Technical Design Document

Swarm

1 Datastore models

* each model will have an ID

a. Planet
    i. position - Dictionary
        1. x - Int
        2. y - Int
    ii. capacity - Int
    iii. owner - String
    iv. population - Int
b. Game
    i. players - Sting List
    ii. planets - Planet List
c. Player
    i. username - String
    ii. hashed_password - String
    iii. salt - String
    iv. stats - Dictionary
        1. wins - Int
        2. loses - Int

2 Client Persistent Data

∅

3 Protocols / APIs

URLS:

a. /register

   POST -> {name, pwd}

   Response -> {error} or {success}

b. /login

   POST -> {name, pwd}

   Response -> {error} or {success}

c. /logout

   GET -> Redirect to Home

d. /search - begins search for player

   POST -> {game_name, map_id}

   Response -> {error} or {success}

e. /game/{id}

f. /practice

g. /play

h. /animation-demo

Socket.IO Events:

Client -> Server

playerJoin -> {name, color}

ready -> {name}

surrender -> {name}

sendShips -> {planet_src, planet_dest, quantity}

Server -> Client

Search -> {name}

Joint Game -> {name}

disconnect -> {}

Send Ships -> {planet_src, planet_dest, quantity}

4 In-memory Data & Memcache

We are using mongodb, which already memcaches common queries and hence there should not be a need to memcache anything for the database on the server. We will memcache common views that are possibly requested a lot of times but do not change. For example, the index page for registering and logging in. We could also memcache the searching for player page and the map selection page since those pages should only change if we add more maps to the game.