Software Design Document

By Reno Sarge, Michael Shliselberg, Patrick Lowry, Daniel Bergman, and Jordan Kaplan

1. Introduction

We are creating the Android application Base Run as a project for the Fall 2016 Course: Software Intensive Engineering ECE373 taught by Professor David Irwin. Base Run is a team-based smartphone GPS game where teams compete to capture and defend bases in order to strategically out-score one another in a set location and time limit. The user will choose the following parameters: location, time limit, radius of where bases can be, and the number of bases (up to a specific maximum bound). The purpose of Base Run is to use the smartphone GPS as a way to create a new, fun, and easy game for users to quickly learn and play.

2. Design Considerations

2.1 Assumptions

The software depends on accessibility of the android operating system and android hardware platforms to integrate the Google API for the UI. This project will require a consistent mobile data connection for all players active in the game. An issue may be the speed at which the server will have to input the client data from several users.

2.2 Constraints

The software depends and may lead to a possible issue of receiving data from multiple clients in a short time span. The game will rely on the abilities of the server to handle the network traffic well, including potentially over 5 concurrent connections.

2.3 System Environment

Client:

- Android OS 4.0 and above
- Android cellular phone hardware platform
- Must interact with UNIX server
- Client programming
    - Java 8
    - Android libraries (API 16)
    - XMLRPC library for networking
- Database Storage
    - Stores randomly generated maps
    - Returns map number, angle and radius of each base in the map

Server:

- UNIX OS
- Server programming
    - C++
    - Standard Libraries
    - XMLRPC_C library for networking
- Database Storage
    - SQLite3
    - Stores randomly generated maps
    - Returns map number, angle and radius of each base in the map

3. Architectural Design

3.1 Overview - Client, UI, and Server

Client:

- Handle all data from user inputs
- Collect all data for outputs to screen

- Communicate with the server to retrieve and store information throughout the game
- Interface with operating system on phone to pull data from hardware

UI:

- Collect all user inputs
- Display all outputs to the user

Server:

- Run and manage all active games
- Communicate with many clients simultaneously to collect and supply active game data
- Store individual client data during game session

3.2 Rationale

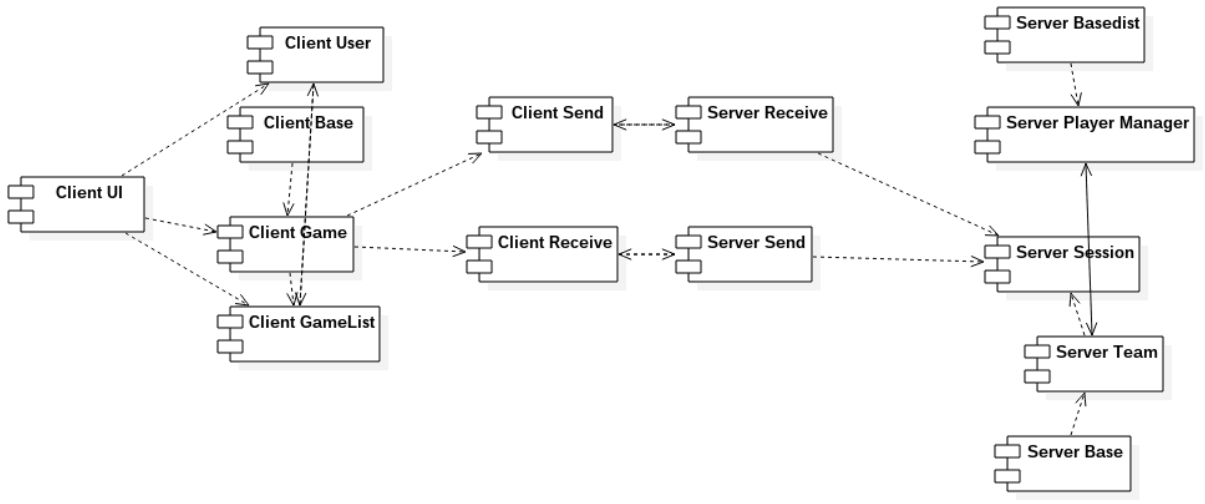
The components being used were selected to provide a simple, client server connection to handle consistent communication between many clients and multiple active sessions. Clients collect data through the UI, and send them to the server for processing and distribution to other clients.

Clients must be able to handle multiple forms of communication, both between itself and the UI which is feeding active information selected by the user, but also has to handle consistent outputs to the screen during gameplay, as well as client-server interactions to receive data pertinent to the game, including the current ownership of bases and their locations.

The server is needed to facilitate communication between multiple clients. All current users in a game must be able to send individual data to the server, including the current location to submit an ownership request for a base and communication to join games and teams. Additionally the server must be able to take this data sent by the clients, and distribute the information to all of the clients in the game in order for each player to receive consistent updates.

3.3 Conceptual View

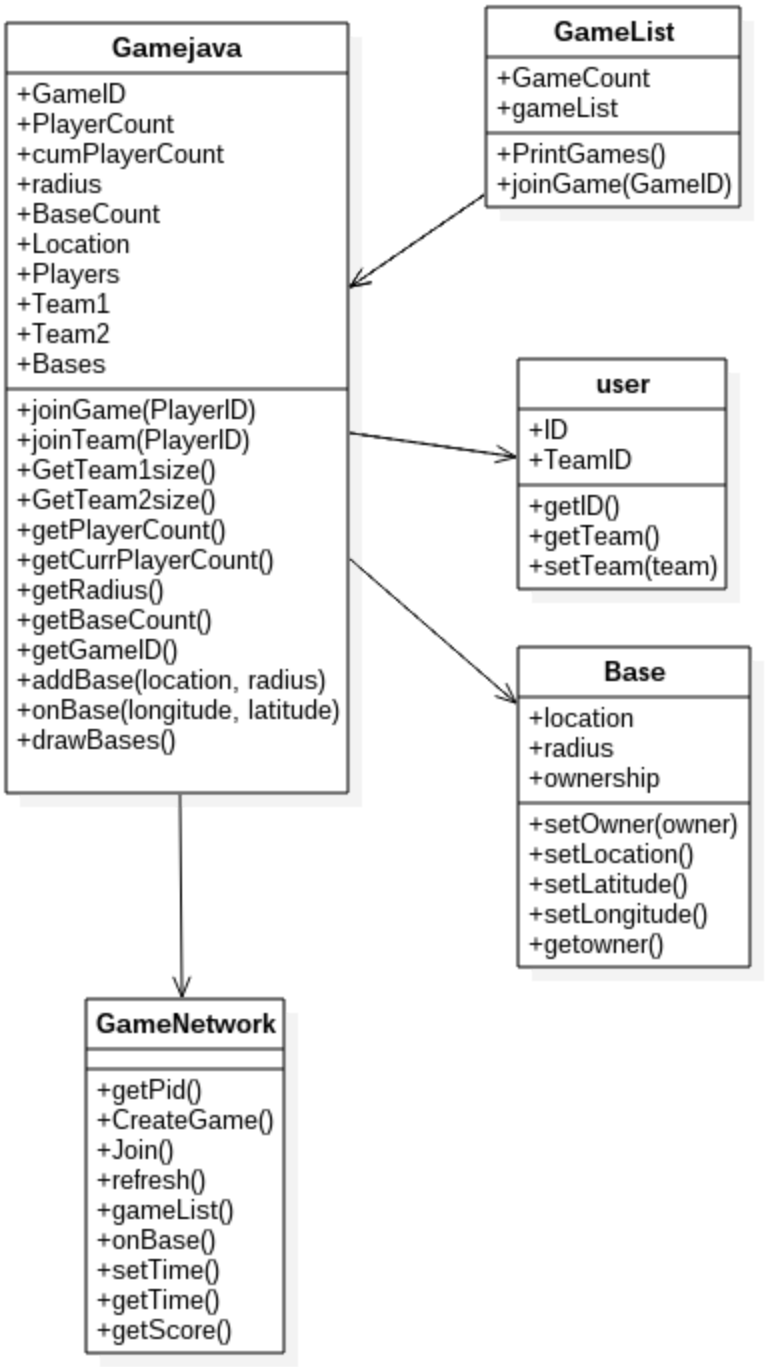
The conceptual view shows the logical/functional components of the system, where each component represents a cluster of related functionality.



4. Low Level Design

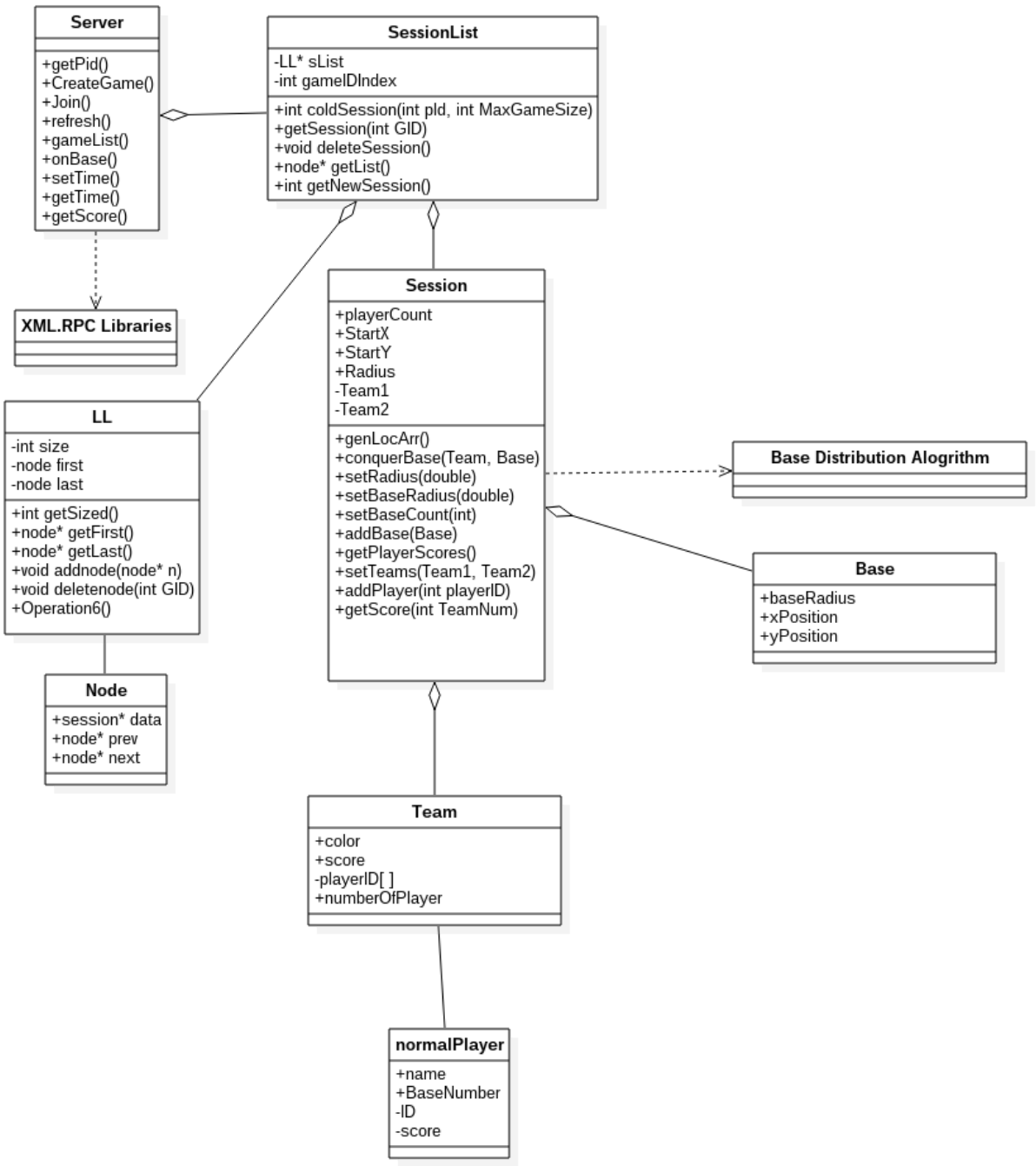
4.1 Class Diagram

Client Class Diagram
Interaction view of the various classes within the client backend.



Server Class Diagram

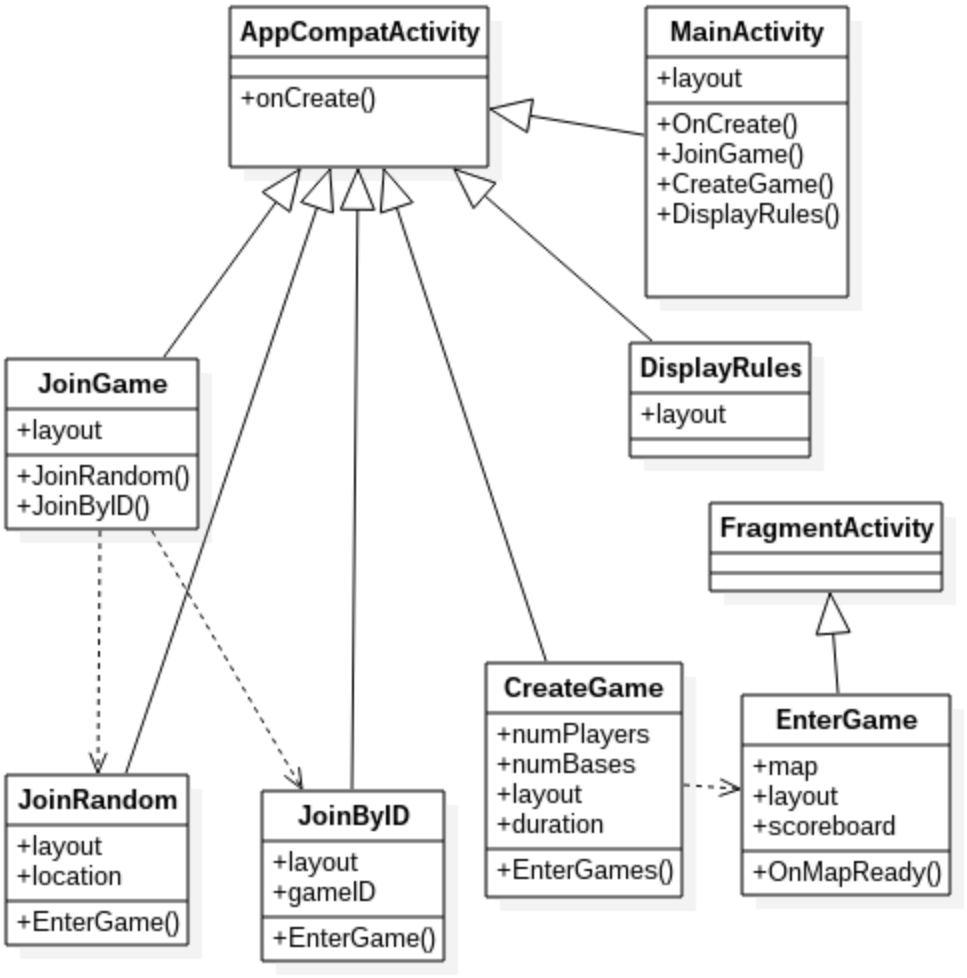
The server takes in vectors and passes them back through theXML-RPC network, and uses the sessionList object to hold all the sessions. The sessionList uses a linked list class LL which has nodes that contain sessions as data, and has a variety of functions for manipulation of the list. A game ‘session’ is a class that through interactions within the game initialization process creates the ‘teams’ and ‘bases’ where the primitive data of game state and details are being held, such as score, number of players, and base locations are. The ‘session’ uses the base distribution algorithm to generate the bases dependent on user input and the number of players. The players are handled within the teams and sessions.



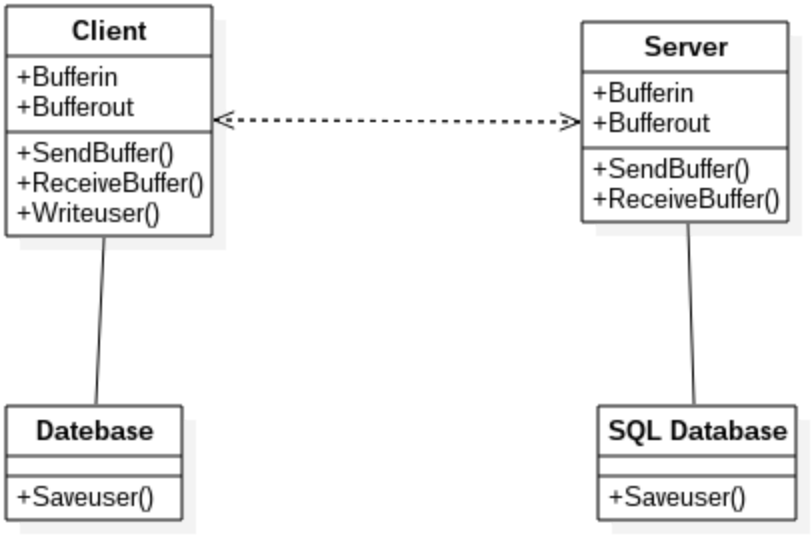
Interface Class Diagram

The Android user interface extends AppCompatActivity in order to be compatible with a higher percentage of Android cellular devices. The MainActivity displays its XML layout when the OnCreate method is called. It determines where the application will go next based on user input from three buttons, which correspond to JoinGame, CreateGame, and DisplayRules. DisplayRules is a basic class that only displays a layout file with the berfrom buttons. From JoinByID, a user-provided game ID will send the application to EnterGame while JoinRandom will give the user a list of available games. The CreateGame class displays its XML layout and retrieves values for number of players, number of bases, game duration, game radius, and time until the game starts. Gameplay extends FragmentActivity and displays the game using Google Maps.

NOTE: In CreateGame after +duration, add +gameDur, +gameRad, +gameStartPos and change EnterGame to Gameplay

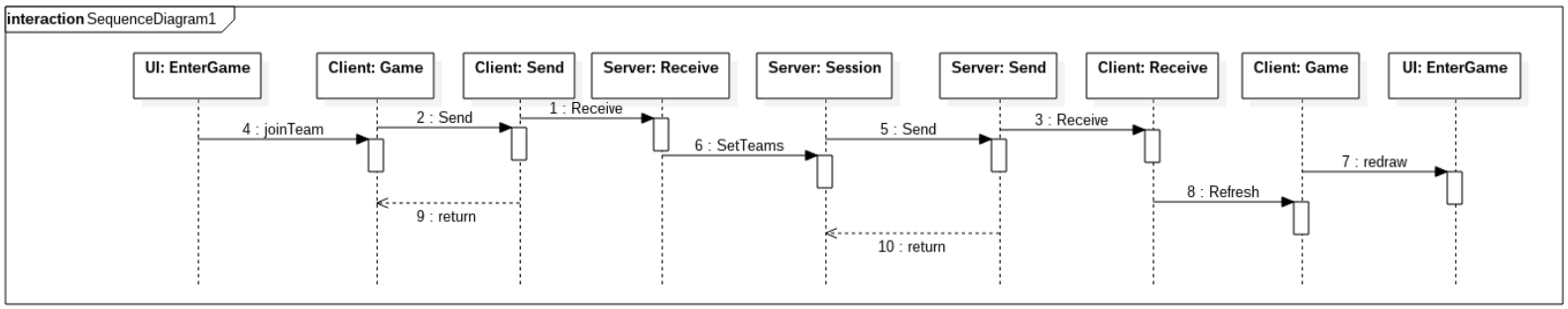


Integration Class Diagram



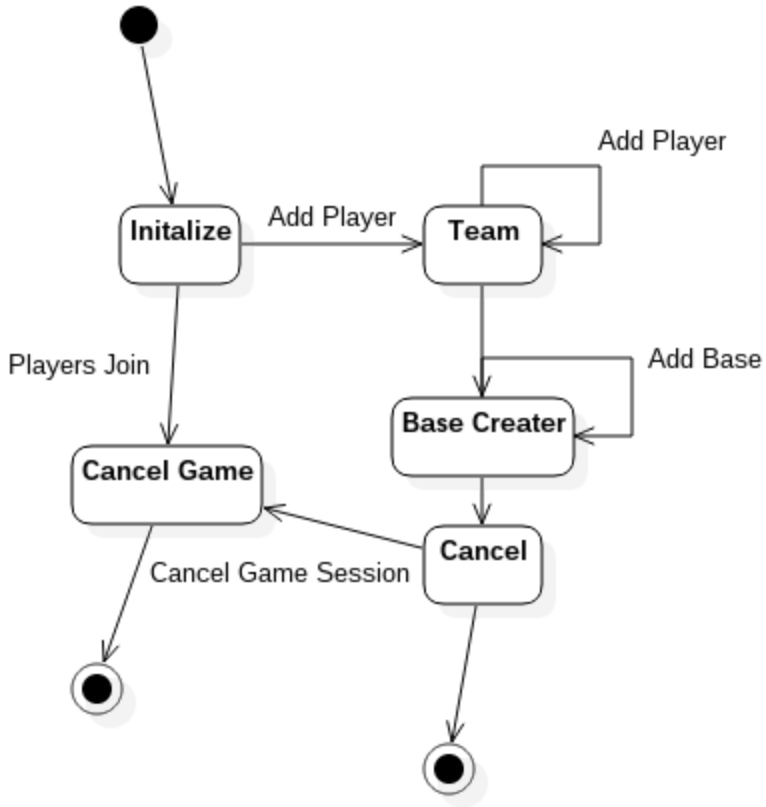
4.2 Sequence Diagram

Join Game



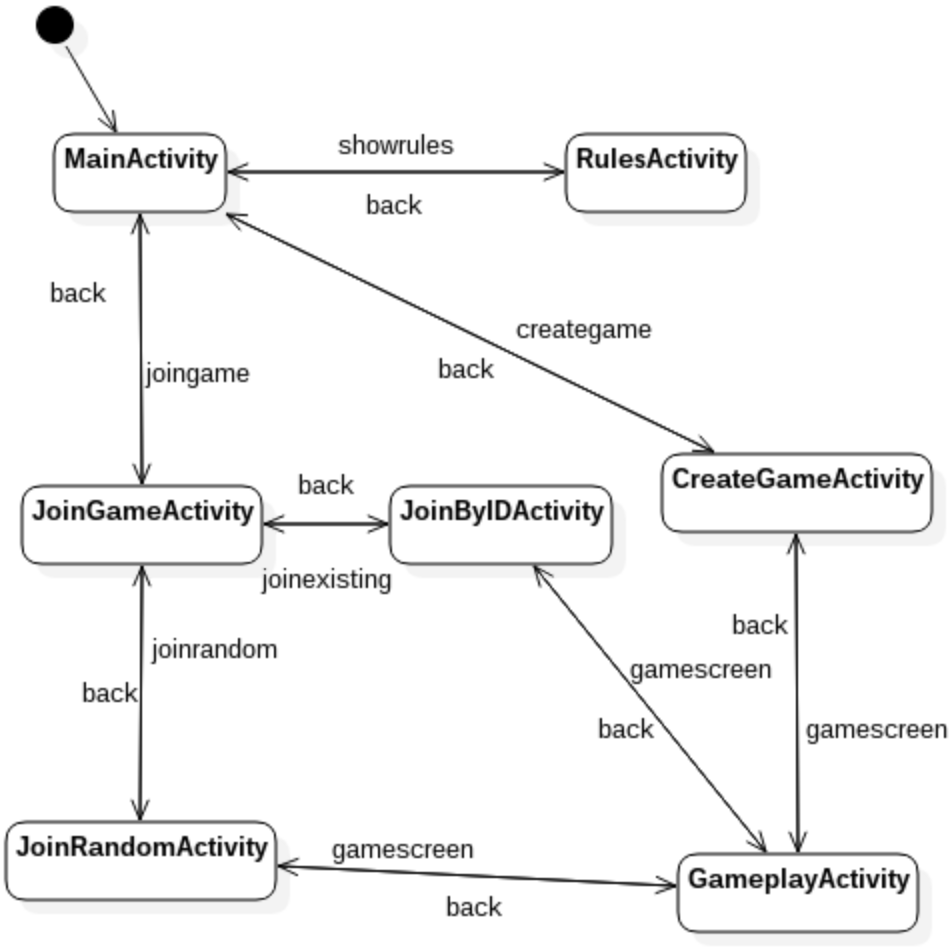
4.3 State Diagram
This diagram shows the lifecycle of object classes in your system

Server State Diagram
So the first state initializes the game, and unless a player isn't added (in which a cancel occurs, notifying players the game won't start), it will move on to the initializing of the teams. Players are added until it reaches the number the administrative player inputted ('maxPlayer'), when from there it will change state to generate the bases. From there it will add bases until it is complete with regard to specification. From there it reaches its steady state where the game itself is finalized, and remains there until game is canceled, through finishing the game.

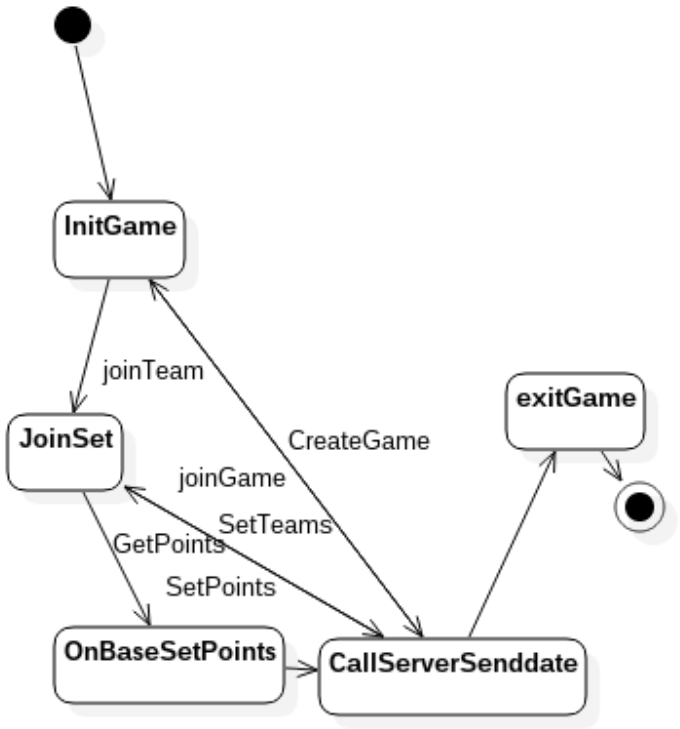


Interface State Diagram

The application opens to the MainActivity, the top-most activity that welcomes users with the main menu. From here, the user can navigate to RulesActivity to learn how to play the game, JoinGameActivity to select a join game option, or CreateGameAcivity to create a brand new game. From JoinGameActivity, the user can either navigate to JoinByIDActivity to join a specific game by its ID, or they can find a random game ID from the JoinRandomActivity. From JoinByIDActivity and CreateGameActivity, the user is taken to GameplayActivity, which is populated with a map and game area where the user will play.



Client State Diagram:

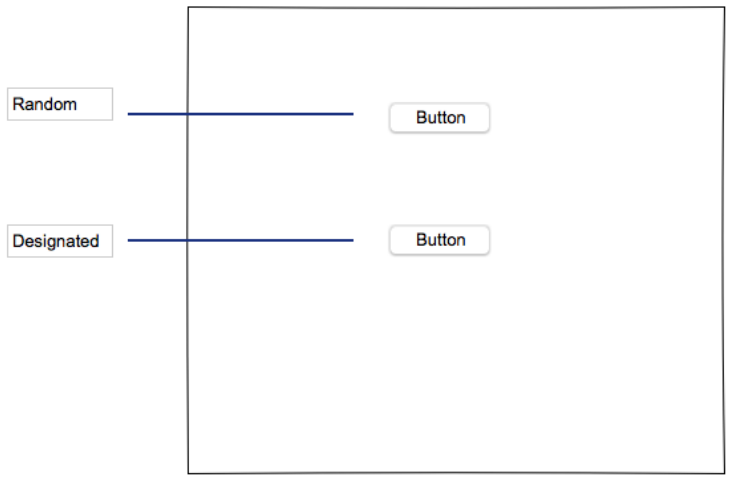


5. User Interface Design

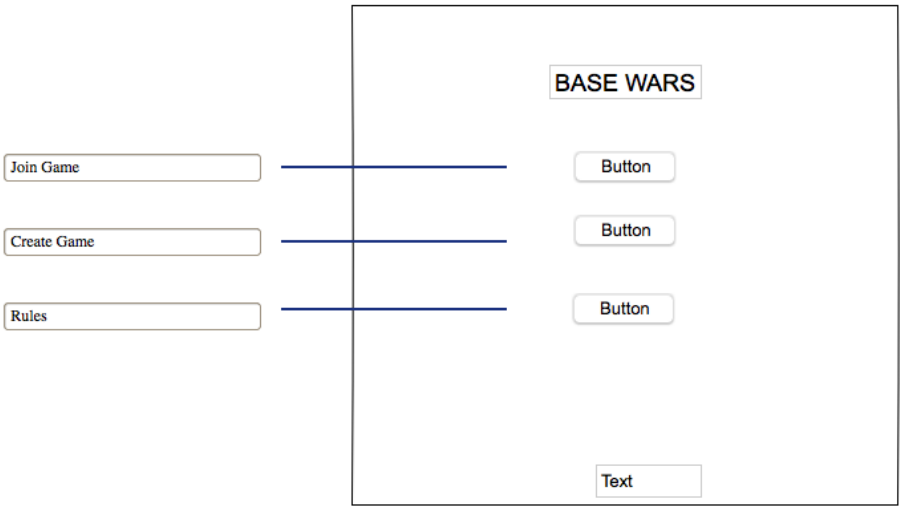
The Android UI interfaces with the Android client, utilizing its core functionality and displaying relevant information received from the server. The Android application will be the only interface that the user has with the project's backend.

The UI is designed with an emphasis on simplifying the user's interaction with the application, in order to create or join game with relative easy. It is designed to be easy to navigate, in order to promote the intent behind our product, which is an easy and quick game that allows the user to interact with their environment. Once the game starts the user is shown on a map of their environment. Below are the specific screens, each with a description of how they operate and are intended to be used by the user.

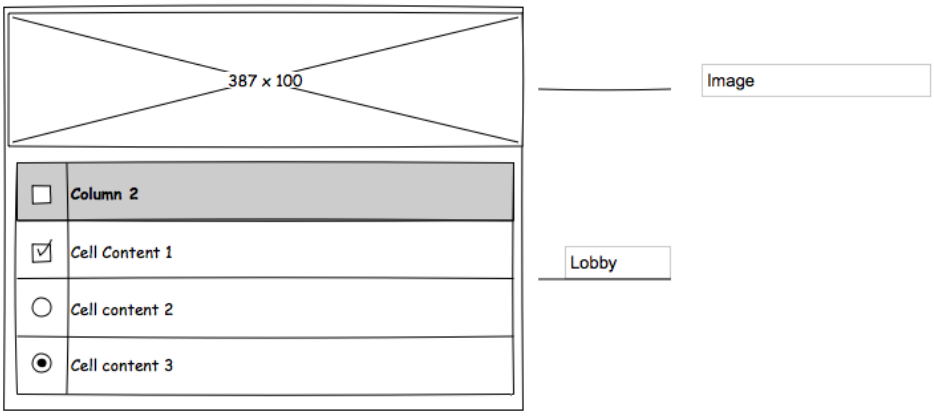
The home screen will display the name of the game in an aesthetic manner. It will provide a link to a rules page comprised of a text file. The two other buttons will direct the user to different screens allowing them to join or create a game.



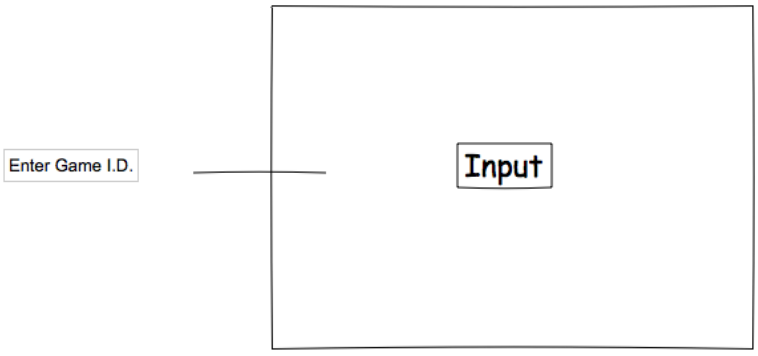
The join game screen will allow the user to indicate if they wish to join a random game, or a designated game.



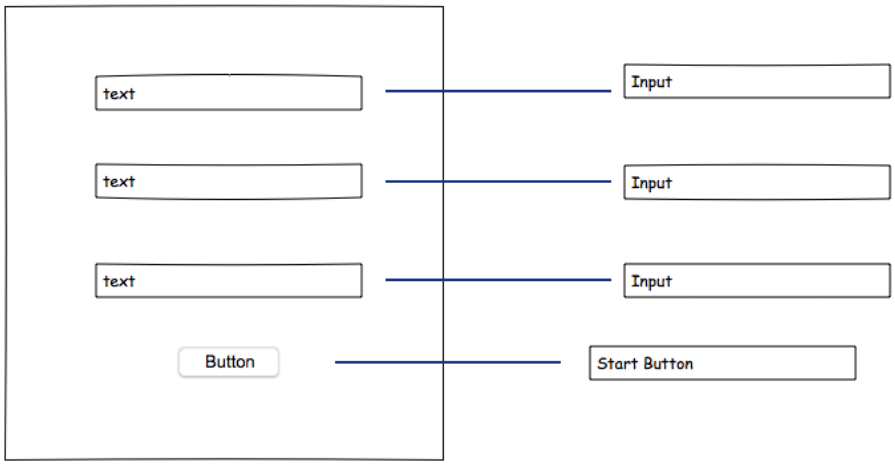
If the user selects to join a random game they will be brought to the join game lobby screen, that lists descriptions of each game lobby they join, which can be either a free-for-all or two team game.



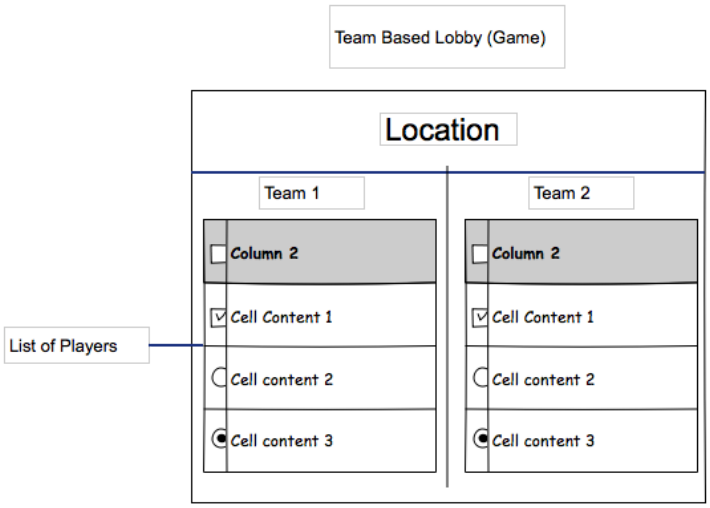
The user is prompted to enter a game I.D. in order to join a specific game if they chose designated game at the join game screen.



If the user selects to create a game from the home screen they will be directed to a screen that will allow to enter three inputs describing their game, which are to be determined later. A button will allow them to create their game once they have entered the three inputs.



This screen shows the lobby of the two team game once it has been chosen by the user, until the game star



This screen shows the lobby of the free-for-all game once it has been chosen by the user, until the game starts.