# Technical Document for Mobile Controller Game

Datastore Models:

- User
  - email (string)
  - games_played (game objects)
  - friends (user objects)
  - status (integer/boolean indicating whether in a room/session or not)
  - session_id (integer indicating the room/session user is in if any)
- Game
  - id
  - players (user objects or email)
  - game played (integer/enum indicating which game was played)
  - winner (user object or email)
- Room/Session
  - id
  - games (game objects. List of games played in this room/session)
  - creator (user object)
  - status (integer/boolean indicating whether the room/session is currently still active or has finished)
  - game_status (integer/boolean indicating whether the room/session is currently in the middle of a game or not)

Data to be stored on client side:

- User local settings (specific adjustments like control configuration for left handed vs right handed)

Protocols/APIs

- For signing in user:
  - URL: /SignIn
  - Method: POST
  - Arguments: email, password
  - Success: Redirect to /Home.
    Error: Stay at /SignIn and return error message (JSON)
- For signing out:
  - URL: /SignOut
  - Method: GET
  - Arguments: None
  - Success: Redirect to /SignIn
    Error: Return error message (JSON)
- Join game room based on ID:
  - URL: /Join
  - Method: GET

- Arguments: RoomID
    - Success: Redirect to /Room?[Id]
      Error: Return error message (JSON)
- Exit game room:
    - URL: /Exit
    - Method: GET
    - Arguments: RoomID
    - Success: Redirect to /Home
      Error: Return error message (JSON)
- Create game room:
    - URL: /Create
    - Method: GET
    - Arguments: RoomName, list of users allowed to join
    - Success: Redirect to /Room?[Id]
      Error: Return error message (JSON)
- Indicate readiness to play:
    - URL: /Ready
    - Method: POST
    - Arguments: SessionToken
    - Success: Return success message (JSON)
      Error: Return error message (JSON)
- Start game:
    - URL: /StartGame
    - Method: GET
    - Arguments: None
    - Success: Redirect to /Game?[Id]
      Error: Return error message (JSON)
- Game quit/exit:
    - URL: /EndGame
    - Method: GET
    - Arguments: None
    - Success: Redirect to /Room?[Id]
      Error: return error message (JSON)
- Test Suite:
    - URL: /testsuite
    - Method: GET
    - Arguments: None
    - Success: Runs test in test suite
    - Error: return error message (JSON)
- Controls for Game:
    - URL: game/test/control
    - Method: POST
    - Arguments: button

    - Success: Balloon size is increased or decreased
    - Error: return error message (JSON)
- Update Game:
    - URL: game/test/update
    - Method: GET
    - Arguments: Newest button to press in sequence
    - Success: Nothing happens
    - Error: return error message (JSON)
- Generate Sequence
    - URL: game/test/sequence
    - Method: GET
    - Arguments: None
    - Success: Array of random ints are returned
    - Error: return error message (JSON)

<u>Manual Test Plan for UI</u>
Phone View:

You'll see 4 arrows, an A-button, a B-button, and a balloon that will get bigger and smaller depending on the buttons pressed. The button you pressed should correspond to the button that's prompted on the screen. If it says "SHAKE IT", use the phone's accelerometer function to be the button press.

<u>Javascript Testing</u>
Append "/testsuite" to the end of the URL to run the qunit tests. These test the Javascript library we wrote