Software Test Document

Card Czar Android App

CMSC 495

Group 2 Final Project

Kenneth Mikkalson

Alton Hinton

Shawn Henson

Sarah Holley

Tara Lawson

Richard Wysong

Table of Contents

[1.0 Test plan identifier 4](#_Toc429327434)

[2.0 Introduction 4](#_Toc429327435)

[2.1 Objectives 4](#_Toc429327436)

[2.2 Background 4](#_Toc429327437)

[2.3 Scope 4](#_Toc429327438)

[2.3.1 In Scope 4](#_Toc429327439)

[2.3.2 Out of Scope 6](#_Toc429327440)

[2.4 References 6](#_Toc429327441)

[2.5 Roles and Responsibilities 6](#_Toc429327442)

[2.5.1 Developer 6](#_Toc429327443)

[2.5.2 Test Team Member 6](#_Toc429327444)

[2.5.3 Test Lead 6](#_Toc429327445)

[2.6 Definitions 7](#_Toc429327446)

[2.7 Suspension Criteria and Resumption 7](#_Toc429327447)

[2.8 Pass/Fail criteria 7](#_Toc429327448)

[2.9 Testing Resources 7](#_Toc429327449)

[3.0 Methodology 8](#_Toc429327450)

[3.1 Overview 8](#_Toc429327451)

[3.2 Unit Testing 8](#_Toc429327452)

[3.3 Integration Testing 8](#_Toc429327453)

[3.4 Final Delivery Testing 9](#_Toc429327454)

[3.5 Bug Regression 9](#_Toc429327455)

[3.6 Test Complete Criteria 9](#_Toc429327456)

[3.7 Test Deliverables 9](#_Toc429327457)

[4.0 Bug Tracking & Reporting 10](#_Toc429327458)

[4.1 Bug Reporting 10](#_Toc429327459)

[4.2Assigning Labels 10](#_Toc429327460)

[5. Approvals 11](#_Toc429327461)

**List of Figures**

[Figure 1: GitHub Issue Form 10](#_Toc429311370)

**List of Appendix**

Appendix A

Appendix B

# 1.0 Test plan identifier

CMS-0214

# 2.0 Introduction

## 2.1 Objectives

The system test plan for the Card Czar Android Application will support the following objectives:

1. Identify responsible persons for quality assurance, test management and testing tasks

2. Define the test plan and testing methods involved

3. Identify resource information for test plan development

## 2.2 Background

The Card Czar Android application is a spin-off of the popular board game Apples to Apples. The purpose of the software test plan is to identify key aspects of software verification and requirements validation.

The Card Czar is an android application multiplayer card game that can deliver hours of fun.  It allows a user to create a new game room and invite friends to play with them. For a game to begin there must be at least 3 players. Once a game is started each player is dealt at random a hand of response cards that will contain a noun (or response) on them. After all players are given the response cards a dealer will be randomly chosen to start the game. The dealer (card baron) will draw a bait card that consists of an adjective (or other text to be responded to). Each player then selects from their cards a noun that they think the dealer will choose as the best response to the question. Once all responses are received the dealer (Card Baron) then picks the answer they like the best and the player whose response it was gets a point and becomes the dealer (Card Baron) for the next round. This continues until a player has reached 5 points and becomes the winner (the Card Czar).

Maximum players: 6

Minimum players: 3

Bait cards: (Adjective/Question Options)

Response cards: (Noun/Answer Options)

## 2.3 Scope

### 2.3.1 In Scope

The Card Czar Test Plan defines the unit, integration and acceptance testing. The test plan will include the following:

1. Testing of all interfaces between Android application modules

2. Testing of all quality requirements

3. Security testing - See Appendix B for addition information

SQL Injection

Use of SSL

4. Total system functionality testing

5. Testing of Input and Output:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Description** | **Trigger Action** | **Input/Output Type** |
| 1 | User selects to host a game | Press hostButton on phone | Phone |
| 2 | Host (AKA ActivePlayer) starts room - enters room name and host name | Press roomStartButton on phone | Phone |
| 3 | Room and Host name record created | Room and Host name passed after roomStartbutton is press. | Web->DB  Tables gamestate, users,questions, answers inserted. |
| 4 | Phone screen refreshes with Room and Users | refresh from DB records for users and room | Phone |
| 5 | User selects to join a room | joinButton on phone | Phone |
| 6 | User joins a room - enters room name and user name | Press userStartbutton | Phone |
| 7 | User record created | Room name and user name record inserted | Web->DB  Tables users updated |
| 8 | ActivePlayer refreshes phone display until sufficient users have joined | Press refreshUsersButton to retrieve data from database | Web->DB query |
| 9 | Phone display refreshes with users | refresh from DB query | Phone |
| 10 | ActivePlayer starts game | ActivePlayer presses gameStartButton | Phone |
| 11 | DB gamestate updated prepares questions and answers for game | Gamestate change passed from phone after gameStartButton is pressed | Web->DB |
| 12 | Phone screen updates as appropriate - ActivePlayer sees questions, Users see possible answers | refresh from DB records containing gamestate, users, questions, answers | Phone |
| 13 | ActivePlayer selects adjective | Select from choices on phone | Phone |
| 14 | User1 - select noun | Select from choices on phone | Phone |
| 15 | User2 - select noun | Select from choices on phone | Phone |
| 16 | User3 - select noun | Select from choices on phone | Phone |
| 17 | DB records are updated after ActivePlayer and User selections | User/ ActivePlayer input on phone | Web->DB |
| 18 | Once all answers are in ActivePlayer picks winner | Select from choices on phone | Phone |
| 19 | Gamestate updated and round winner updated | Winner chosen by ActivePlayer | Web->DB |
| 20 | Round winner becomes ActivePlayer | Refresh from DB | Phone |

### 2.3.2 Out of Scope

The following will not be tested during the testing phase:

1. Physical requirements of database systems. To include size or throughput

2. Physical aspects of any Android mobile device

3. Performance aspect of database queries

## 2.4 References

1. Project Plan, Card Czar Android Application

2. IEEE Software Engineering Standards Committee, “IEEE Std 829 1998, IEEE Recommended Practice for Software Test Documentation”, 06 August 2002.

## 2.5 Roles and Responsibilities

### 2.5.1 Developer

The software development team will be responsible for development of all use case scenarios, which the testing team will use to develop the test case scenarios. The development team will also conduct all necessary unit tests prior to integration testing. A minimum of one developer from each development team will be required to be available during integration and acceptance testing.

### 2.5.2 Test Team Member

The test team members will be responsible for development of test case scenarios, regression, and acceptance testing. The integration team will perform the necessary tests based on the overall project schedule. Any software deficiencies identified during testing will be entered as bug reports in GitHub. Bug Reports related to failed testing items will be closed by a test team member after they have passed a retest.

### 2.5.3 Test Lead

Test Lead will be overall responsible for software testing events post software delivery in weeks 5-7. Following an integration period the test team will perform a regression test to test new functionality as well as any bug fixes that are ready for retest. At the end of the final integration period the test team will perform final testing and prepare for requirements verification testing.

## 2.6 Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Bug | A software deficiency in a released software version |
| Bug report | Tracking mechanism for annotating software bugs and tracking through the correction process. |
| Unit test | Test performed by the software developers prior to software release. |
| Pass/Fail criteria | Rules used to determine whether a software item or feature passes a test. |
| GitHub | External Configuration Management site |
| SAT | Satisfactory |

## 2.7 Suspension Criteria and Resumption

If at any point during integration testing or system level testing that a software bug is preventing the testing from continuing any further than the test will be suspended. If a test is suspended the appropriate bug reports will be generated as well as a testing suspension notice will be routed to the entire team to develop a fix or plan forward.

Once the software has been corrected and pass unit testing. The new code will be integrated into the system and testing will resume. The failed test will be rerun in entirety to ensure software functionality.

## 2.8 Pass/Fail criteria

If any one test has <90% pass rate of test steps the test will be flagged as a failure. If >90% of the test is completed satisfactory then the test will be marked passed with bugs. Tests with >90% pass rate will be rerun once all software fixes have been re-integrated into the system.

## 2.9 Testing Resources

To successfully conduct integration and final product testing a suite of testing resources are needed. Those resources are documented in Table 2.9-1. For the most part all resources are required in order to perform testing, but if any workarounds are available they are also described in the table.

|  |  |  |
| --- | --- | --- |
| Resource | Resource Description | Workaround if Resource not Available |
| Android Phone or Emulator | A physical android phone or android emulator software to run the application | No workaround available |
| CardCzar Game | The application to be tested | No workaround available |
| Server (any hardware or virtual machine) that can support Apache, PHP and SQL | An Amazon Linux AMI server | No workaround available |
| Apache http server | linux http daemon | No workaround available |
| SQL server | MySQL server | It might be possible to install another type of SQL database, however we are limited on what Amazon provides for free |
| PHP | PHP with php-mysql module. This is the go-between for the CardCzar app and the SQL database | No workaround available |
| Game Files | Files stored on the Linux server that contain the game cards (bait and response cards) used to play the game | No workaround available or allow the users to provide their own files. |
| GitHub | Used to document bugs, questions or enhancement generated during the testing phase. | Bugs and enhancements will be documented in word (refer to Appendix A for details) and sent via Gmail to the team. If a tester has a question then standard Google Hangout conversation will be used. |

Table 2.9-1: Testing Resources

# 3.0 Methodology

## 3.1 Overview

The following section will describe the various testing methods that will be used for the Card Czar Application throughout the system development life cycle. This test will be the basis for test development but may be revised as needs change throughout the SLDC.

All testing will be conducted manually using phone emulation software and tools such as Test Rail and Blue Mix that are available for a free trial basis. Any mention of phone or phone display are referring to testing with a phone emulator. Bare metal phone hardware and automated testing tools will not be used.

## 3.2 Unit Testing

Unit testing will be performed during software development by the developer who wrote the code to ensure that the code is functioning as expected. Any stubs or drivers that are required will be developed and saved for future use.

## 3.3 Integration Testing

Integration testing will be performed after each development milestone during the system development. Integration testing will be conducted by the test team members with the support of the software development teams. Test case and scenario based testing will be used to test the individual modules and software functionality being delivered during that milestone.

Final integration testing will primarily consist of phone communication with the Apache Web Server and MySQL Database Server. Ensuring connectivity is working properly and permissions are correct.

## 3.4 Final Delivery Testing

Final delivery testing will be performed post final integration period. Test team members and software developers will be required to be on hand for delivery testing. Delivery testing will include all test cases used during integration testing as well as scripted step by step testing which will be the baseline for the user acceptance testing.

See Appendix B for detailed testing information

## 3.5 Bug Regression

All bugs that are in the “ready for retest” phase will be tested after they have been delivered to the development lab for testing. High Priority and Medium Priority bug reports will be tested as soon as they are delivered into the development lab. Lower priority bugs will be tested during regression testing at the end of the integration milestone.

## 3.6 Test Complete Criteria

Testing will be complete when the following conditions have been met:

1. All Unit tests have been completed
2. All test cases have passed
3. All High ad Medium Priority bugs have been resolved and retested SAT
4. All Low Priority bug reports have been properly documented
5. Final testing documentation has been signed by Testing Manager

## 3.7 Test Deliverables

The following documents will be delivered to the review group upon completion of testing:

System Test Plan

Test Logs

Test case summary

Bug report summary

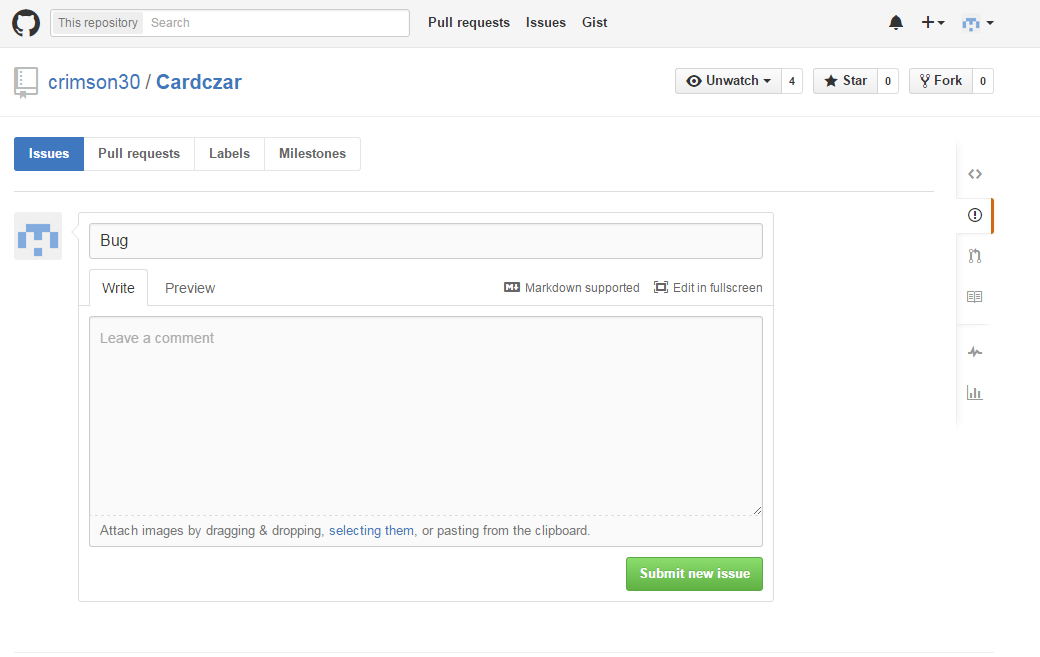
System test summary

# 4.0 Bug Tracking & Reporting

## 4.1 Bug Reporting

All software bugs will be logged using the GitHub project Repository. Test team members and developers will open a new bug ticket when software bugs are discovered. GitHub bug tickets should be filled out to the maximum extent possible by the test team member to ensure that the software developer fully understands the problem and what the expected result is. Any Github form not filled out correctly will be immediately rerouted back to the test team for further details before the lead developer assigns the ticket to a developer. The screen shots below show a standard bug reporting form to be completed by the test team member.

Figure 1: GitHub Issue Form



## 4.2Assigning Labels

Additionally, GitHub allows for the assignment of Labels for each issue entered. These labels assign keyword identifiers to the issue, and multiple labels can be assigned to an issue. These labels are meant to help the team better sort and prioritize the issues to be worked. For instance, as integration testing is occurring the test team can enter issues into GitHub. Those issues can be a bug, question or enhancement and are indicated as such by a label of that name attached to the issue. Those issues will also have a priority label to indicate the order in which they should be worked, again with the appropriate priority label attached to the issue. The issues can also have other labels that indicate what area/functionality the issue addresses (such as database, app functionality, etc). While these labels will help the team address and prioritize the issues entered, they are not meant to replace the issue description. As stated in section 4.1, each issue should have a full description of the bug/question/enhancement being documented. See Table 4.2-1 for the list of GitHub labels and their descriptions.

|  |  |
| --- | --- |
| **Label Name** | **Label Description** |
| Android | Issue specific to Android SDK or Android functionality (e.g. android.handlers) |
| Beautification | Indicates a cosmetic change to the application that had nothing to do with functionality (e.g. change in color scheme) |
| Bug | Indicates an error found with the application’s funcationality |
| Duplicate | Issue is a duplicate of another issue previous entered. |
| Enhancement | A suggested enhancement or improvement to the application but does not address an error in current functionality |
| Help wanted | Issue author is requesting help from another team member |
| High Priority | The issue is of the upmost importance and should be addressed as quickly as possible. |
| Invalid | Indicates that the issue is not a valid bug or enhancement, such as outside the scope of the application. |
| Java | Indicates a bug, question or help needed with the Java portion of the application |
| Low Priority | The issue is of the lowest importance and should be addressed after all High or Medium issues are addressed. |
| Medium Priority | The issue is needs to be addressed but after all High priority issues have been resolved. |
| PHP | Indicates a bug, question or help needed with the PHP portion of the application |
| Question | The issue author has a question regarding the system, e.g. how a specific button should work |
| Security | Indicates a bug, question or help needed with the Security portion of the application |
| SQL | Indicates a bug, question or help needed with the SQL portion of the application |
| WontFix | The issue is valid but will not be fixed. Could be due to time or resource limitations. |

# 5. Approvals

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_**

**Test Manager Date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_**

**Development Project Manager Date**

**Appendix A**

**Bug Report Template**

**1. Report Identifier**

BR-XX

**2. Summary**This section will identify the testing environment that the bug was discovered in. Test case number or test step number should be identified here. This bug report template may not be required if the bug tracking in GitHub is found to be sufficient.

**3. Bug Description**

This section should match the description of the problem on the JIRA bug report ticket.

**4. Impact to Testing**

If testing was halted or severely impacted, describe the scope of the problem here and what functionality can no longer be tested until this bug is resolved.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Tester Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Test Team Manager Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Lead Developer Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Project Manager Date

**Appendix B**

**Test Log Template**

**1. Log Identifier**

TL-XX

**2. Summary**

This section will describe the testing environment. Test case or scenario test number should be identified here.

**Security Testing**

Online free software such as IBM Blue Mix will be used to check the application software for security issues such as SQL injection issues

Online free web inspection software (TBD) will be used to review the Apache Web server for proper security configuration.

**Application Testing**

Using section 2.3.1 - 5 inputs and outputs the following test scenario will be completed from start to finish on one round with 4 players, of which 1 is the host (ActivePlayer).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Description** | **Trigger Action** | **Input/Output Type** | **Expected**  **Result** | **Correct (Y/N)** |
| 1 | User selects to host a game | Press hostButton on phone | Phone | Button press causes phone display to go to Start Room page |  |
| 2 | Host (AKA ActivePlayer) starts room - enters room name and host name | Press roomStartButton on phone | Phone | Button press causes phone display to go to Room Activity page |  |
| 3 | Room and Host name record created | Room and Host name passed after roomStartbutton is press. | Web->DB  Tables gamestate, users,questions, answers inserted. | Records appear in database |  |
| 4 | Phone screen refreshes with Room and Users | refresh from DB records containing users and room | Phone | Phone screen refreshes with users and room |  |
| 5 | User selects to join a room | joinButton on phone | Phone | Button press causes phone display to go to page to enter room and user information |  |
| 6a | Error Checking  User joins a room - enters room that doesn’ t exist.  name and user name | Press userStartbutton |  | “Room doesn’t exist error displayed on phone. |  |
| 6b | Error Checking  User joins a room - enters username is already taken | Press userStartbutton |  | “Username taken” displayed on phone. |  |
| 6c | Error Checking  User joins a room - enters game not yet started | Press userStartbutton |  | “Game not started” error displayed on phone |  |
| 6a | User joins a room - enters room name and user name | Press userStartbutton | Phone | Button press causes phone display to go to Room Activity page |  |
| 7 | User record created | Room name and user name record inserted | Web->DB  Tables users updated | Records appear in database |  |
| 8a | Error checking  Active Player tries to start game with less than 4 players | Press refreshUsersButton to retrieve data from database |  | “Insufficient Players” error is received |  |
| 8b | ActivePlayer refreshes phone display until sufficient users have joined | Press refreshUsersButton to retrieve data from database | Web->DB query | Phone display should match users table in DB for that room |  |
| 9 | Phone display refreshes with users | refresh from DB query | Phone | As each user joins, phone display updates with appropriate usernames for specified room. |  |
| 10 | ActivePlayer starts game | ActivePlayer presses gameStartButton | Phone | Phone display changes to Gameplay Activity screen |  |
| 11 | DB gamestate updated prepares questions and answers for game | Gamestate change passed from phone after gameStartButton is pressed | Web->DB | DB should update with “active’ gamestate |  |
| 12 | Phone screen updates as appropriate - ActivePlayer sees questions (adjective cards), Users see possible answers (dealt a hand of noun cards) | refresh from DB records containing gamestate, users, questions, answers | Phone | Phone screen changes for each user and Active Player. Users receive 4 noun cards and ActivePlayer picks and adjective card |  |
| 13 | ActivePlayer plays adjective | Select from choices on phone | Phone | Users see selected adjective card on phone display |  |
| 14 | User1 - select noun | Select from choices on phone | Phone | ActivePlayer phone display shows user selection as unidentified |  |
| 15 | User2 - select noun | Select from choices on phone | Phone | ActivePlayer phone display shows user selection as unidentified |  |
| 16 | User3 - select noun | Select from choices on phone | Phone | ActivePlayer phone display shows user selection as unidentified |  |
| 17 | DB records are updated after ActivePlayer and User selections | User/ ActivePlayer input on phone | Web->DB | DB records for each user should show proper adjective and noun cards in play |  |
| 18 | Once all answers are in ActivePlayer picks winner | Select from choices on phone | Phone | Phone display updates to End Game screen |  |
| 19 | Gamestate updated and round winner updated | Winner chosen by ActivePlayer | Web->DB | Each users phone display updates to show winner. Adjective card goes to winner. DB updates winner to become ActivePlayer as well as update field (points) to track number of wins. |  |
| 20 | Round winner becomes ActivePlayer | Refresh from DB | Phone | Phone display refreshes. Round winner now has ActivePlayer screen and other 3 players have user screens. |  |
| 21 | Repeat steps 1-20 above at least one more time to make sure only current game and current room information is stored and displayed to all players. | | | | |

**Sample Test Data Set - Adjective (Bait Cards) and Nouns (Response Cards)**

|  |  |  |  |
| --- | --- | --- | --- |
| Czar Bait Cards | | | |
| A Bad Haircut | A Desert | A Million Dollars | A Haunted House |
| Actors | Bears | Basketball | Bill Gates |
| California | Christmas | Diamonds | Donuts |
| Emergency Rooms | Family Reunions | Fruit | George Washington |
| Gravity | Halloween | Hamburgers | Jail |
| Judge Judy | Knife & Fork | Las Vegas | Lawyers |
| Mars | Marriage | NASCAR | Ninjas |
| Oxygen | Peanut Butter | Popcorn | Rap Music |
| Salads | Show | Trick | Television |
| The 1980s | Umbrella Drinks | Video Games | Waterparks |

|  |  |  |  |
| --- | --- | --- | --- |
| Czar Response Cards | | | |
| American | Arrogant | Beautiful | Best |
| Classic | Cold | Crazy | Dangerous |
| Easy | Evil | Fresh | Fun |
| Goofy | Harsh | Healthy | Influential |
| Juicy | Little | Lucky | Misunderstood |
| Neat | Odd | Playful | Powerful |
| Quiet | Rare | Rough | Scary |
| Serious | Short | Stupid | Terrible |
| Tough | Terrific | Unhealthy | Violent |
| Warm | Weird | Wired | Yucky |

**3. Test Log**

Fill out the attached .xls and route electronically or print and route paper copy with the Test Log.

