

Cards Against Society

Milestone 4

SW Engineering CSC648/848 Fall 2019

Section 4

Team 203

05 December 2019

Team Lead: Jose Castanon – jcastan6@mail.sfsu.edu

Back-End Lead: Leslie Zhou – zzhou2@mail.sfsu.edu

Database Master: Shota Ebikawa – sebikawa@mail.sfsu.edu

Git Master: Daryl Ortiz – dortiz4@mail.sfsu.edu

Front-End Lead: Brian Le – ble2@mail.sfsu.edu

<u>Revision #</u>	<u>Version #</u>	<u>Date</u>
Milestone 4	Version 1	12/05/19
Milestone 3	Version 1	11/21/19
Milestone 2	Version 2	11/07/19
Milestone 2	Version 1	10/24/19
Milestone 1	Version 2	10/09/19
Milestone 1	Version 1	10/03/19

Table of contents

1. Product Summary	2
2. Usability Test Plan	5
3. QA Test Plan	8
4. Code Review	
5. Self-Check: Best Practices for Security	
6. Self-Check: Adherence to Original Non-Functional Specs	

1. Product Summary

Have you ever come home from a long day out, taken off your Heelys and your bejeweled, boot cut jeans, turned on some Big Time Rush, and sat in your lonely room in nothing but your socks staring at your cardboard cutout of Dolly Parton, contemplating whether to go back into the cold, miserable world in order to meet up with your fake friends for card game night? If so, then Cards Against Society is the game for you!

Cards Against Society is a highly distasteful, strategic game of ad libs, where you pick a card based on it's offensiveness to the ad lib. Players will turns drawing an ad lib card and select the most messed-up/funniest statement to complete that ad lib.

Cards Against Society combines the millennial trend of table-top games with the convenience that technology offers in the modern age. Now you can enjoy the company of your friends from the comfort of your own home. No more feeling guilty for licking your Cheeto crusted finger tips when hanging out with your friends, or feeling that you're missing out on quality time and memories while sitting at home staring at your computer screen in the dark!

Compared to other online Cards Against Humanity clones, Cards Against Society allows you to play multiple matches concurrently to play with two different friend groups. Also, if you wish to create your own drama and make your friends question your friendship, you have the ability to create custom cards.

Functions :

- Unregistered Users
 - Shall be able to create a new account
 - Shall be able to view Terms of Service
- Registered Users
 - Shall be able to log into their account
 - Shall be able to log out of their account
 - Shall be able to view all active games
 - Shall be able to join games
 - Shall be able to create new games
 - Shall be able to come back to a game at any time
 - Shall be able to create custom cards
- Hosts
 - Shall be able to set game rules on room creation
 - Shall be able to start a new round
- Lobby
 - Shall display all open games and its Room Name and the number of players in the room
 - Shall display public chat room
- "My Games" Page
 - Shall display all current active games for the logged in user

- Game Rooms
 - Shall display user's cards
 - Shall display a private room chat
 - Shall display timer for player to submit a play
 - Shall show relevant game information to the player
- Admin
 - Shall be able to modify or remove data entries
 - Shall be able to stop on-going games
 - Shall be able to view all pages
 - Shall be able to delete users
 - Shall be able to see a list of all registered users
 - Shall be able to see a list of all on-going games
 - Shall be able to have access to admin navigation bar

URL = <http://52.53.156.79:4000/>

2. Usability Test Plan

User satisfaction is the goal for the application and the game room is where it is expected to deliver that. It is important that the user is satisfied with all components of the game room and is having fun. The Game Room is the most important page in the whole application, reason being is the whole purpose of Cards Against Society is to be enjoyable for the user to satisfy user needs. This section is critical and has the most impact on the whole site as it contains usability from other components such as the chat feature and the game itself. To measure user satisfaction is to look into usability metrics. The usability metric will measure effectiveness, efficiency, and user satisfaction.

The first usability metric is Effectiveness where we want to see if users believe the game room system is useful, easy to learn, and that it does what the user wants. The goal for the user is to play the game, use the components fluidly, completely and ultimately find enjoyment. Using the chat feature and other components while playing the game will contribute towards user goals and user satisfaction. We want to see how complete the game room is and if the user interface is visually appealing. From here we can test other usability metrics, efficiency and satisfaction.

To test for efficiency, we can look into efficiency in time, effort and design. How long it takes for the components of the game to proceed will impact how long the user has to wait. The chatbox should instantly receive and send messages. The UI of the application should be easy to use and visually appealing.

Finally we can test the most important goal on how satisfied the user is. For all the components of the game room, we want to measure how comfortable the user is engaging with the different functions in the room. The usability test plan wants to find out if all the components are simple, easy to use and are understandable for the user. The results of the effectiveness, efficiency and user satisfaction will allow for usability to be measured.

To access the application, the users can use various operating systems and browsers. The application can be run on Google Chrome version 77.0.3865.90 on Windows 10 version 10.0.15063 Home Build 15063 or on macOS 10.12 Sierra and above.

To get to the Game Room, the user must first create an account. They will then log in to the lobby using the information they used to register. From the lobby they can either join a game or join from the list of games or create a room themselves.

Cards Against Society is for ages 17 and above. With cards that touch on sexual conduct, racial issues, humanitarian crises, inappropriate statements surrounding society and a database full of other taboo topics, this game is best played by people young adults and above.

The URL of the system to be tested is <http://52.53.156.79:4000>. What will be tested are the components of the game page. User goals in which user goals are being met. How long will the user wait between input and output of the chatbox. How appealing is the UI and if it was easy to understand, use and if the user had fun using the application.

Users will be tasked to test out different functions of the game room such as the chat box and gameplay. Users will join a game room and play the game in order to enter the game room. For the chatroom, the user will type and send messages in the chat box. They will also switch between tabs to send messages privately or publicly. For the gameplay, the user will play the game by reading the black card prompt and submitting a whitecard(s). Users will be asked about the chat box, how easy it was to navigate and if the user had fun.

Questionnaire

Cards Against Society - User Satisfaction

1. The chat box works perfectly.

Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Select	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

2. The game was easy to play and understand.

Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Select	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

3. I had fun playing the game.

Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Select	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

3. QA Test Plan

There are many different kinds of bugs and they can occur for a number of reasons. Some bugs are easy to diagnose and fix while others are hard to detect. The Game Room will be tested for Quality Assurance as it is the most important function in the application. Most if not all of the potential bugs will come from the Game Room. It is the area where the most components and implementation reside.

There will be three tests for Software Quality Assurance, OS compatibility, pulling information from the database and game functionality. The tests will have specific input and a specific result that will dictate whether or not the test will pass or fail.

To test whether the application will work on different operating systems, the user will be using specific versions of Windows and Mac. For the Hardware, the user will be using a Macbook running macOS 10.15 and another laptop using Windows 10 10.0.15063. This will test whether the application will be consistent and usable on different platforms.

Another test is to test whether or not the database, MySQL was used to store data. To see if the data is successfully being retrieved from the database will signify whether or not the database is working correctly. For the test, the user will enter the game room to conduct observations on whether data from the MySQL server is being displayed on the game cards. Upon success, the cards will display data that was retrieved.

The last test will test functionality with gameplay. The user will test whether or not there are complications within the game room. Some examples of this are components not loading, some aspects of the pages are freezing, or functionality is not working as expected. When the user enters the game room, the user should be able to select a card and submit it.

Being the page with the most work done on, there is a more likely chance to discover a bug. Bugs contribute to user satisfaction on whether they enjoy the feature of an application or not. If there are any bugs that can be found anywhere, then it is best to notified of their existence so they can be addressed as soon as possible. The Game Room contains the chatbox, player list, the navbar and the game itself, all of which can contain bugs. If a bug is found in the game room, then it is likely to be a component found on another part of the application. Finding and correcting bugs will open an opportunity to correct not only the game page itself, but other pages as well.

Quality Assurance Test Plan

Test #	Test Description	Test Input	Expected	Results
1	User tests out operating system compatibility.	The user uses operating system Windows 10 10.0.15063 and MacOS 10.15 to run the application in the Game Room	The application is the same on different operating systems.	Pass
2	Checking if all cards are being displayed to the user	User entering game room	The user is able to view both white and black cards in the game	Pass
3	Testing whether the user can select and submit a card	The user chooses a card and clicks on the submit button	Card is consumed and submitted	Pass

4. Code Review

a) Coding Style

- Encapsulate inline-styles in a single variable
- Start a new line after the opening curly brackets when initializing objects and functions
- No whitespace between JSX/HTML tag's angle brackets and its element name
- Use arrow function for the on-event handlers
- Have each objects in a separate line when importing more than 4 objects from a given module

Naming principles:

- Pascal case for Classes (React Component) and JSX file
- Camel case for variables and functions
- Lowercase for json file and image file

b) Code Reviews

Jose's Code

```

/*
  Daryl Ortiz
  - Commits are perfectly placed and simply and clearly explains the implementation.
  - Everything is structured well. much thought was put into the code.
  - The structure of the game container follows exactly like the mockups.
  - Overall everything about the page contains no issues to me and is very well done.

  Leslie Zhou Team 3
  Comment:
  Great code in general
  the styling of declaring function is different from other files

  Brian Le
  - Holy long batman!
  - Great variable + function naming
  - imports could be categorized / grouped
  - Good comments on functions
  - Easy to follow indentation

  Shota Ebikawa Team 3
  Comment:
  The imports are indented in organized fashion, especially the one with the
  react-bootstrap. I also like how you added few snippets of comments to the
  functions that deals with a complicated logic, such as submitSelection.
  One thing I would do is to add a bit more comment to the gameLayout
  function to help team members understand its logic more easily.
*/

import React, { Component } from "react";
import io from "socket.io-client";
import {
  Button,
  ButtonGroup,
  Row,
  Col,
  Container,

```

Daryl's code:

```

/*
Shota Ebikawa Team 3
Comment:
The JSX tags are indented in a consistent manner. Overall, I had an easy
time understanding each blocks of codes. One thing I would do is to delete
unused components from react-bootstrap, such as the Button component. Other
than that, there are not much to say from my end.

Jose Castanon
Indentation is not correctly formatted. Page could be better organized. Consider using react Row and Col to better
structure the page.

Leslie Zhou Team 3
Comment:
Great page that does the job

Brian Le
- Personally I would reorder the imports to categorize
- Overall, everything is nicely segmented and categorized for easy reading
*/

import React, { Component } from 'react';
import Header from './Components/Header';
import { retrieveCookie } from './Components/Cookies';
import './app.css';
import ListRooms from './Components/ListRooms';
import UserSearch from './Components/UserSearch';
import { Container, Row, Col, Button } from 'react-bootstrap';

export default class Admin extends Component {
  state = {
    userid: retrieveCookie('userid')
  };

  render() {
    return (
      <body>

```

Leslie's code:

```

/*
Brian Le
- Nicely categorized, but the 'react-bootstrap' imports could be one lined rather than 3
- Unused mount 'componentDidMount()'
- All functions in same area :ok_hand:
- HTML is nicely indented (:

Daryl Ortiz
- Everything is very well structured
- Although it is easy to understand by looking at the code itself, it would also help to have comments.
- Examples of the Custom Decks are all helpful to new users trying to figure out what kinds of custom cards to
make.

Jose Castanon
- Code is very well laid out
- Using a loop to find which deck was clicked is a bit inefficient, although there's not much happening in the
page so it's probably not going affect performance.

Shota Ebikawa
The code is formatted consistently, such as the JSX tags
and I was able to understand every blocks of code. One
thing I would do is to add few lines of comment for the
handleClick. By doing so, team members will have an easier
time understanding the given logic.
*/

import React, { Component } from 'react';
import { Redirect } from 'react-router-dom';

```

Shota's Code:

```

Leslie Zhou Team 3
Comment:
marginStyle is empty
not resetting the form after submit
*/

/*
Daryl Ortiz
- Although everything is understand, comments would make everything more clear.
- Everything is indented nicely.
- My only suggestion is to use containers so that the game room doesn't overlap between other modules.
*/


/*
Brian Le
- First glance, everything nice and uniform
- Imports can be categorized by 'from's and 'css'
- Nice and neat coding, easy to read
- Is there anyway to not start with a fragmentation?

Jose Castanon
- I would suggest using a class to better manage states.
- Good formatting, very organized
- Maybe use a css file for styling
*/

import React from "react";
import "../About.css";
import {
  Button,
  FormGroup,
  FormControl,
  FormLabel,
  FormCheck,
  Dropdown,
  Modal
} from "react-bootstrap";

```

External Review from Team 5

 **Amir A** 4:25 PM
jose.js ▾

```

1  const express = require('express');
2  const Sequelize = require('sequelize');
3  const bodyParser = require('body-parser');
4
5  const router = express.Router();
6  const models = require('../models');
7
8  const app = express();
9  const op = Sequelize.Op;
10
11  /**
12   * --Amir Team 5
13   * Overall Comment:
14   * I like the how the routes are organized by their usage and easy to find
15   * I would love to see some more comments inside the code to understand better what every route is for
16   * And lastly I see most of the routes do a lot of work as they being called. Wouldnt be better to split the work in different
   functions and just export
17   * the functions ? It would help alot with the organization
18   *
19   */
20
..

```

5. Self-Check: Best Practices for Security

Major assets we are protecting are our user's emails and passwords as they are needed to create an account and to access to an account. When a user is creating an account, the input for emails, usernames, and passwords are checked. Although not protected, the inputted username must be unique. Emails must contain an 'at' symbol (@), and end in a '.com'. Passwords are required to be at least eight characters long, and checked twice to see if the user input the password twice correctly. Upon creation, the user's password is then encrypted and stored in our database.

6. Self-Check: Adherence to Original Non-Functional Specs

Original Non-Functional Spec	Status
System shall update game data in real time	DONE
System shall update chat message data in real time using sockets	DONE
Game states shall be persistent, saved by the system and loaded when a player joins	DONE
Application shall be developed using the software stack declared in M0	DONE
Application shall be deployed using an AWS EC2 instance	DONE
Data shall be stored in a MySQL database	DONE

Application shall display game without the use of images	DONE
Application shall allow up to 5 users in one game room	DONE
Only relevant data shall be collected, such as user's win rate	DONE
System shall be compatible with at least two of the major browsers, versions listed above	DONE
UI shall be implemented using React	DONE
Game animations shall be implemented using React-Animations	DONE
Application shall load a default game state when a room is created	DONE
Application shall run off of the Master branch of the team's git repo	DONE
Database shall store all of the game cards to be used during the game	DONE
Application shall have a responsive game UI	DONE
Application shall keep public and private chat in their respective places	DONE