**Test Case Spec (QA)**

Test Strategy/Approach

Front end --top down

Back end --unit testing

Top Down Integration Testing:

Each time there was added functionality to the game (after each sprint) the game was thoroughly tested. After each term the test cases grew in number as new and old functions/ test cases were used to ensure that any changes had no side effects.

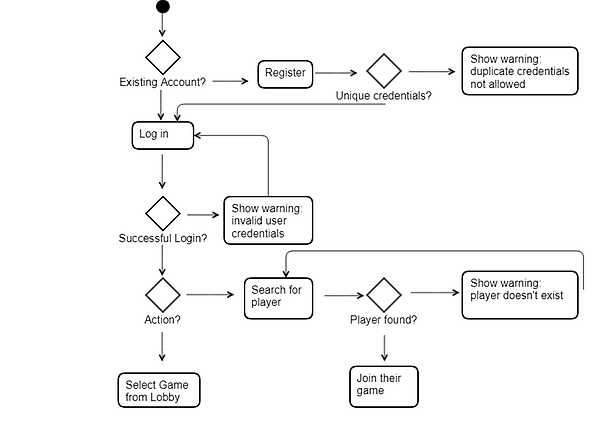
The regression testing was done manually via postman/git.

Smoke testing was done alongside development by developers prior to handing off modified version to QA.

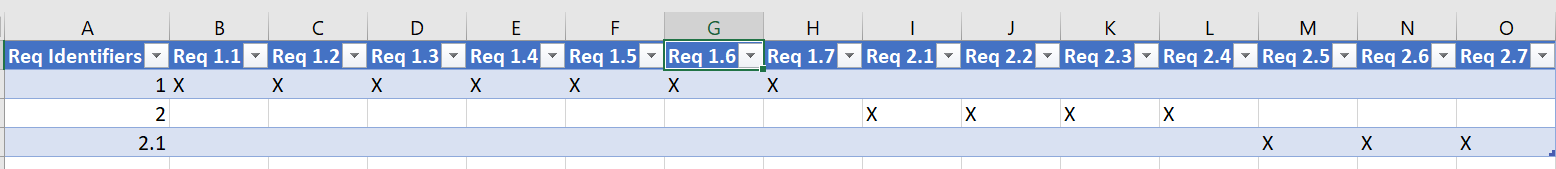
Test Plan

Test case design was scenario-based testing, focusing on how user interacts with the system. Test cases were created to cover most use cases.

QA would interact with user interface as if they were a user without any knowledge of back end, sometimes providing possible/likely, invalid input to test whether it would crash the program.



[enter use case for front end]

Tracability Matrix

Requirements:

1.1: user login

1.2: user creation / register

1.3: create game

1.4: join game by game name or by username of user in the game

1.5: Google maps instance

1.6: marker for each user in the game instance

1.7: marker position updating according to each user's real geolocation

2.1: zones are instantly captured by the player on it, if both players are on it, owner doesn't change

2.2: 1 point per zone owned per 5 seconds (parameterized in GameConfiguration.js)

2.3: game has two end conditions:

I. first player who reaches 100 points wins (no ties)

II. timeout after 10 mins of game creation; winner is the person with the most points (no ties)

2.4: score, time, zones, zone owners, marker for self, & marker for enemy is displayed to user

2.5: created directories for project deliverables

2.6: updated frontend/app.json so it's build-able to android at least

Test Cases/Results

Back End Functions:

1. router.get('/', function (req, res) -- run function to see if all users signed into game are returned [successful]
2. router.post('/register', function (req, res) -- create several users and test that each user is made an account [successful]

--created duplicate usernames, emails, and passwords to see if error message returned [error]

Error: when several users have no email provided, duplicate emails are created

3. router.post('/login', function(req, res) -- check that registered users are verified successfully, also test usernames that aren’t registered to see if error ensues [successful]

4. router.get('/:id', function (req, res) -- check that function returns registered users and notifies when a user doesn’t exist [successful]

Front End Functions:

1. Response--checks for duplicate key [successful]
2. if a game becomes empty (no users), remove the document from the db [successful]

# If a user joins a game, but is already in a game, send him a notification event about this [successful]

# Google maps won’t display [error]

1. Trying to use create game to rejoin or using join room name throws the syntax error if the room name already exists [error]

# React-native-maps slight movement of the map [error]

# Don't allow troops to be below 0 [successful]

\*Since last sprint, all outstanding bugs have been resolved