Cards Against Society
Milestone 1

SW Engineering CSC648/848 Spring 2019

Section 4

Team 203

03 October 2019

Team Lead: Jose Castanon

Back-End Lead: Leslie Zhou

Database Master: Shota Ebikawa

Git Master: Daryl Ortiz

Front-End Lead: Brian Le

Revision #	**Date**	**Contributor**	**Summary of Changes**
1	10/2/19	Jose Ortiz	Fixed executive summary to explain the goal of the game. Added summary and extra feature to competitive analysis.

Table of Contents

1. Executive Summary

Cards Against Society is an online real time game based on the card game Cards Against Humanity – a game which is most fun when played with friends. In a typical game of CAH, the goal is to reach a certain amount of points to win, one player will choose a black card, which contains a question or blank space. The other players will draw white cards to their hand and then select one of them which contains a response that they think is the funniest response to the black card. After all the players have picked a response, the black card holder will read the prompt on the black card along with each white card to fill the blank space. The black card holder will then pick the response they think is the funniest, and the player who picked that response will gain a point. The game continues like so, and the winner is the person who reaches a set amount of points – usually around five points.

The purpose of this project is to provide the user the same fun experience that they would get if they were playing the game in real life, while allowing them to interact with other people online. Cards Against Society will aim to replicate the experience of playing Cards Against Humanity in real life virtually, while also allowing the users to play the game as normal. They will also be able to communicate in real time with other players and create public or private games. Cards Against Society, unlike other online clones, will focus on simplicity and the user experience which will allow newcomers and returning players to jump into the game with ease.

2. Main Use Cases

Actors

John - Host / General User

- College CS student
- Loves playing Cards Against Humanity
- Likes playing with his friends
- Recently moved off to college, misses playing CAH with his friends

Paul - New User

- Has not played Cards Against Humanity before
- Wants to play with his friend who lives far away

Karen - General user

- Plays Cards Against Humanity online for fun

Use Cases

John:

John, a college student, loves playing Cards Against Humanity with his friends. Limited by the distance between him and his friends back home, he turns to Cards Against Society to be able to play with them. Upon entering the site, he is prompted to register or log in before continuing.

John creates an account and logs on, and is now able to see the main lobby, along with all the ongoing public games and the public chat. He creates a new lobby titled “John’s room” so that his friend can locate his game later on.

Inside the game, John sees his friends join. He sends a message inside the rooms private chat saying, “What’s up!”. During the first round of the game, John is the black card holder. He waits for the other people in the lobby to pick a card, and can see when they pick a card. He then proceeds to pick his favorite white card and gives a black card to the funniest player.

During the game, John’s laptop dies. He connects his charger and logs back into the site. In John’s “My Games” page, he can see all the ongoing games that he is in. He finds his last game and joins it, and can continue playing as before.

Paul:

Paul’s friend group is too tired to go outside tonight and all decide to stay home and play computer games instead. His friends decide to play Cards Against Humanity, but Paul has never played it before. He is sent a link to play Cards Against Humanity from his friend and he is prompted to either log in into an existing account, or create a new one. Paul then creates an account by adding an email, password, and selecting a username which all players can see.

He is then prompted if he has ever played Cards Against Humanity before, and he selects no. He is given a picture of basic rules and instructions on how to play the game.

Upon completion, he is redirected to the locked (private) room which was created and hosted by his friend that sent the link. Once connected into the room, he sees the usernames of his four friends displayed on top of the chat box with the scoreboard values set to zero next to them.

He then uses the chat box to announce he is connected into the room, and the host commences the game by giving him five randomly selected cards.

Karen:

Karen is bored and wants to play Cards Against Humanity online. She logs in into her existing account and browses the homepage for open and non-filled room with a funny name but all the rooms with funny names are filled with the maximum amount of players.

Karen decides to host a new room with the name “Funky Monkey” with the settings set to have eight maximum players in the lobby, requires five points to win, and room unlocked to allow random users to join at any time.

Once created, a few players instantly join in and Karen has the option to kick a player via button next to their usernames. Next to the chat button, she can access the room settings to edit player limit, point requirement, and whether the room is accessible to the public.

3. List of Main Data Items and Entities

Entities

- User
    - Host
        - Creates a public or private room, sets game rules
    - Regular user
        - Joins any game
- Rooms / Lobbies

Data items

- User Database
- Current Games Database
- Card Database

4. Initial List of Functional Requirements

- Users shall be able to register an account
- Users shall be able to log into an account
- Users shall be able to create game lobbies
- Users shall be able to join game lobbies
- Users shall be able to find their current games
- Users shall be able to come back to a game at any time
- Users shall be able to send messages through the chat
- Users shall be able to participate in any amount of games concurrently.
- Users shall be able to go back and forth between each game

5. List of non-Functional Requirements

- System shall update game data in real time
- System shall update chat message data in real time
- Game states shall be persistent, saved by the system and loaded when a player joins
- In a game, players will be shown relevant game data such as which player is the black card holder or the number of black cards each player has.
- Messages shall be seen only by the users in that lobby

6. Competitive Analysis

	Pretend You're Xyzzy	Pictures Against Humanity	Cards Against Society
User Interface	+	+	++
Custom Cards	-	+	+
In-game Chat	+	+	++
Private Room	+	+	+
User Auth	+	+	++
Tutorial	-	-	+

In comparison to our competitors, we plan to implement a more welcoming and exciting user interface. Our competitors offer similar features to each other, and we plan to implement those features. However, we will add a tutorial for those who are not familiar with Cards Against Humanity, which will make it easy for newcomers to start playing.

7. High-Level System Architecture and Technologies used

Web Server

- Host: Amazon Web Services 1vCPU 1GB RAM
- OS: Ubuntu Server 16.04

Technologies

- Server side language: Javascript
- Front end: React
- Backend: Node.js, Express.js, Socket.io
- Database: MySQL 8.0.17

Browers:

- Mozilla Firefox: (ver. 67, ver. 68, ver. 69)
- Google Chrome: (ver. 75, ver. 76, ver. 77)
- Safari: (ver. 10, ver. 11, ver. 12)
- Edge: (ver. 41, ver. 42, ver. 44)

8. Team

- Team Lead: Jose Castanon
- Back-End Lead: Leslie Zhou
- Database Master: Shota Ebikawa
- Git Master: Daryl Ortiz
- Front-End Lead: Brian Le

9. Checklist

- Team found a time slot to meet outside of class - **ON TRACK**
- Github master chosen - **DONE**
- Team decided and agreed together on using the listed SW tools and deployment server - **DONE**
- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing - **DONE**
- Team lead ensured that all team members read the final M1 and agree/ understand it before submission - **DONE**
- Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.) - **DONE**