**TapWater**

**System Testing Document**

**Authors:**

Cody Rogers

David Fontana

Jonathan Hooper

Kon-Kon Chao

**Table of Contents**

Introduction 2

Test Items 3

Features To Be Tested 6

Testing Approach 7

Pass/Fail Criteria 8

Testing Environment 9

**Introduction**

This document specifies the test plan the TapWater System. The test plan outlines the procedures and test cases to be used to test the requirements outline in the Systems Requirements Specification Document.

The TapWater system consists of 3 major components: A server, an iOS application, and an Android application. Each subsystem will require a slightly different testing strategy. The approach for the iOS and Android application will be similar because they are structured the same.

The features that need to be tested involve logging drinks, viewing drink history, scheduling drink notifications, user authentication, and synchronization. All of the features will need to be tested on the mobile application. Only user authentication and synchronization need to be tested on the Rails server since those are the only features implemented on that platform.

**Test Items**

**Log Drinks**

1.1: New Drink

**Input:** Create new drink request with ‘drink’ option

**Output:** New Drink with ‘drink’ option, unique unique identifier, and current date for the drink date

**Environment:** iOS/Android device running the TapWater mobile application

1.2 New Glass

**Input:** Create new drink request with ‘glass’ option

**Output:** New Drink with ‘glass’ option, a unique unique identifier, and current date for the drink date

**Environment:** iOS/Android device running the TapWater mobile application

1.3 New Bottle

**Input:** Create new drink request with ‘bottle’ option

**Output:** New Drink with ‘bottle’ option, unique unique identifier, and current date for the drink date

**Environment:** iOS/Android device running the TapWater mobile application

1.4 Drink Save Time

**Input:** Create new drink request

**Output:** A new Drink with the ability to prepare another one before the new drink button animation has finished

**Environment:** iOS/Android device running the TapWater mobile application

**Drink History**

2.1 View Drink History

**Input:** View Drink History request

**Output:** A table of drinks with drink dates and drink categories

**Environment:** iOS/Android device running the TapWater mobile application

**Drink Notifications**

3.1 Schedule Drink Notifications

**Input:** Set 5 drink goal with 12:00 PM notification start time and 6:00 PM notification end time

**Output:** Drink notification responses at 1:00 PM, 2:00 PM, 3:00 PM, 4:00 PM, and 5:00PM

**Environment:** iOS/Android device running the TapWater mobile application

**User Registration**

4.1 Server Registration

**Input:** JSON request with params username=’unique\_user’, password=’123abc’, password\_confirmation=’123abc’

**Output:** JSON response with params username=’unique\_user’ and a device token

**Environment:** TapWater server running on a remote host

4.2 User Uniqueness

**Input:** Sequential JSON request with params username=’duplicate \_user’, password=’123abc’, password\_confirmation=’123abc’

**Output:** JSON response with params username=’duplicate\_user’ and a device token, followed by a response indicating that the username, ‘duplicate\_user’ is already taken.

**Environment:** TapWater server running on a remote host

4.3 Password Confirmation

**Input:** Sequential JSON request with params username=’password\_confirm \_user’, password=’123abc’, password\_confirmation=’abc123’

**Output:** Response indicating that he password and password\_confirmation do not match

**Environment:** TapWater server running on a remote host

4.3 In-App Registration

**Input:** New user request with values ‘mobile\_user’ in the username field, and ‘123abc’ in the password and password confirmation fields

**Output:** A response indicating that the user has been creating

**Environment:** iOS/Android device running the TapWater mobile application

**Log In**

5.1 Valid Server Login

**Input:** JSON request with username and password parameters that are known to be valid

**Output:** JSON response with the username and a device token

**Environment:** TapWater server running on a remote host

5.2 Invalid Server Login

**Input:** JSON request username password parameters that are known to be invalid

**Output:** JSON response indicating that the username and password do not match

**Environment:** TapWater server running on a remote host

5.3 Valid In-App Login

**Input:** Log in request with a username and password that are known to be valid

**Output:** Response from the application indicating the user has logged in

**Environment:** iOS/Android device running the TapWater mobile application

5.4 Invalid In-App Login

**Input:** Log in request with a username and password that are known to be invalid

**Output:** Response from the application indicating the user failed to log in

**Environment:** iOS/Android device running the TapWater mobile application

**Log Out**

6.1 Log Out Request

**Input:** Log out request from the application from the logout button

**Output:** Response from the application indicating the user has logged out

**Environment:** iOS/Android device running the TapWater mobile application

**Synchronization**

7.1 Synchronization

**Input:** A drink is create on an authenticated device and synchronized

**Output:** The drink should appear on another device authenticated by the same user

**Environment:** 2 iOS/Android devices running the TapWater mobile application

**Features To Be Tested**

**Testing Approach**

**Pass/Fail Criteria**

**Testing Environment**