Group #5: The Programmers Who Say C

Members: Tyler Poff, Mike Shoots, Travis Brown,

February 15th, 2015

Project Proposal for COSC 340: Software Engineering

Fill In The _______!

A Multiplayer Card Game App for Mobile Devices

TABLE OF CONTENTS:

1. Summary:

Cards Against Humanity is a popular party card game first released in May of 2011. Within a week it was the number one selling board game on Amazon. It’s simple premise and fun gameplay enticed players and provided them with hours of entertainment. However it is not without its flaws. Its crude material is not appropriate for younger audiences and by its very nature as a physical card game means that it requires organization between people in order to set up a game. Our team believes that we can create a mobile app that encapsulates the fun of Cards Against Humanity and allow players to connect to others through a client-server relationship capturing the social aspect of the game while keeping it appropriate for a wide audience.

Our team will develop this app for Android powered devices using the Unity3D game engine as the main user Interface and a Python powered Server running the Flask technology to manage and connect players to each other. We will not only focus on crafting an app that only encompasses the base game however. We will seek to create a flexible app that allows users to customize how the game is played by changing the scoring system and win conditions. This will keep the game exciting and fresh even for veteran players.

Our team believes that we can provide a well-designed, social and mechanically flexible adaptation of Cards Against humanity that is appropriate for a wide audience.

2. Introduction:

On December 1st, 2010, a new campaign launched on Kickstarter by a small group of developers hoping to finance the creation and distribution of their tabletop card game, Cards Against Humanity. Within 2 weeks the campaigned reached its initial goal of $4,000.00, and by the campaigns conclusion on January 30th, 2011 had raised over $15,000 just under %400 of its original goal[1]. In May of 2011 Cards Against Humanity officially released and was quickly met with positive reviews and astounding sales, becoming the number 1 selling game on Amazon[1]. Critics gave glowing reviews for the game, praising its simple premise and humorous gameplay. However, despite this praise, the game was not without its criticisms. *Many noted the mature material was not suitable for a younger audience and although the game is fun, actually setting up a time and place for the game to take place is a challenging prospect in our mobile society. Our team's project seeks to create a mobile app that captures the fun of Cards Against humanity and make it accessible to players who would normally struggle to organize a physical game.

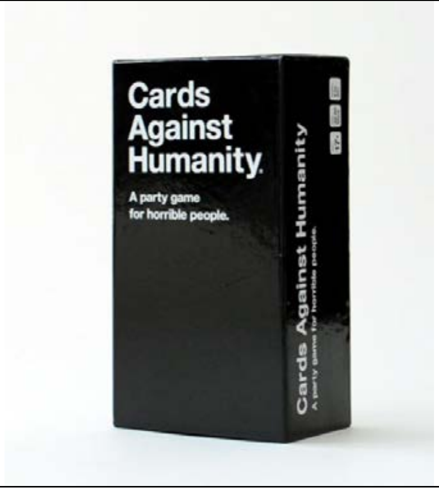

Fig 2.1: The Cards Against Humanity Game
Image Source:
https://cardsagainsthumanity.com/v8/images/mg_desktop-d6fe8468.jpg

3. Customer Values:

While developing this project our team will strive to make a product that appeals to a wide array of customers. From the fans of the original Cards Against Humanity, to younger audiences looking for a new game to play with their friends, we will seek to add attributes to our app that appeal to all audiences.

3.1 Customer Need:

At its core, Cards against Humanity is a very simple game. There are two types of cards: The black cards have fill-in-the-blank style sentences or questions and the white cards have objects, concepts or people that are used to answer the black cards. Every round one player is chosen as the card "czar" who draws a black card, the other players must then play a white card and the czar picks the white card he find the most amusing. The player who played the white card chosen by the czar gets a point for that round and the process repeats until a player reaches a predetermined amount of points[2]. The game's simple premise and opportunity for humor was well received by critics. Thrillist praised the game saying it's "the game your party deserves"[3].The game soon developed a large player base and soon became the number one bestselling board game on Amazon.com[1]. The game has become a commercial success with the Chicago Sun-Times estimating that Cards Against Humanity has earned at least $12 million in profit [4]. Players posting funny match ups between Black and White cards on social media also helped boost awareness of the game and sales. From a humble Kickstarter to a popular party game,

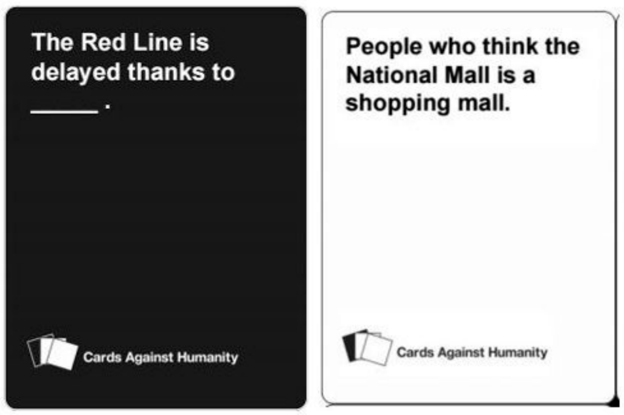
The Red Line is delayed thanks to ______ .
Cards Against Humanity
People who think the National Mall is a shopping mall.
Cards Against Humanity

Fig 3.1: An example of a Black Card and a White Card used in the game. The white card is meant to fill in the blank space in the black card.
Image Source: http://cdn.cstatic.net/images/gridfs/550b0bcff92ea10b540067f3/cards%201.jpg

Cards Against Humanity can only be described as a commercial success. However, despite its popularity the game is not without its flaws as noted by several reviewers.

The BoardGameGeek rated it a 6.48/10 stating the crude and politically incorrect content of the game was offensive to certain audiences[5]. Even the A.V. Club, which gave the game a positive review, was quoted as saying it was "a sort of Apples To Apples for the crass and jaded"[6] The material often makes use of events, people and organizations that when put in certain contexts may be offensive. In some cases the material in and of itself may be harmless enough but when paired with another cards takes on a meaning that is very insulting. Fig 3.2 and Fig 3.2 each show an example of potentially offensive cards. This offensive content may turn potential players away from the game or cause unnecessary conflict to arise while playing.

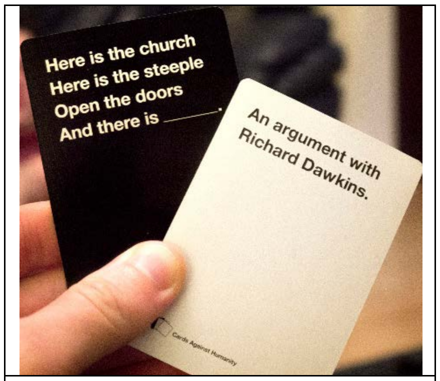

Fig 3.2: Although each of the above cards in this figure may seem relatively harmless, when paired together it may appear offensive to religious groups of people.
Image Source: https://upload.wikimedia.org/wikipedia/commons/4/4f/Cards_Against_Humanity_%2815711676205%29.jpg

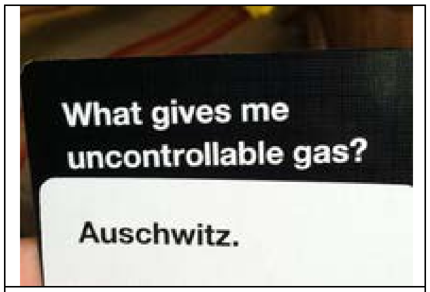

Fig 3.3: This is an example of the inappropriate content included in Cards Against Humanity.
Image Source: http://data.whicdn.com/images/150802574/large.jpg

The physical attributes of the game itself also limit when and where the game can be played. Although many people enjoyed the game despite its offensive content, in our fast paced society getting the time and people to properly play the game is quite a challenge. Organizing three or more players, finding a time and place to host the event, and ensuring the cards themselves are in a playable condition all detract from the gaming experience and potentially alienate customers.

Players need a quick and easily accessible way to play the game without the additional hassle of organizing an event or the potential uncomfortableness of an inappropriate card being played.

3.2 Proposed Solution:

In an age where mobile gaming is on the rise our team believes that this game is a prime candidate to receive a well-crafted and flexible mobile adaptation to allow players to enjoy the experience without all the additional hassle of organizing a physical event, without losing the social aspect of the game. With an app players can quickly hop into a match with other players quickly and easily. With the rise of mobile gaming over the past few years the time to develop this app is ideal. Fig 3.2 clearly shows an increasing trend in the rise of mobile gaming. With the popularity of smart phones it has never been easier for players to enjoy a game when they want and where they want.

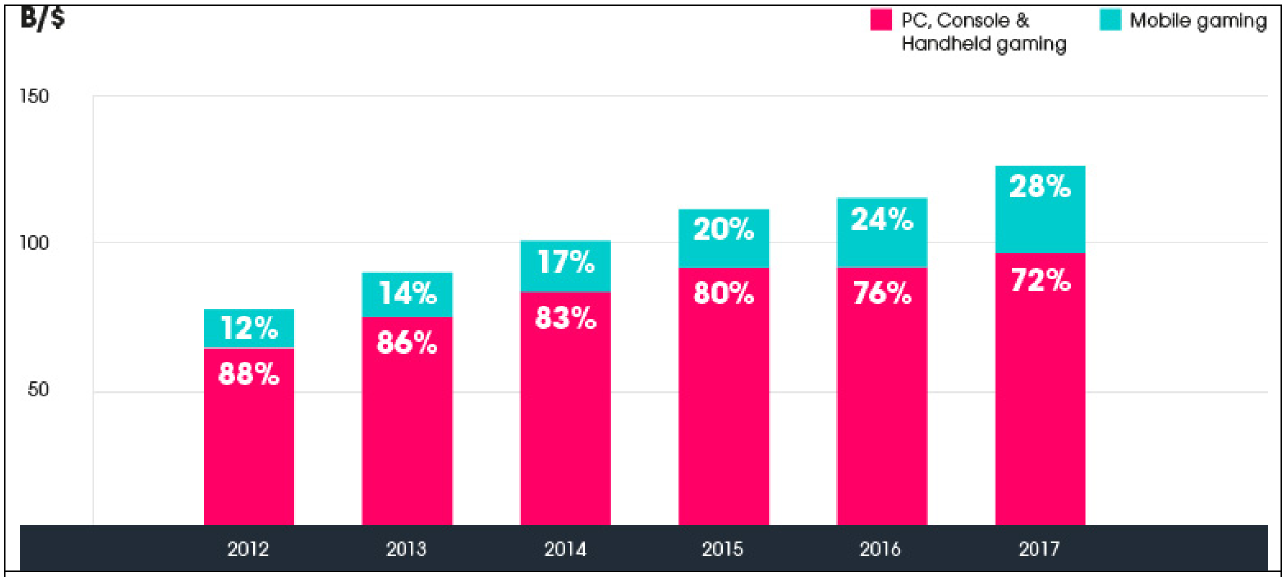

Fig 3.4: Based on the graph above from unity3d.com the market share of mobile games has been rapidly growing since 2012. This trend demonstrates the clear advantage of developing for mobile platforms as it is a growing market.
Image Source: https://unity3d.com/public-relations

While mobile adaptations of Cards Against Humanity exist, many are implemented poorly or contain mature content not appropriate to younger audiences.

These implementations of the game are also very limiting in ways for players to communicate. Of the products we have researched only one implemented a way for players to communicate and even in that case it was not user friendly solution. Not a single app attempted to differentiate the rules of the original as well. A variety of game options will ensure that a game remains fresh well after its initial launch. With a static rule set players can soon become bored and move on to other more varied apps.

To keep our game engaging for the customer we will implement a variety of game options that can give the player a more varied playing experience. Our team will implement these options into a variety of different game modes that the player can choose from. Currently our team is designing 3 separate games modes, each implementing a different scoring technique. The first mode is currently nicknamed Anarchy Mode. In this mode every player will vote for a card and each vote counts as a point for the player who played it. This mode is predicted to be very fast paced as it is possible to achieve a high score very quickly if a player chooses very popular cards. The second game mode is nicknamed Democracy. In this mode each player is allowed to vote, but only the card with the highest vote count at the end of the round get a point. The final game type is called Monarchy. This game mode closely resembles the original game rules from Cards against Humanity. One player is chosen at random to be the King at the start of each round. The king does not play a fill-in card and is not eligible for points in the round that he is king. However, the King is the only player allowed to vote on cards at the end of the round. For each round a new King will be chosen. Each player will get an opportunity to both be the King and distribute points, and be a regular player and earn points.

Each of the listed game modes will have several adjustable options that encompass all of them. These options include the maximum point threshold, and an adjustable move timer to ensure that each player chooses an action within a set time limit. With these game modes and options we hope to keep the player engaged with our app.

Our project will seek to create an app that captures the fun of Cards Against Humanity while easily connecting players with a game and ensuring that it's content is devoid of crude or offensive material. We will seek to provide a well-designed, social and mechanically flexible adaptation of Cards Against humanity that is appropriate for a wide audience.

3.3 Measures of Success:

In order to gauge how successful our product is we will rely entirely on customer feedback. A game can be the most technically sound piece of software ever developed but if the end user does not enjoy using it then the software is virtually useless. This is especially true of games. Unless the customer base enjoys playing our game then our project will be a failure. As such we will rely on feedback from testers and user reviews to gauge whether our game is a fun and enjoyable. If the players love our game, then our game will have served its purpose.

4. Technology:

In order to complete this project our team will have to use and implement a wide variety of technology. The basic components of our project are the Client and Server software. Each will present their own challenges and will require their own tools to run. The Client will not only have to manage a graphical user interface but it must also have an intuitive menu system with clean easy to use gameplay mechanics and it must also be able to communicate with the server software. The Server, on the other hand, must be able to handle multiple users playing multiple games reliably and effectively with minimal noticeable delay to the end user. We must also develop accurate test to ensure that each component is functioning correctly under various circumstances.

4.1 General Overview of Technology:

The project can be broken into two components; the client and the server. The client is the basic interface for which users will provide input to the game and will communicate to the server these inputs. The server will broadcast these choices to the other users and keep track of the current game state, such as when the players should play a card or vote for cards already played. The server will also track the current points for each player and will determine a winner based on the points the players have accumulated.

4.2 Client:

The client will serve as the interface for the player to play the game. The client will allow players to indicate what games they wish to join, what cards they wish to play and any other actions the player wish to carry out. It will consist of several classes all working together to allow the user to experience the game. It will have to handle user input and be able to send and receive messages from the server. The client will be a state machine; each state will have its own screen which will display the unique inputs for that state. We will use the Unity Engine to develop the graphical aspects of the client. Unity is a powerful tool used to create 3D and 2D games across a multitude of platforms with very little custom coding on the part of the developer. All of the nuances of developing for multiple platforms are handled by the Unity engine, allowing us to focus on the design of our game.

4.2.1 Client Classes:

The client software will implement a variety of classes, each with their own unique purpose that will help flesh out the game. Fig 4.2.1 shows a diagram of the class hierarchy for the Client Software.

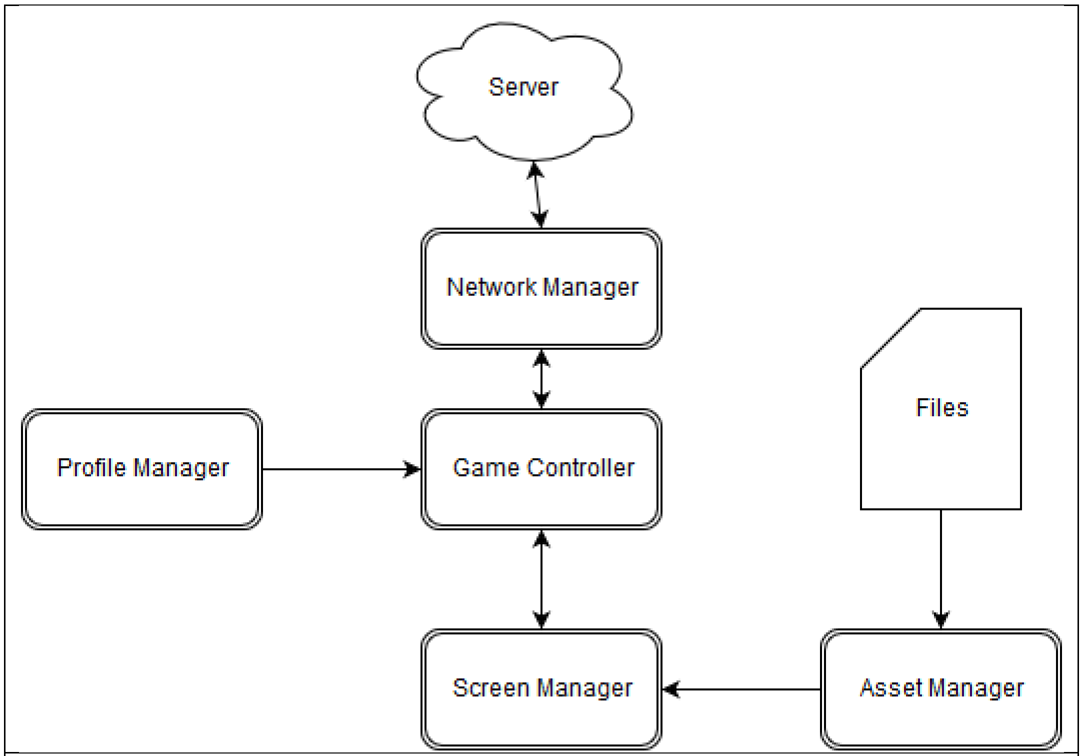

Fig 4.1: High level block diagram for the client software

Game Controller Class: The game controller class runs the game and tells every other class what to do. Most of the important data will be in the game controller to include current game rules, the statements and cards in play,

Network Manager Class: The network manager establishes a socket connect to the server. When then game controller needs to send information it uses the network manager. Information coming from the server will be received by the network manager, and relayed to the game controller.

Profile Manager Class: The profile manager loads and stores profiles locally. Each profile will be its own class which will be stored in a list in the profile manager.

Screen Manager Class: The screen manager draws everything to the screen and takes input from the user. The screen states will be controlled with an enum in the game controller.

Asset Manager Class: The asset manager is primarily responsible for loading files at run time and providing data to other classes. All of the textures and sprites used by the screen manager class will come from the asset manager. Additional sound files for music and sound effects will also be stored in the asset manager. The loading screen is used at the start of the game running and waits until the asset manager has finished loading files.

Profile Class: The profile class will need to consist of all of the variables that define the current user. This means that the profile will need a name and some form of icon to distinguish this profile from others. Additionally profiles should contain a list of friends for use in connecting users directly to game lobbies with friends already in them.

4.2.2 Client Screens:

The Client will essentially be a state machine. There will be multiple states each corresponding to an action the player wishes to carry out, such as a game state that will allow the player to play the game or a Join Game State allowing players to join a game. Each of these states will be presented as screens to the user. Fig 4.2.2 shows a diagram showing a flow chart for the various state screens.

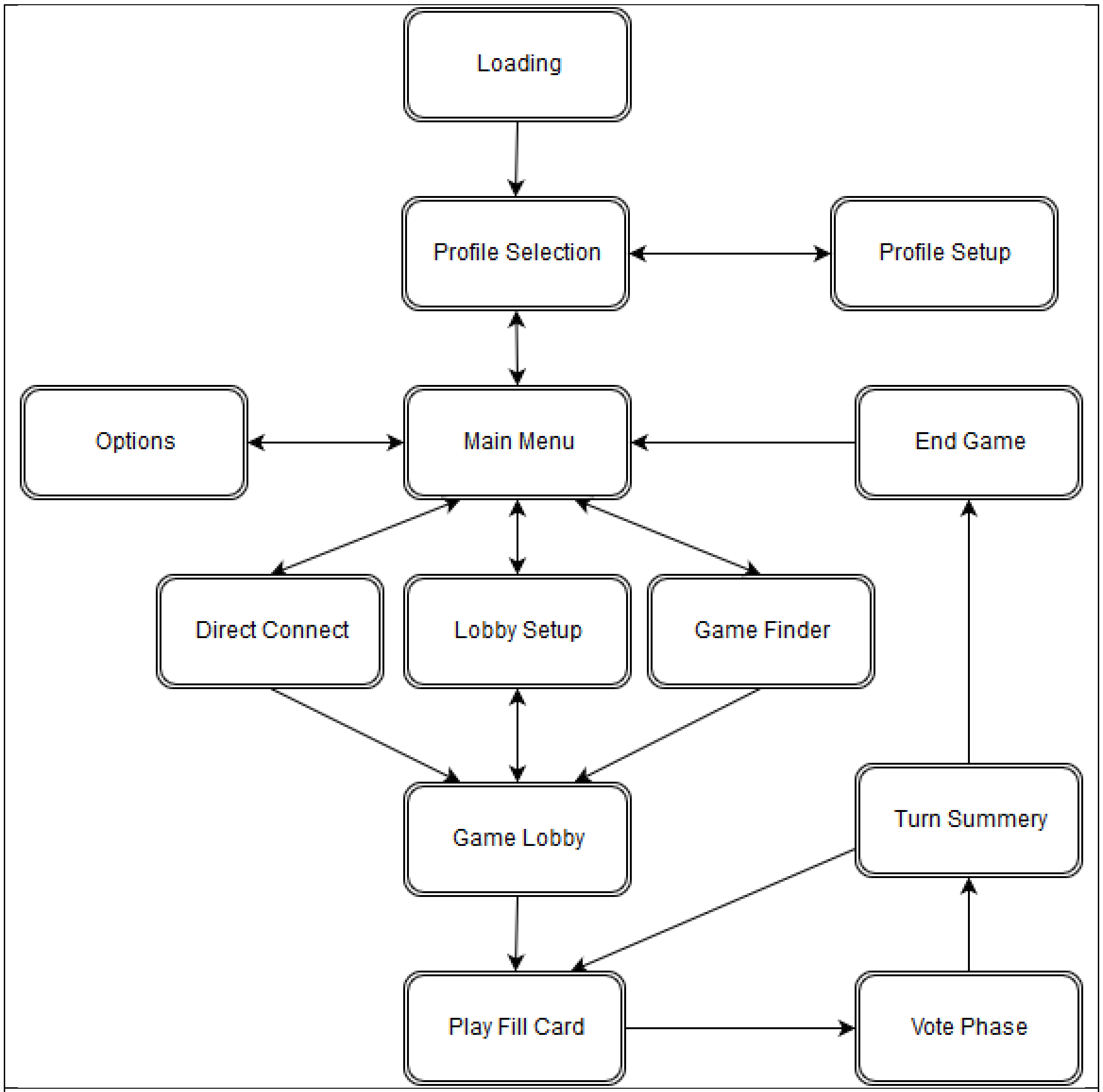

Fig 4.2 Flow chart for the client states

Loading Screen: The loading screen serves the purpose of giving the player something to look at while files are loaded. Since the game is really simple the need for the loading screen will probably only be at the initial launch of the client program.

Profile Selection Screen: The profile selection screen is the first screen that users are able to interact with. Users must first select or create a profile to start using the client program. If there are no profiles, then the screen will only have the option to go to the profile setup screen.

Profile Setup Screen: The profile setup screen is linked to the profile selection screen, and is used to either create a new profile, or edit an existing profile.

Main Menu Screen: The main menu screen serves as kind of a logical central hub to the program; from it users can connect to lobbies, go to the options screen, or go back to the profile selection screen. If the user quits out of a game they will be returned to the main menu screen.

Options Screen: The options screen is for the user to change aspects of how the program works. Music and sound effect volume can be changed or completely muted. If the game gets animated sprites, then the option screen will allow that feature to be turned off.

Direct Connect Screen: The direct connect screen needs to allow the user to connect to a particular. Ideally the user can directly connect to a friend's game lobby, but the minimum would be the ability to enter a lobby ID to join the related game lobby.

Game Finder Screen: The game finder screen will allow the user to choose several options and automatically connect them to a lobby that matched the selected options. If a game cannot be found in a timely manner, then the game will switch to the lobby setup screen.

Lobby Setup Screen: The lobby setup screen allows a user to create a new game on the server, which others can join. The options to include in the lobby setup screen include player limit, ready button use, lobby max time auto start, game mode, win condition threshold, turn limit, turn time limit, player kick threshold, card discard options. Once the user is done the game will shift to the game lobby screen.

Game Lobby Screen: The game lobby screen is like a waiting area for users to see information about the match rules and other players, while people are joining and getting ready. The screen will likely have a ready button for each user, so the game will only start if everyone wants to play. A time limit could be specified such that the game will start after so much time has passed irrespective of the ready button.

Play Fill Card Screen: The play fill card screen is the first screen of the actually game playing screens. In this screen the game displays the statement to fill, and the cards the user can play. The user should be able to drag their cards onto the statement card, which will change the text on that card to the statement with the fill card replacing the blank. Once the user has selected their fill card they will hit a submit button, and transition to the vote phase screen. Fig 4.1 shows the current design mockup for the Play Fill Card Screen.

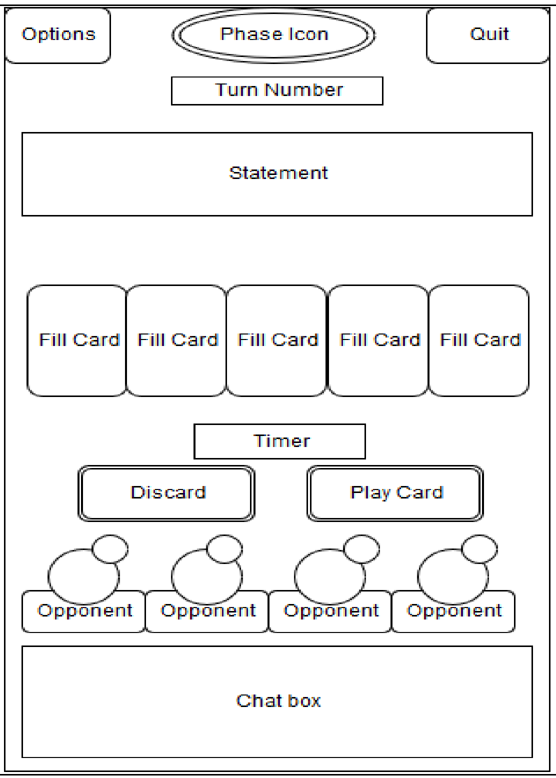

Fig4.1

This is the current design for the play screen.

Vote Phase Screen: The vote phase screen is similar to the play fill card screen except that this time the user is selecting from their opponents fill cards. The user will not be able to tell who played any of the cards that they are voting on. Once the player votes, the game moves onto the turn summery screen.

<u>Turn Summery Screen:</u> The turn summery screen serves as a place holder for rounds, and lets the players know who played each fill card, and the points awarded. Once the user is done with the turn summery screen the game will go back to the play fill card screen to start the next round, or it will go to the end game screen.

<u>End Game Screen:</u> The end game screen is similar to the turn summery screen except that summarizes the entire game and declares the winner(s). Once the user is done viewing the screen they are given the option to go back to a lobby with the same settings as the last game, or just go back to the main menu screen. Fig 4.2 shows the current design mockup for the End Game Screen.

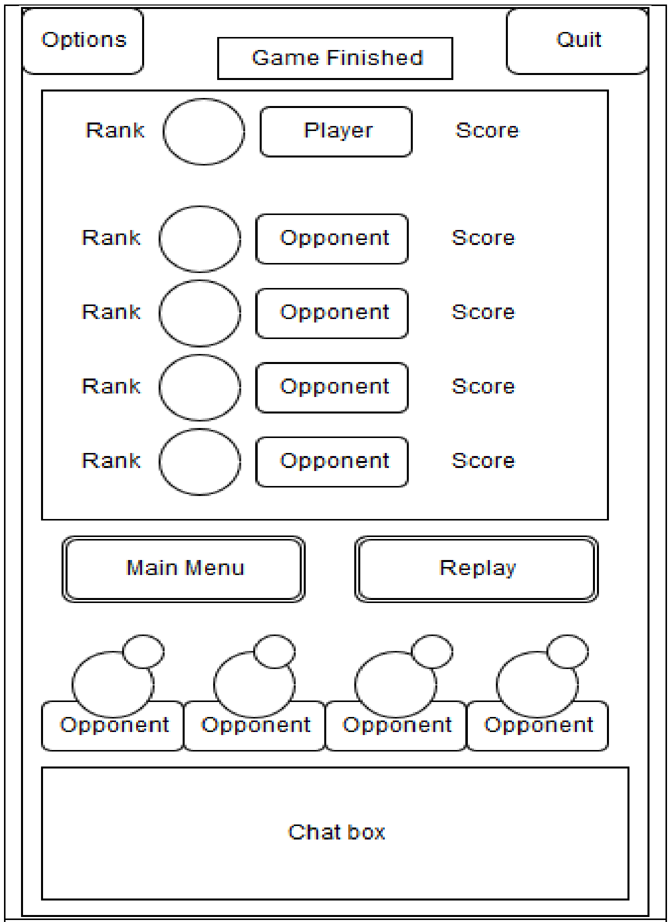

Fig4.2

This is the current design for End Game Screen

4.3 Server:

While the user interface for the game is implemented on the client, the software that drives the game is included in the server. The server has two responsibilities: connect the players, and manage the game. Since this is a turn-based card game, the server-side calculations don't involve any physics, graphics, or any other complicated calculations. This piece of software will be implemented in Python to allow for rapid prototyping and flexible development. Python should be able to handle the required tasks without a significant performance decrease.

4.3.1 Networking:

One of the reasons Python was chosen for the server implementation are the vast multitude of choices for web implementation. The Flask micro-framework provides the basic functionality of a web server. Flask was chosen over alternatives such as Django and Twisted for its small footprint and ease of use. Flask has a socketsIO library to implement the web sockets that will communicate with the client's .Net sockets. Communication via sockets will allow communication channels to remain open and allow the gameplay to flow smoothly without having to re-establish connections.

The server will have a process that is constantly listening for clients to connect. When a new connection is found, the server will get identification information from the client. Initially, this will just be a username and profile image id, but future plans could involve password credentials to retrieve profile information from a server-side database. Once the server has an identity matched with a connection, it will pass that information to the matchmaker. Initially, the matchmaker will be a part of the connection process but could be split into a separate process if we need to provide more complicated matchmaking.

4.3.2 Class Interactions:

The server will consist of four primary classes. There will be a Profile class, a Table class, a Player class, and a simple Deck class. The Deck class is a wrapper for a list of cards that includes a function to distribute a random card. The profile will contain all information about a player that exists outside of a game. This includes a username and a profile image, and could be upgraded to include a password and other permanent data such as match history and friend list. The player class contains player data that exists in-game, such as their hand and current score. This class will include functions to allow the player to draw and play cards as well as cast votes. The Game class will represent a game in progress. It will have a list of players, the card decks, and any game options that affect gameplay.

4.4 Tests:

In order to ensure that the various pieces of software can function reliably we will develop and deploy various tests for the Client and Server software. These tests include manual inspection of both the client and server software by members of the development team, scripts designed to test input and outputs to ensure error handling and reliability under heavy user interaction, and user review sessions where we will allow users to test our project and receive feedback from the players.

Developer Analysis: While developing these pieces of software each member of the team will individually test each component. This involves the team playing the game, testing the various options to ensure they are functioning, and in general making sure the software seems to be functioning properly. While these may not be the most thorough of methods it is our first line of defense in ensuring software quality.

Server Tests: To test the server we will develop several python scripts that will provide random inputs through its socket connections. We will develop two versions of these test scripts, one that will provide valid, but randomized input and another which will randomly generate inputs.

The first script will keep track of the current game state, monitor the signals sent by the server and respond with signals that will mimic the client signals. This script will in essence play the game. This script will test the various user inputs, such as card selection and chat functions and can help stress test the maximum occupancy of the server.

The other script will test the robustness of the server by generating bad inputs. This script will ensure the server can withstand a multitude of incorrect inputs gracefully and without fatal error. With both of these script tests we can ensure that the server will be robust, durable and dependable.

Client Tests: Currently our team has no additional tests for the client side software apart from us verifying the various input methods are functioning correctly. Our client will be fairly simple, serving only as an interface to the server, as such we feel that stress test may be unnecessary for this particular piece of software and manual inspection will suffice for quality control.

User Review: The most important of these tests is the user review sessions. Once our project is in a playable state our team will begin seeking out independent testers for our product. We will allow users to test our project and question them on various aspects of the game. We will then use this feedback to improve on our game's system, interface and content.

With these tests we hope to perfect our project and craft an app that is both fun and technologically sound.

5. Team:

We are a 3 man team of computer science students from the University of Tennessee. Our team has limited experience with mobile platforms and networking and as such there will be many unknowns during development. Many of the tools we must employ are new or not well known by the various team members. In order to counteract this learning curve for the various technologies our team member's roles will be fixed, allowing them to focus entirely on a single technology without worrying about learning several API's. Although each member will assist the other where they can, we believe that having dedicated roles provide the best option to develop this app using technology that we have had little exposure to. Although this project may be challenging for us, our team believes that we can create an impressive application in the allotted time.

Travis Brown:

Travis has some experience programming in Python, but not on the same scale as this project. Most of his experience has been using regular expressions and manipulating lists. He has a basic understanding of networking and has implemented a bare-bones chat server written in C. The Flask framework and SocketIO library are both completely new for Travis.

Role: Primary Server programmer.

Skills: Programming in C, C++, Python.

Mike Shoots:

Michael has academic experience with programming in C/C++ and python, and personal experience working with the .net framework in C#. Michael has created simple games with Unity 3d, but the games were purely single player. Michael has some understanding of networking, but no experience implementing it for a game on Unity. Additionally Michael has no experience porting games to mobile platforms like Android, but Unity should allow for easy porting. The primary difficulties Michael will face with this project come from getting a Unity client to communicate with the server via .net sockets, and porting the client to Android.

Role: Primary client programmer, Implementing the game features and interactivity in the client application, Migration of client from PC to android.

Skills: Programming in C, C++, C#, and Python. Experience working with Unity 4 and Unity 5 editors. Experience working with Microsoft Visual Studio.

Tyler Poff:

Tyler has experience in a multitude of languages including: C, C++, C#, Python, Ruby, R, Java and JavaScript. He has also worked on several unreleased video games in various genres. He has an interest in graphics and has experience with both DirectX and OpenGL graphic libraries as well as several graphical creation toolkits such as Blender and the Adobe Creative Suite. Although he has experience in graphics programming he has never used the Unity framework, however his experience with C# scripting and game development makes him confident that he can pick it up with relative ease. He also has some experience with socket networking in C# having made a IRC client that can connect to various chatrooms such as the chatrooms hosted by the streaming service Twitch and the anonymous chatroom Omegle. His primary responsibilities on this project are general programming on both the Client and Server side and asset creation.

Role: Client Programmer. Server Programmer. Asset Creation.

Skills: Programming in C, C++, C#, Python, Ruby, R, Java and JavaScript. Experience with 3D graphics. Experience creating assets using the Blender 3d Creation Suite. Experience in Adobe Creation Suite. Experience in Audio Editing. Experience in general Game Design.

6. Project Management:

Although the technical side of the project is of great importance, with any team project there must be a significant amount of time attributed to managing the project itself. Crafting and maintaining a development schedule, keeping the team organized and busy with their tasks, ensuring that legal and ethical dilemmas concerning the project are dealt with and managing and collecting resources are just some of the numerous activities that must be managed carefully. Without careful consideration to these tasks a project may fail, not because of technical problems, but because the team lost their direction and could not pull together.

6.1 Project Schedule:

In order to ensure our team completes the project by the deadline we will hold regular team meetings. Every Monday, Wednesday and Friday we will meet at 8:30 AM for 30 minutes to highlight our progress or discuss potential hindrances concerning the project. We will have an in-depth meeting once a week on Friday at 12:00 PM to discuss the current state of the project, discuss what we will pursue in the coming week, highlight successful ventures of the past week and reevaluate the project schedule if necessary. With these regular meetings we will remain on track and be able to address any hindrances that may affect the project quickly and efficiently. These meetings will also serve to keep each member of the team accountable, ensuring that every member completes their individual components on time.

Although these weekly meetings will assist us in keeping us accountable, without a roadmap to guide our project we may find ourselves directionless in our work. To avoid this, our team has set weekly goals and milestones that we will achieve in order to complete this project. The following table outlines our team's weekly goals.

Weekly Goal Schedule.	
Date:	Goals:
March 4th	• Basic network connection between Client and Server • Initial Fill-In card list • Initial Situation Card list • Draft of User Interface layout • Draft of server-client communication protocol • Artwork for client draft.
March 11th	• Basic User Interface Implemented • Card selection implemented • Server-Client communication protocol implemented • Single User/Single Game instance, Server Implemented • Anarchy Mode Implemented • Testing begins
March 18th	• Client Sound Effects implemented • Client background music implemented
March 25th	• **Status Report Due with Minimum viable System** • Multiple player support for Client • Multiple player support for server • Testing scripts for server implemented and deployed
April 1st	• Multiple Game instances support for server • Multiple Game instances support for client • Game finder implemented • Open player testing
April 8th	• Chat support for server • Chat support for client • Democracy Mode implemented • Monarchy Mode implemented
April 15th	• Artwork for client finalized • Fill-In card list finalized • Situation card list finalized
April 22nd	• **Final Project Presentations**
April 29th	• **Final Project Report Due**

With these team meetings, weekly schedule, and the determination and dedication of our team we are fully confident that we can develop a fully functional, well-designed and entertaining mobile application before the final deadline.

6.2 Release Versions:

During Development we will have multiple revisions of our product; each subsequent build will improve upon the last adding more features and improving upon the existing systems. The Minimum Viable Product will be the base iteration of our project and will be a skeleton of what we will choose to implement. We will build upon this minimum version until we achieve a variation that implements the majority of our planned features. This will become our Initial Release Version and we will make this iteration available to the public.

Minimum Viable Product: Our Minimum Viable Product will consist of a single game mode with predetermined options. The client will open to a Main Menu with two options: Start Game and Exit. Upon selecting Start Game, the client will connect to the server to play a single game with "AI" opponents. When the game ends, the client will return to the main menu to start another game. Only one player will be able to access the server. This should be sufficient to get an idea of what the finished product will be like, and will give us a solid foundation to build upon.

Initial Release Version : Our Initial Release Version will build upon our minimum product. The Main Menu will have two additional buttons: Login and Options. Login will allow the player to choose a username and profile icon and Options will allow players to turn music and sound on or off. The server will be able to accommodate multiple concurrent games, and will have a way to replace players that aren't participating. When joining a game, there will be an option for a timed game in addition to the standard game mode. The Initial Release Version will also include original sprites for the background, introduction or splash screen and player icons.

6.3 Project Constraints:

Since this project is seeking to create a mobile adaptation of an existing game, one may wonder if there may be legal action taken against its development. Another potential issue is the content of the game being seen as offensive and discriminatory.

Although we are seeking to create an adaptation we do not see any legal ramifications from the developers of Cards Against Humanity. The original game is distributed under a Creative Commons BY-NC-SA 2.0 license[2] This license permits redistribution and alteration of the game so long as there is no monetary gain for the project. Our team has no desire to sell or collect revenue from our adaptation; as such we fall safely within the Cards Against Humanity's license.

As for the content of the game, our team will heavily modify the source material to remove any discriminatory or offensive content. During this editing process will strive to keep the spirit and humor of the game intact to the best of our ability. We will also look into developing our own content to add to the game to help fill the void left by the removed content. We will seek to create appropriate substitutes for cards that we feel should not be in our application.

6.4 Resources:

During development we will have to find or create a variety of different resources. For the technology we will have to research ways to implement our desired systems for both the Client and Server. We will also need to find or create visual assets for the User Interface. Both Unity3D and Flask have an abundance of documentation and a thriving community in which to ask specific questions that may arise during development.

For assets to use in the interface, while there are plenty of resources online to draw from, many of these artistic repositories come with many legal restraints. As such we feel it's best if we create our own artwork for our game to avoid any legal complications that may arise. While the final product may have original depictions, we will use stand in graphics during the initial phases of development. All files pertaining to the project will be managed via the Version Control service Gitlab.

With the abundance of resources at our disposal we feel confident that we can find everything we need to complete this project.

7. Reflection:

To Be Added

8. Works Cited:

[1] “Case Study: Cards Against Humanity” (2012, July 26) retrieved from https://www.kickstarter.com/blog/case-study-cards-against-humanity

[2] "Cards Against Humanity." (2011) retrieved from https://cardsagainsthumanity.com/

[3] "A Party Game as Despicable as Your Friends." (2011, February .2) retrieved from https://www.thrillist.com/entertainment/cards-against-humanity_comedy_toys

[4] “Eight nerds get rich off a game where Oprah sobs into a Lean Cuisine” (2014, May 16) Chicago Sun-Times

[5] "Cards Against Humanity Review." (2011) retrieved from http://boardgamegeek.com/boardgame/50381/cards-against-humanity

[6] “A Card Game For Assholes” (2013, January 30), retrieved from http://www.avclub.com/chicago/articles/a-card-game-for-assholes,49896/

[7] "The Leading Global Game Industry Software." (2015, December 01) Retrieved from https://unity3d.com/public-relations