Status: Completed

As a: user I want to: answer a multiple choice question because answering questions is fun.

Functional Confirmation:

- A questions is provided to the user and the user can answer it.
- The user can choose between a number (4) of possible answers.
- The user is notified if their answer is correct

Non-funcional Confirmation:

- All of the question is in view no matter the size of it.

Story identifier: sMQuestions
Story name: More than one question
Status: Completed

As a: user I want to: be able to answer more than one question because only one question is not enough for the game to be fun.

Functional Confirmation:

- After answering a question a new one should be provided.
- The new question should not be a question that has alredy been answered.

Non-funcional Confirmation:

- TODO?

Story identifier: sHGame
Story name: Host a game
Status: Uncompleted

As a: user I want to: be able to host a game so that my friends can join.

Functional Confirmation:

- The host does not have to provide any extra info except a possible password to setup a game.
- The host can remove players from his "room".

Non-funcional Confirmation:

- hosting a game should be low effort.

Story identifier: sJGame
Story name: Joining a game
Status: Uncompleted

As a: user I want to: be able to join a game that someone else is hosting because playing with someone is more fun that playing alone.

Functional Confirmation:

- The app searches for a "room" and when it finds a game it populates a list for the user to pick.
- The user is only connected if they pick one of the entrys in the list.
- The user can leave the "room" and terminate the connection at any point.
- When the user is prompted they are given some sort of identifying info about the game/room they are about to join.

Non-funcional Confirmation:

- Joining a "room" should require very few steps.

Story identifier: sHotSwap
Story name: Hot swap
Status: Uncompleted

As a parent I want all of my children to be able to play on the same device because it's too expensive to get all of them their own phones.

Functional Confirmation:

- The game has a "Hot swap" mode where you can select the amount of players at the start and subsequently play on one device.
- The game adapts the different game modes to the single device limitation.
- All local players are given the same question and when all of them as answered the correct answer is shown.

Non-funcional Confirmation:

- TODO

Story identifier: sChooseWhenJoiningGame
Story name: Choose when joining a game
Status: Uncompleted

As a: user I want to: be able to choose which game I am joining instead of just joining a random game.

Functional Confirmation:

- The user will be able to see a list of available games.
- The user can choose from the list to join the right game.

Non-funcional Confirmation:

- TODO

Story identifier: sStartGame
Story name: Start a multiplayer game
Status: Uncompleted

As a: host I want to: be able to start the game because that is why I am hosting it in the first place.

Functional Confirmation:

- When the user starts the game it should start for all connected users.
- The game should not start until the host has chosen to start it.
- The game should show the same questions to all users.
- The game should not move on from a question until all users have provided an answer or the time has run out.
- The alternatives for a question should be displayed in the same order for all connected users.
- If a user disconects from the game, the game should continue unless the player that left was the host. If the host leaves the game all other user should be disconnected.

Non-funcional Confirmation:

- All delays and slowdown should be avoided.

Story identifier: sUsingPowerUps
Story name: Using power ups.
Status: Uncompleted

As a: user I want to: Be able to use powerups and bonuses on questions if i'm sure i know the answer since I like games that are tactical.

Functional Confirmation:

- A player will be able to buy power ups with points earned through answering correctly.
- A power will last for a limited period only ex. three question rounds.
- A player can only buy a certain power up once.

Non-funcional Confirmation:

- TODO

Story identifier: sLeaderBoards
Story name: Leader boards
Status: Uncompleted

As a: user I want to: continually get an update on whom is in the lead since It can be hard to keep the score in my head.

Functional Confirmation:

- After each question the players will be able to see a list of the players with their amount of points and their placement.

Non-funcional Confirmation:

- TODO

Story identifier: sLottery
Story name: Lottery
Status: Uncompleted

As a player i want to have power ups and bonuses that reward good players since it makes the game more interesting and competitive.

Functional Confirmation:

- When answering correctly a player will have more chance to win power ups which will protect the players placement and take it even higher increasing the points earned per question.
- A working lottery that distributes winnings to all players.

Non-funcional Confirmation:

- TODO

Story identifier: sCosmetcis
Story name: Cosmetcis
Status: Uncompleted

As a: User I want: to be able to buy cosmetics in game since it makes the game cooler.

Functional Confirmation:

- The user will be able to buy cosmetics such as icons or special colors to stick out when being high on the leaderboard.
- The cosmetic items do not give any advantage in game but it adds character and customization to the game.

Non-funcional Confirmation:

- TODO

Story identifier: sLotteryDraw
Story name: Lottery draw
Status: Completed

As a: player I want: to get some randomized items from the lottery.

Functional Confirmation:

- All the players will be able to get a random item each time the lottery starts.

Non-funcional Confirmation:

- TODO

Story identifier: template
Story name: A template
Status: Completed

As a: x i want: y because of z

Functional Confirmation:

- a functional req

Non-funcional Confirmation:

- a non-functional req

Story identifier: template
Story name: A template
Status: Completed

As a: x i want: y because of z

Functional Confirmation:

- a functional req

Non-funcional Confirmation:

- a non-functional req

Use the template from the course website and list all user stories here. It is fine to have them in an spreadsheet (or other applications, such as Trello) at first, but they must end up here as well.These user stories should describe what the user will be able to do. Write the user stories in language of the customer, and give them a unique ID. List the user stories in order of priority.You need to annotate an user story whether or not it is implemented. We need to know which user stories are implemented, such that we can check this during the oral presentation.

2.2 Definition of Done

For a user story or a feature to be considered to be done there is a set of requirements that it needs to pass. Some of these are specific to the story/feature while others are more general. This section outlines the requirements that all stories/features have to fulfill to be considered done. in other words the definition of done.

- All code clears all of it's tests and is 100% covered.
- All code has relevant comments explaining it's function and purpose.
- All public classes are documented with Java Doc.
- All comments and documentation is written in English.
- All code has passed a code review to guarantee a uniform style and level of quality.

In this section you list the acceptance criteria that are common for all user stories. For example, the code should reviewed and tests, it should be under version control, etc.

2.3 User interface

todo

Include sketches, drawings and explanations of the application's user interface. Describe the navigation between the different views.

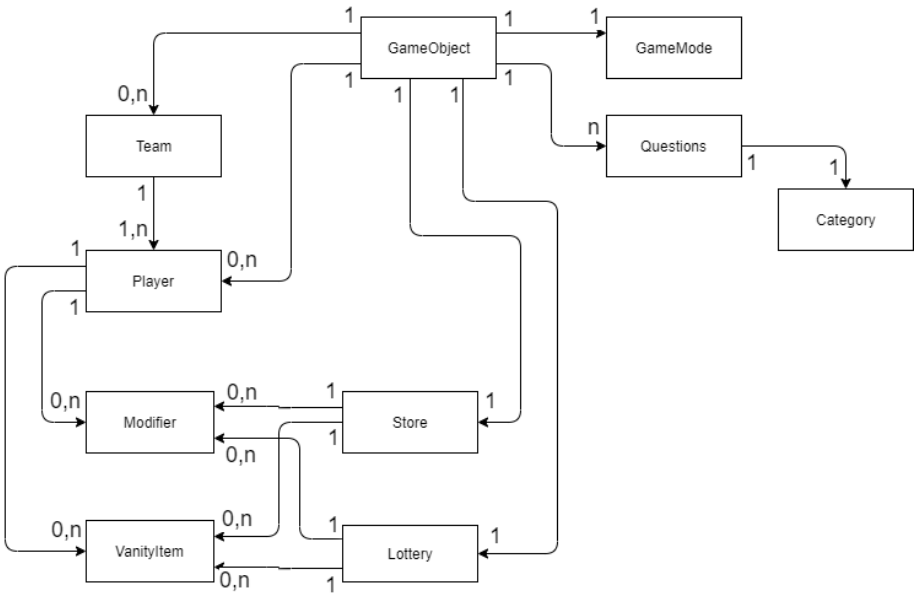

Figure 1: Domain Model

3 Domain model

Give a high level view overview of the application using a UML diagram.

3.1 Class responsibilities

3.1.1 QuizGame

The class QuizGame acts as the aggregate object of the model, in other words the QuizGame is the access point through which outer parts of the application communicates with the model.

3.1.2 Questions

The Question class is responsible for holding all information that is relevant to a question. E.g actual question and possible answers. The class is also responsible for checking if a given answer is correct.

3.1.3 Category

Category is an enumerator that is responsible for representing the different categories that a question can be a part of. The enumerator is also responsible for a number relevant behaviour such as returning a category based on a text string or an index.

3.1.4 Store

The Store class is responsible for holding items and making it possible for players to buy items that can affect their score during gameplay. This class holds values such as which items that are available in the store. This class also holds the necessary logic to complete a transaction.

3.1.5 Lottery

The Lottery class is responsible for managing the lottery prize giveaway system.

3.1.6 Modifier

The modifier class is responsible for the items that can alternate the stats of a player. A modifier holds all the values which determines if the stats of a player will be affected positively or negatively. A modifier can affect the stats by the score of a question or the time taken to answer.

3.1.7 VanityItem

The VanityItem class is responsible for the cosmetic items in the game. This class holds values such as the price and the duration of an item as in how many rounds this item can last.

3.1.8 Player

The Player class is responsible for holding all player specific information. Attributes such as current score, player name, id, modifier stats and owned vanity items.

3.1.9 Team

Team class is responsible for keeping track of players in a specific team, team combined score and team name.

Explanation of responsibilities of classes in diagram.

4 References

https://developers.google.com/nearby/connections/overview

List all references to external tools, platforms, libraries, papers, etc.