Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

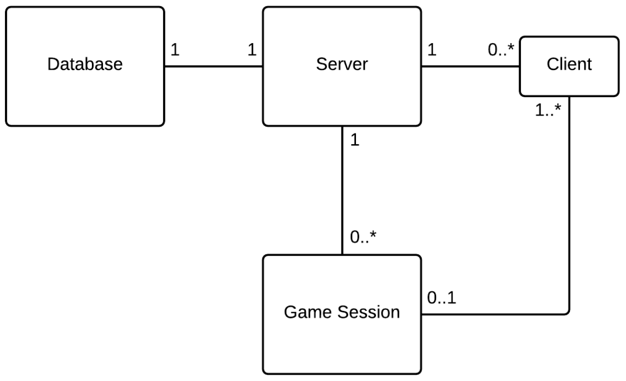
Design Document: Minimap

Purpose:

The system we will design will allow multiple clients to connect to a server. Clients will then be able to see other clients' GPS coordinates based on which clients are in their current group session to allow friend-finding between clients. This will be extended in multiple games that will enhance the outdoor world with the video game realm.

Design Outline:

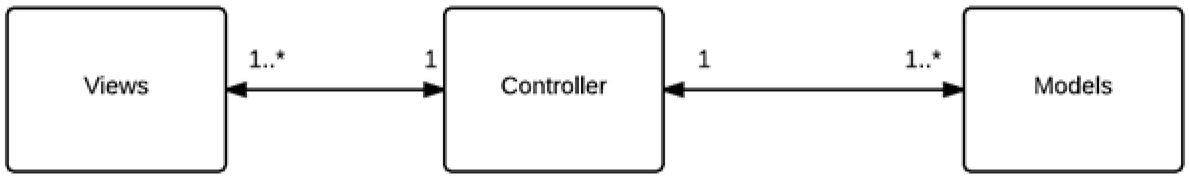
In this project we will employ the use of a client-server model. We will have one server running, and 0 or many clients will connect to the server upon running the mobile application. To connect clients to each other, the server will take requests from the clients to create sessions that can hold two or more clients. The session will allow clients to then request more clients to be added to their session, and the server will handle clients disconnecting from the session.



The server will be split into four main components. The basis of the server will put be in charge of starting different game sessions. Each game session will be propagated by multiple client objects. These clients for the second component. Each client object will be in charge of any communication with the corresponding mobile client. The game sessions form the third component. These sessions will manage all game specific messaging. Any actions that happen as part of a game, such as a flag capture, will flow through one of these sessions. Multiple sessions can run at the same time with different clients, allowing for multiple games to happen simultaneously. The final component of the server will be a MySQL database. This database will store all relevant information such as the current gps location of every client.

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

Client



Client: The client will display and receive the information for our application. We will use the MVC pattern (Model View Controller) to manipulate the data. It will be and Android application that is written in Java and XML.
Views: The view will display the information on the screen and the user can interact with it. This is how we will capture user input to send to the controller for processing.
Controller: The controller will send and receive information from the views and the models. It will determine where the data goes and what classes should be accessed.
Models: The models hold all the data for the application. It will send information to the controller if requested and can take information from the server when needed. It will be the connection between the application and the server as well.

Design Issues

Functional Issues:

1. What architecture model should we use?
    a. Server - Client
    b. Peer - Peer

    Decision: We decided to use a Server-Client architecture. This will allow us to run most of the game specific logic on the server. This reduces the workload of the mobile application, potentially saving battery power. Additionally, the clients must only send and receive information from one place, instead of opening connections to all the other clients in the game.
2. Which game types will we make?
    a. Friend Finder
    b. Marco Polo
    c. Capture the Flag
    d. Sardines
    e. Slender

f. Snake
g. Assassins

Decision: We decided to do games a-e. We did not choose Snake because the uncertainty around GPS locations (~15 ft) would make this game very glitchy. We didn't pick Assassins because we did not feel that it would translate to a minimap that well. Being able to see where your target was would make the game very unfair.

3. How do we add users to game sessions?
   a. Games consist of two group sessions that connect
   b. Players connect individually to game sessions via game invite

Decision: We decided to make players connect to games via an invitation from someone else in the session. At this point the player will then choose which team he will join. By doing it in this method, players can move between sessions more freely and there's no initial setup prior to starting a game.

Non-Functional Issues:

1. How should the database be managed?
   a. MySQL
   b. MongoDB
   c. SQLite

   Decision: We decided to go with MySQL for database management because it was the more familiar of the two to our group.

2. Which mobile platform should the app support?
   a. Android
   b. IOS

   Decision: We determined that Android was the better choice, as the large majority of the team uses Android phones, and Android has a much larger market share, so our app would be able to reach a wider user base.

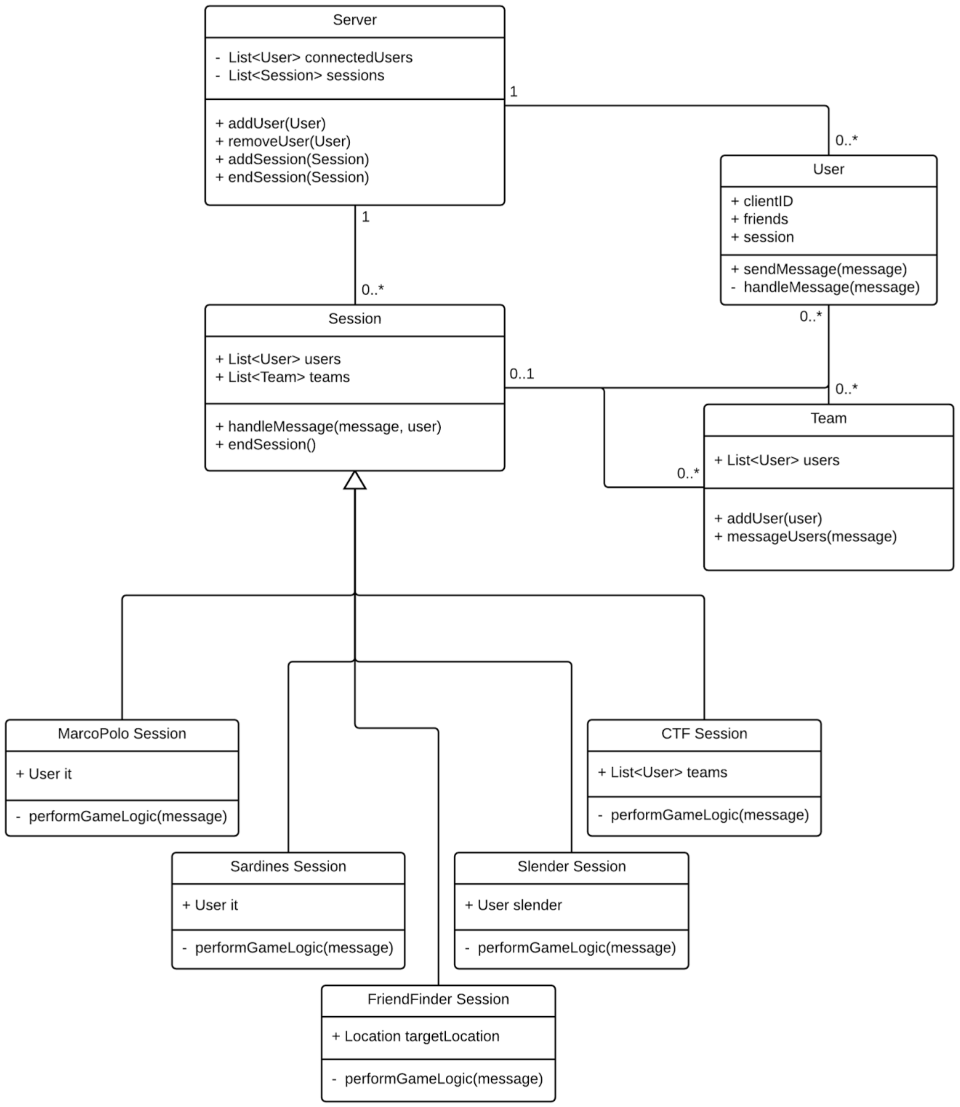
3. How should users be handled?
   a. Google+ integration
   b. Facebook integration
   c. Independent registration

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

Decision: We opted to tie our users to Facebook accounts to streamline the creation of friend lists. We decided that it would be much more convenient for our users than creating a separate account and manually searching for each friend. We opted for Facebook integration over Google+ integration due to the wider user-base.

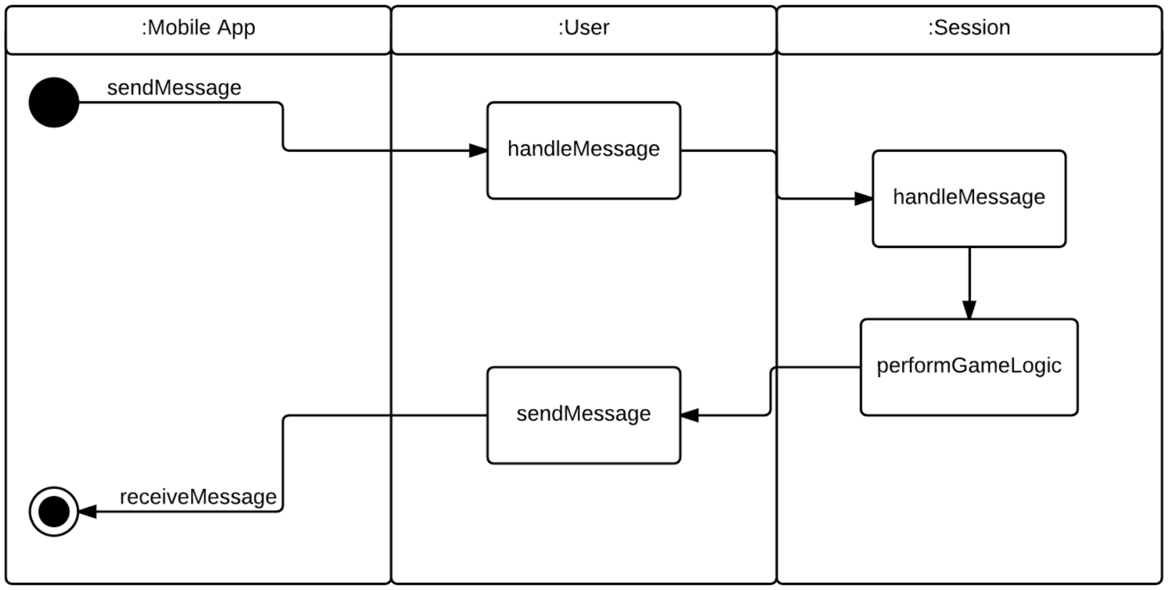
Design Details

Server:



Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

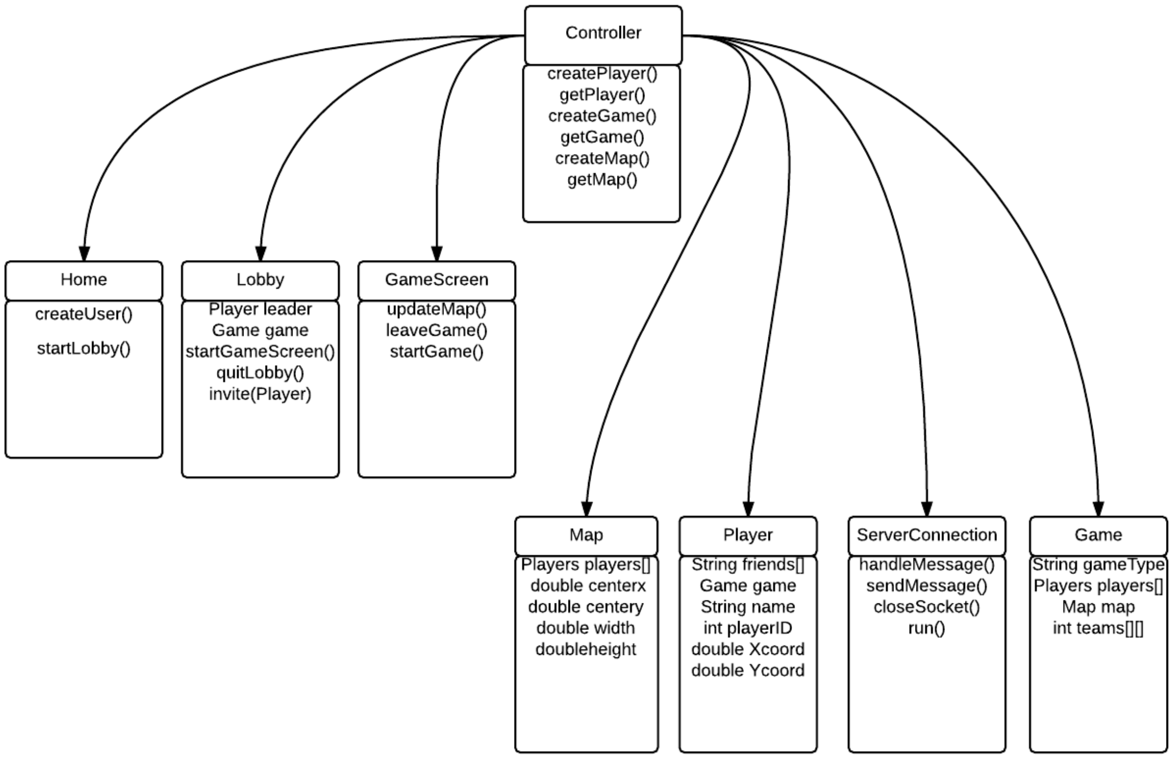
- The server will contain a list of all connected users. Whenever a user connects, it is added to this list.
- When new games are created, they are added to the list of currently running sessions in the server.
- The User class manages all socket communication through two methods, sendMessage and handleMessage. The User class will also contain information such as the clientID and the current session.
- Each game type will be a subclass of the Session class. The Session class will contain all general purpose logic while the specific game sessions will contain game specific logic.



- When a client sends a message to the server, it is received by its User object. The User object then forwards the message its current game session. The game session performs any needed game logic and, if needed, sends an update to all the clients in the session.

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

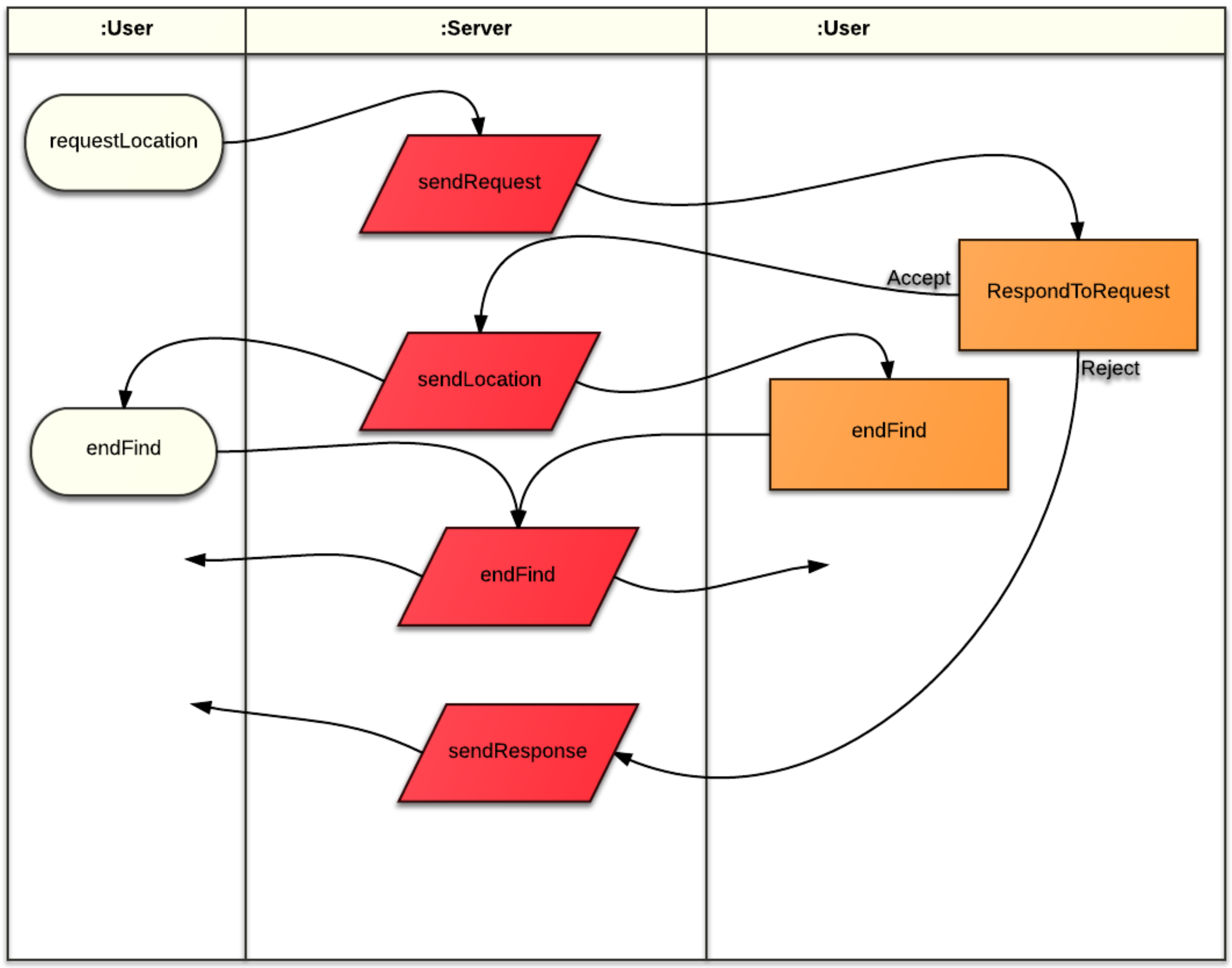
Client:



The client will be made up of three screens: Home, Lobby, and GameScreen. Home is where a user will login and select a list of games. Lobby is where the player can add players to the game and set the gametype. The GameScreen is where the map will be and other effects depending upon which game is being played.These views will be set by the controller class and will send user input there. The controller will modify our models: Map, Player, ServerConnection, and Game. Map will hold the data for people on the map and where the map is set to. A Player will represent the user and players in the game. The ServerConnection will deal with sending and receiving messages from the server. Finally, Game is a superclass for all the games we create.

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

Friend Finder:



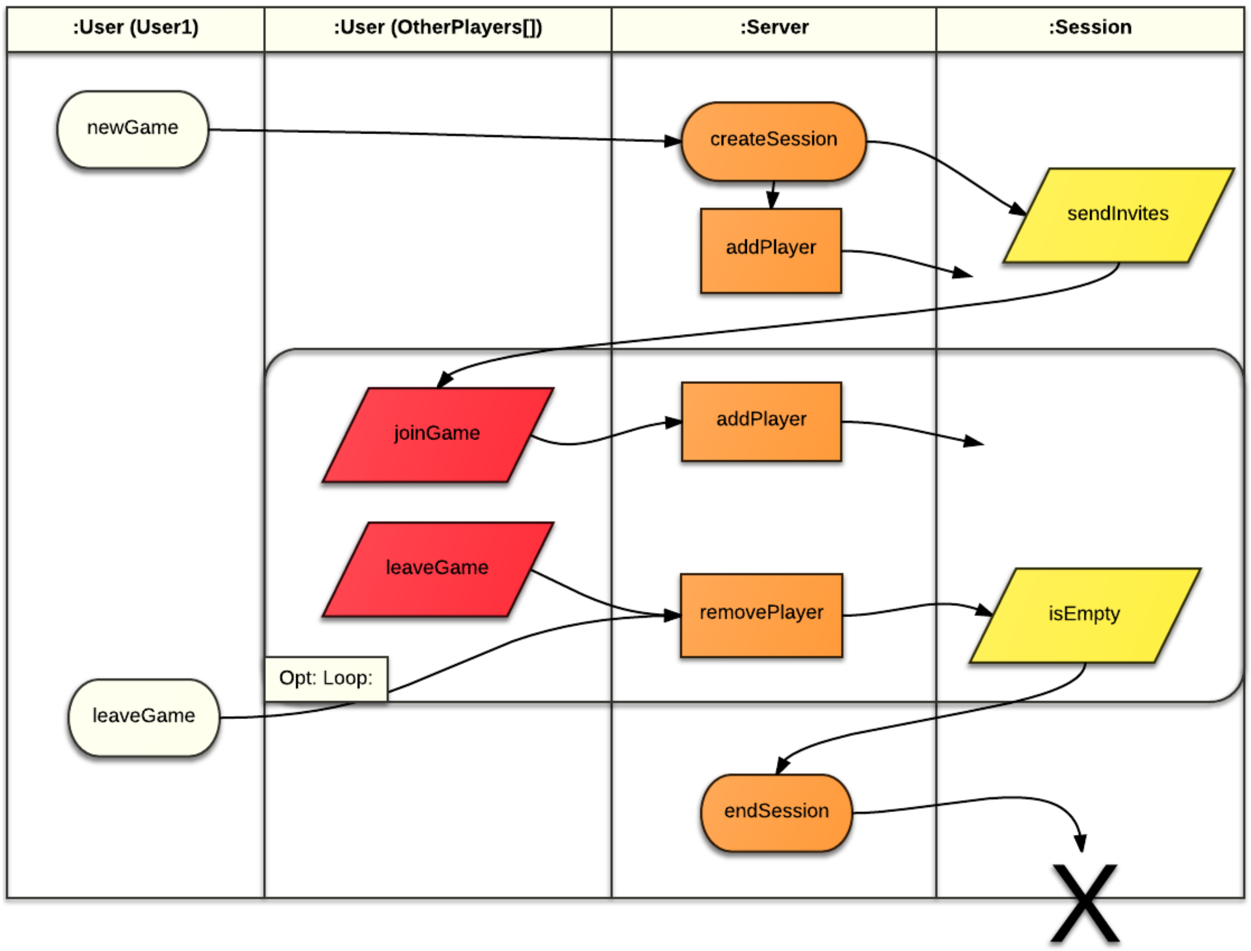
- A user can use “Friend Finder” to find friends
  - a user can use “friend finder” to find multiple friends at a time
- Sequence of events
  - User1 initiates the process by requesting for the location of another user
  - The server will send a notification to User2
    - Notification will present identity of User1
    - Notification will allow for User2 to either accept or reject the request
  - If User2 accepts the request
    - The server will send the location of User2 to User1 and vice versa
    - Once one of the two Users ends the process either by finding the other user, manually ending the process, or closing the app, the Server will end the process for both users

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

- If User2 rejects the request or is unavailable (i.e. does not have the app running)
  - The server will send a notification to User1 that the request was terminated

- Once a User accepts the request of another User to share their locations, a map will appear on the screen of both users that shows the location of the other user
  - Once the users are too close to accurately use GPS coordinates, The app will show a message that tells the User that they have found their friend
    - Once the user closes the notification, the process will terminate
    - alternatively, the user can manually stop the process at any time by tapping the “Cancel” button

Games:



Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

All games will follow a similar process for game initialization. The starting user will choose what game they want to play. This user will select game specific options such as the boundary size. The user then selects which friends to send an invitation to. This notifies the server to start a game session. Once all users have responded to the invitation the host can hit the start game button. This starts the game logic part of the session.

Capture The Flag:

- Setup:
  - Flag locations are set, and all players must have starting positions near their flag.
- Goal: Capture the enemy's flag and bring it to your team's side of the line of scrimmage before the opponent does the same.
- Logistics and Gameplay:
  - Tagging: Tagging is done by assuming that if two players are standing within 10 feet of each other, and both players are standing still for more than 3 seconds, then the person who is on the opponent's side of the line of scrimmage is considered tagged. The tagged persons phone will display the person as tagged. Until they are tagged in by someone on their team, they cannot capture the flag again. The player must be in the jail area to be tagged back in.
  - Jail: Jail is located near the flag zone, where players that have been tagged will stay until an ally tags them back into play. Upon being freed, the player released must re-enter his side of the map before returning to enemy lines.
  - Follows the rules of traditional outdoors CTF
    - The match is won by grabbing the enemy's flag and bringing it to your side of the field designated by a predetermined line of scrimmage.
    - Tagging an enemy on your side of the line of scrimmage sends them to a "jail" zone where he/she must be freed by a teammate tagging him/her.
- Game Ending:
  - When one team brings the enemy flag to their side of the line of scrimmage, that team wins. Matches will typically be a best of three, but can be adjusted.

Marco Polo:

- Setup:
  - A new Marco is selected each round. The rest of the players are Polos.
  - The Marco is given a map and a Marco button with a timer next to it.

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

    - The Polos have a Polo button and an indicator giving a general indication of how far the Marco is as of the last Marco signal.
- Goal:
    - The purpose of the game is to evade detection on the Polo turns for as long as possible. The longer the Polos evade detection, the more points they get.
- Logistics and Gameplay:
    - After a time interval displayed by the timer, the Marco can press the button to send a Marco signal to the Polos. This action sends a notification that the Polos must respond in a timely manner or they lose all of their points earned from the current round.
    - When the Marco tags a Polo by entering a small radius, each player is notified via vibration and the Marco gets a tally mark denoting that one of the Polos has been eliminated.
    - The Marco signal timer duration is inversely proportional to the distance between the Marco and the nearest polo, to prevent longer distance games from becoming stalemates.
    - The Marco timer will also be shorter for games with a larger number of players, to encourage shorter rounds and ensure that the game finishes before the players get bored.
- Game Ending:
    - The game ends after every player has been a Marco. At this point the scores will be tallied and the winner selected.

Sardines:

- Setup:
    - One User will hide from all of the other players
        - Users that are hiding are Sardines
    - All other users will search for the Sardine(s)
        - These users are Seekers
- Goal: Find the Sardine(s)
- Logistics and Gameplay:
    - Each player will have a minimap which will give them visuals of where the Seekers are on the map
        - No one will be able to see the location of the Sardine(s)
    - Seekers run around looking for the Sardine(s)
        - Once a Seeker finds the Sardine(s), he or she becomes a Sardine
            - this User's location disappears from everyone's minimap
- Game ending:

    - Once all players have found the Sardine(s), the game ends
    - An option will appear to restart the game
        - The next Sardine is the first Seeker who found the Sardine(s) last game

Slender (Name may change later if needed):

- Setup:
    - Group of players vs an AI or person representing Slender
    - Players will split off in their own directions looking for objects in the virtual world, all while trying to avoid “Slender” who will slowly be approaching players
    - Player who initializes the game will set the game boundaries (maximum playing field to be determined)
- Goal: Get as many “objects” as you can, or survive longer than other players and have the highest score
- Logistics and Gameplay:
    - Each player will have a minimap which will give them visuals of where all other players are on the map
    - There will be score modules corresponding to each player with:
        - how many points they have
        - how many relics they have
        - distance from the other players
    - As slender approaches, his dot on the radar will flash and be visible to all players within a certain radius
    - A low thumping sound will get more frequent as slender nears a player
    - The closer players are, the more likely slender will be to go after them, making it advisable to split up
    - The game ends when either someone has won (found all of the relics) or if everyone has been killed by slender
        - in event of the later condition, winner is determined by point count
    - Players continuously send location to the server
    - When players get in range of certain objects, it appears on their minimap
    - when players get close enough to the object, the server will register it as a “collected” object for that player, and assign points corresponding to the item’s value
    - Server calculates next position for slender based on nearest player and if players appear clustered
    - Server decides to send appropriate signals to all players within intervals corresponding to distance

Team 4
Michael Crabill
Joe Coy
Zachary Deganutti
Nikolas Ogg
Corey Pitzo
Matthew Tracy

        - Note: Only players who are closest to slenderman (half of players in game) will see his location blip on the minimap)
- Game Ending:
    - Player killed by slender:
        - If player gets in certain range of slender, he is considered dead
            - client state changes to “spectator”
        - Spectator mode:
            - player sees entire map
            - player can keep track of all player movements (who are still in the game, including Slenderman
            - Scoreboard is displayed so that they can see how everyone else is doing.
        - Win the game by collecting all artifacts (or all players are eliminated:
            - Server recognizes player has won, so it sends a signal to activate a ring on all devices to signal the game is over
            - Winner is displayed along with scoreboard and other information.
            - There is an option to play again or exit the game.