

System Test Documentation

# Magic: The Gathering™ - Unofficial Collector’s Compendium

Icon

Description automatically generated with medium confidence

Version: 1.0.0

Written: 04/27/2023

[GitHub Repository](https://github.com/CSmith1998/MagicTheGathering-UnofficialCollectorsCompendium)

Pittsburgh Technical College

REVISIONS

Within this document, being the first in its series, no revisions to previous documents or plans are present. Instead, this version (v) of the System Test Documentation (STD) for the project of the working title “Magic: The Gathering™ - Unofficial Collector’s Compendium” shall set forth the first iteration of this documentation to act as a basis for any future changes or revisions.

# Revision Chart

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Description of Version | Date Completed |
| Draft | Caine Smith | Initial draft. Created for distribution and comment review. | 04/27/2023 |

TABLE OF CONTENTS

# Index

1. Introduction 3
   1. System Overview 3
   2. Test Approach 3
2. Test Plan 4
   1. Features to be Tested 4
   2. Features not to be Tested 5
   3. Testing Tools and Environment 5
3. System / Software Test Cases 6
4. Additional Material 9
5. Appendix A. Test Logs 10
   1. A.1.1 Log for Test 1.1 10
      1. Test Results 10
      2. Incident Report 10

INTRODUCTION

# System Overview

This project will make use of a combination of testing tools and utilities to ensure that key functions and features all work as anticipated for the sake of delivering a consistent and reliable product. Of the products to be used, the three (3) main testing utilities will be JUnit (Java Testing), XUnit (C# Testing), and Selenium (Automated Browser Testing).

Due to the wide range of languages in use within this application, it is important to keep in mind when to use each individual tool. JUnit should be utilized primarily when testing the custom web API service, and a combination of Selenium and XUnit should be used for testing functions and behaviors of the web application interface.

# Test Approach

This project shall follow a Test Driven Design approach. For each notable section, a minimum of three (3) probing, meaningful tests will be written to ensure that key features and behaviors function exactly as intended. Any deviation should be noted with the creation of an issue within the project’s KanBan and handled as soon as possible as failure to do so could result in further problems as development progresses.

To ensure that there is a minimum of bugs in the system, tests should be run regularly. At minimum, 3 tests should be run during every sprint. One at the start, one in the middle, and one at the end. Further tests are encouraged, but not required.

TEST PLAN

# Features to be Tested

Important features to be tested are as follows:

* 1.1 GET Handler (RESTful Service)
* 1.2 POST Handler (RESTful Service)
* 1.3 PUT Handler (RESTful Service)
* 1.4 DELETE Handler (RESTful Service)
* 2.1 User Registration
* 2.2 User Login
* 2.3 Role Restricts Section of App
* 3.1 Add New Card to Collection
* 3.2 Search Card in Collection
* 3.3 Edit Card in Collection
* 3.4 Delete Card in Collection
* 3.5 Details of Card in Collection
* 4.1 Manage User Roles
* 4.2 View User Login Data
* 4.3 Delete User
* 4.4 View Use Statistics

# Features not to be Tested

The following features will not be tested with testing software and will thus be ignored or bypassed during testing.

* Captcha (Must test manually or using special bypass)
* Email Verification (Must test manually)
* Home Button
* User Logout
* Mobile Functionality
* 3rd Party API (No direct tests. Assumed to work.)
* CSS Elements

# Testing Tools and Environment

Testing will take place within Java, ASP.NET MVC, and C# environments and, as such, will utilize the tools JUnit, XUnit and Selenium to ensure thorough and effective testing. Selenium tests will be conducted through Mozilla Firefox using the Gecko Driver due to the inconsistency of testing results that are achieved through Chrome.

Testers are required to keep previous tests available for use and should periodically run old tests to ensure previous functionality has not been compromised.

SOFTWARE / SYSTEM TEST CASES

| ID | Description | Steps | Expected |
| --- | --- | --- | --- |
| 1.1 | Send a GET request to Web Service and receive correct information in return | 1. Construct request with or without search parameters.  2. Send request to Web Service  3. Check response for Status Code 200  4. Assert that expected results match actual. | API will return with a status code of 200 and, depending on input parameters, no cards, one card, or many cards are returned. |
| 1.2 | Send a POST request to the Web Service and receive an OK Status code. | 1. Construct request.  2. Send request to Web Service  3. Assert that response for Status Code 200 | API will return with a status code of 200. |
| 1.3 | Send a PUT request to the Web Service and verify change. | 1. Construct request  2. Send request to Web Service  3. Check response for Status Code 200  4. Use GET request to assert that the record has been changed | API will return with a status code of 200, and searching for previous record using GET shows updated results. |
| 1.4 | Send a DELETE request to the Web Service and verify change. | 1. Construct request  2. Send request to Web Service  3. Check response for Status Code 200  4. Use GET request to assert that the record has been deleted | API will return with a status code of 200, and searching for previous record using GET will retrieve no results. |
| 2.1 | Registration with new email address succeeds. | 1. Enter Email  2. Enter Password  3. Select and Answer Security Questions (If present)  4. Click Register | Redirect to Email Confirmation page |
| 2.2 | Login with correct email and password succeeds | 1. Enter Email  2. Enter Password  3. Bypass captcha (if present)  4. Click Login | Redirect to 2-factor authentication, if present. Otherwise, redirect to Landing Page. |
| 2.3 | User’s role affects availability of certain features | 1. Login with a user  2.1. Assert that user cannot see Admin links  2.2. Assert that an admin can see the link. | Administrative tool links are not present.  Tools ARE present for administrators. |
| 3.1 | Add a new card to user’s collection | 1. Click Collection  2. Click Create New  3. Input Card Name  4. Click Search  5. Input additional details  6. Click Submit  7. Assert that card appears in collection | On submit, user is redirected to either the card details, or their collection.  New card appears within the collection. |
| 3.2 | Search for a Card in a user’s Collection | 1. Click Collection  2. Input Card Name or other details  3. Click Search  4. Assert that card(s) exists | Card that is known to be within the collection appears within the list as anticipated. Vague searches show only relevant cards. |
| 3.3 | Edit a Card in a user’s Collection | 1. Select a Card from Collection  2. Click Edit  3. Change location or other detail  4. Click Update  5. Search for card by name  6. Assert that card details were updated. | Card details change to reflect edit that was made. |
| 3.4 | Delete a Card from the user’s Collection | 1. Select a Card from Collection  2. Click Delete  3. Confirm Delete action  4. Search for card by name  5. Assert that card is not in collection | Card that was previously within the collection is no longer present. |
| 3.5 | Retrieve details of a Card from a user’s Collection | 1. Select a Card from Collection  2. Assert that Card contains one or more instances of the card in one or more locations.  3. Click Details on one instance  4. Assert that pertinent details are present. | Cards are appropriately nested and drilling down to a specific entry gives the appropriate information. |
| 4.1 | Administrator can modify a user’s role. | 1. Login as an Admin  2. Click Admin Tools  3. Click Manage Users  4. Select User  5. Change Role.  6. Check User’s Details.  7. Assert that change appeared. | User’s role is updated successfully and displays the new role in the menu. |
| 4.2 | Administrator can view a user’s login data. | 1. Click Admin Tools  2. Click Login Data  3. Select User  4. Assert that Login/Logout data exists. | Login/Logout data exists and shows the appropriate information / details. |
| 4.3 | Administrator can delete a user from the service | 1. Click Admin Tools  2. Click Manage Users  3. Select User  4. Click Delete User  5. Confirm Delete Action  6. Assert that user is no longer in user list. | User is no longer listed in the list of users after delete action is performed. |
| 4.4 | Administrator can view Use Statistics | 1. Click Admin Tools  2. Click Usage Statistics  3. Assert that appropriate details exist. | Usage statistics shows the expected information. |

ADDITIONAL MATERIAL

APPENDIX A. TEST LOGS

This section exists as a placeholder for future tests. As production on the project has not yet begun, no tests have been written or performed, so no logs exist. This section will be revised in future reports to include the required documentation.

# A.1.1 Log for Test 1.1

## Test Results

## Incident Report