TapWater

System Testing Document

Authors:
Cody Rogers
David Fontana
Jonathan Hooper
Kon-Kon Chao

Table of Contents

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

The following features of the TapWater system are to be systematically tested:

Log Drinks

Logging drinks refers to the process of creating drinks from the main TapWater screen. When the user taps one of the 3 buttons on the screen (“New Drink”, “New Glass”, “New Bottle”) the app will create and save a drink of the given category.

Drink History

The drink history refers to the log of drinks the user has created during his or her use of the application. The user should be able to go navigate to the drink history screen and see a table of their drinks with the category the drink belongs to (“Drink”, “Glass”, “Bottle) as well as the date it was created.

Drink Notifications

Drink notifications are reminders the user can create to remind them to drink at different points during the day. The user sets a drink goal and a start and end time. The application will then schedule the reminders to go off at even intervals throughout the day between the start and end time. The number of reminders corresponds to the number of drinks the user has left to reach their goal.

User Registration

User registration refers to the process of creating an account that is saved on the TapWater server. Users must create an account in order to take advantage of features provided by the server, namely synchronization. To create an account, the user navigates to the account creation screen and enters the information for the account they wish to create.

Log In

Logging in is the process of associating a device with a user account. When the device is associated with an account, it will allow the user to synchronize the data on that device with other devices they are logged into. To log in, the user will navigate to the log in screen and enter their account information.

Log Out

Logging out is the process of removing the association between a user and a device. When the user logs out they will no longer be able to synchronize data between different devices. To log out, the user presses the “Log Out” button.

Synchronization

Synchronization is the process of synchronizing drinks between a user’s devices. To synchronize a drink, first create the drink on one device. Then synchronize the device with the server. This will store the drink on the server. After synchronization with the server is complete, synchronize the second device with the server to receive the drink. After this process is complete, the drink should appear in the drink history on both devices.

Testing Approach

The testing approach depends on the environment being tested. The two environments are the Rails Server and the Mobile Application. More information on these environments can be found in the *Testing Environments* section. In both environments we are using black box testing.

Rails Server

When testing the rails server, we will use at tool that will allow us to send JSON requests with the parameters we need to test with. A good suggestion is the cURL command line tool with the –d flag.

Mobile Application

Testing the mobile application will involve human testing. A QA engineer will run through the test cases with a device in front of him. For synchronization, two devices will be necessary.

Pass/Fail Criteria

After test procedures are followed and the response is received, the QA engineer will determine if the test has passed or failed. A test has passed if the response from the application matches the expected response from the given test case.

The issue of whether or not a test failing once and then passing again will be handled on a test case by test case basis. Some tests may fail due to issues out of the control of the developers, such as network connectivity problems. Test cases that introduce a situation like this, which are primarily the test cases that test the behavior of the server environment, should be handled with that in mind.

Testing Environment

The application involves 3 testing environments: the Rails server, the iOS application, and the Android application. For convenience, the iOS and Android environments have been rolled into one iOS/Android Application or Mobile Application environments. That is because these environments have the same requirements and features, so the test cases for them are the same.

Rails Server Environment

The rails environment refers to the Rails Server running on Heroku. The rails server interfaces with the rest of the system via a JSON API.

Mobile Application Environment

The mobile application refers to the TapWater application running on either an iOS or Android devices. The mobile applications should be a feature parody of each other so the test cases are the same, but each environment will need to be tested individually.