Test Plan

Grand Slam Baseball Allstars 2: Electric Boogaloo

COP 4331, Spring, 2018

Team Name: Team MLB Card Game 2

Team Members:

- Daniel Carman
- Malik Henriquez
- Ronald Marrero
- Brian Wengier
- Kaleb Yangson

Contents of this Document

I. Test Plan

Overall Objective for Software Test Activity

The goal of this software test plan is to provide our team with a framework for ensuring the reliability of our system and for assessing its adherence to our requirements. The tests outlined in this plan will assist us in finding faults with our system so that they may be corrected before the system's delivery. Additionally, the tests are designed in a way such that they can be mapped backwards to each requirement in order to demonstrate a requirement implementation.

Description of Test Environment

The majority of unit and system testing will be done by the developers of the system on android emulators run on windows machines. The emulators will be running Android version 6.0, API 23, as it offers relatively good compatibility with modern devices and our team is familiar with it. Each team member will create a different Android Virtual Device (AVD), corresponding to a different popular phone running Android 6.0. This will allow us to gain some insights as to how the system might run on a range of hardware systems.

Running the unit testing in emulators rather than on physical devices will allow us to perform some automated testing that would be incredibly tedious to do by hand. One the system has a working build, system testing will first be performed on emulators and then will be performed on physical devices, to ensure the system remains functional in its final installation environment.

Stopping Criteria

Testing will be conducted in blocks, with each block being focused on one major module of the system and lasting 1-2 hours in duration. During each block, a single test case explicitly concerning that module will be repeated to ensure each component in that module is functioning as expected. These test cases are designed to require the proper functioning of all components in their module, meaning that they will not properly conclude if even 1 component in the module is not functioning correctly. If any faults are discovered during testing, they will be noted with details regarding the fault and, if possible, reproduction steps for the fault.

If a fatal issue is encountered that prevents the proper functioning of the system past a certain point, the block will conclude prematurely and the team will work on correcting this fatal fault. Once it is corrected, the team will attempt to reproduce the fault, and then resume with the current testing block if the fault cannot be reproduced and the proper documentation has been created.

After a block has concluded, everyone on the team will be assigned one or several faults to investigate. Their goal is to find our what is causing the faults to occur and how to prevent them from occurring, either by means of data guarding, error handling, or logic debugging. Once the fault is corrected, that team member will document what changes were made both in code and in documentation to assist future system maintenance specialists. After all faults have been corrected, the block will be run once again to ensure the proper functioning of the system. The system will be deemed good enough to deliver after it successfully passes all blocks one after another.

Description of Individual System Test Cases

Test Case No.	Test Objective	Test Description	Test Conditions	Expected Results
Case 1	This test shall demonstrate the proper functioning of the login module.	An automated test will be used to test this module. Each correct pair of user and password strings will be input into the login screen, and an attempt will be made to log in to the system. The process will be completed with each correct user name with an incorrect password to ensure the system is properly authenticating login.	See test environment	The system will allow login for each account on the first pass, and not allow login on the 2nd pass.
Case 2	This test shall demonstrate the ability to create a new user.	Manual testing shall be used to test this functionality. First, the tester shall attempt to make a new user account with a previously unused username. After this, the tester shall attempt to create a new user with that same username in order to test how the system handles an attempt at a duplicate name. To conclude the test, the tester shall attempt to recover their username and password using their email account.	See test environment	A new user account will successfully be created on the first try, but the attempt will be denied the 2nd try. The tester shall receive email instructions containing their username and a link to reset their password.
Case 3	This test shall demonstrate the ability to play a full, simulated game of	Manual testing shall be used to test this functionality. First, the tester shall attempt to find a game when there is no other user information on	Pass 1: The test shall be conducted on an empty database. Pass 2: The test	The results of the game will be reflected in the user database for both games.

	baseball.	the server to test if the system can generate a fake team. This test will be passed if the tester can get through the game with only minor cosmetic issues. Next, the tester will repeat the test when the database is populated with other user data to test if it can retrieve another user's team.	shall be conducted on a populated database.	
Case 4	This test shall test the system's ability to let a user view their collection and edit their teams.	This test shall be conducted manually. In this test, the tester shall go into the collection screen, view their collection, and attempt to create a new team from their collection. Then, they shall attempt to edit their team after creation. If the tester can do all of this without faults, this test shall be passed.	The collection of the tester will be populated with enough cards to form a team.	The new team and the changes made to it will be visible in the database.
Case 5	This test shall test the system's ability to let the user purchase packs with in game currency.	In this test, the tester shall first attempt to purchase a pack with the requisite currency on their account. After this, they will attempt to purchase a pack without the requisite currency.	The tester account will have exactly as much currency as it takes to get a pack	The first pack should be purchased successfully, with the cards in it showing up in the database. The second pack should be unsuccessful.
Case 6	This test shall test if the system allows the user to smoothly exit the application.	In this test, the tester will attempt to exit the application via the exit option on the main menu. If the system updates any changes to the database and then closes the application without fault, this test will be passed.	See test environment	Any changes made to teams should be updated to the database before close.
Case 7	This test shall	This test case shall be	See test	Each option

	examine the functionality of the options menu.	performed manually. In this test, the tester shall examine each option within the options menu and determine if it is functioning properly.	environment	should either have a clear effect on the settings of the app or a clear visual output in the case of the help screen.
Case 8	This case shall test quality requirements.	This test case will be done manually.This test is broken up into 7 subtests, which each test particular quality requirements (requirement IDs in parentheses): 1. Must run on a version of android above 4.4 (10) Output: Pass 2. The database will be built by our team with MySQL, the app will be created with unity and C# (19, 20) Output: Pass 3. The app will be tested on android emulators and phones (21) Output: Pass 4. Connections and user data will be secured with SSL and HTTPS (23, 30) Output: Pass 5. Must display a game's winner via a graphic (18) Output: Pass, Fail 6. The system shall have an uptime of	This test must be performed extensively on actual phones; a real hardware environment has a higher chance of faulty behavior when compared to an emulated environment.	This test will be evaluated on a per-subtest basis. For example, the system can pass for subtest 1, but fail for the rest of them, or vice versa. For time based requirements, the system should show high robustness, staying active without fatal faults for a long period of time. Qualitative judgements given should be as objective as possible.

		99%, shall be available to customers 24/7, shall email admins on a fatal fault, and shall be restarted within 5 minutes of failure. (25, 27, 28, 29) Output: Pass, Fail 7. The system must be visually pleasing (26) Output: Pass, Fail Subtests 1-4 (and also their requirements) are self sufficient and thus will never impede this test's success. Test 5 is a binary decision; pass if there is a winner's graphic, fail if there isn't. Test 6 will be evaluated as passed if the system is capable of running without failure for the entire testing block. Test 7 will either pass or fail based solely on the tester's decision.		
Case 9	This test case shall test all components of the system that connect to the database for functionality.	This test will be performed manually.This test is broken up into 6 subtests, which each test particular database related requirements (requirement IDs in parentheses): 1. Ability to login (1, 1.1, 22) Output: Pass, Fail 2. Ability to make new account (2) Output: Pass, Fail	This test must be performed extensively on actual phones; mobile data connections can be much more unreliable when compared to LAN or WiFi.	This test will be evaluated on a per-subtest basis. For example, the system can pass for subtest 1, but fail for the rest of them, or vice versa.

		3. Ability to search for team on the database (3) Output: Pass, Fail 4. Ability to update game stats to the database (4.2, 4.3) Output: Pass, Fail 5. Ability to retrieve user data (9, 9.1, 9.2, 9.3, 9.4) Output: Pass, Fail 6. Ability to maintain an active connection until user is inactive for 10 min. (11, 24) Output: Pass, Fail Any non fatal faults found during testing should be noted and placed in the pool of need-to-fix bugs.		

Trace of Individual Test Cases to the Requirements Traceability Matrix

Req ID.	Requirement Description	Architecture Reference	Design Reference	Test Case Reference
1	The system shall allow the user to login to the system	User interface, Database	User, Database Model	Cases 1, 9
1.1	The system shall notify the user in the event of a failed login	User interface, Database	User, Database Model	Cases 1, 9
2	The system shall allow the user to create a new account	User interface, Database	User, Database Model	Cases 2, 9
3	The system shall allow the user to search for a	User interface, Game Engine,	Main Menu, DatabaseModel	Cases 3. 9

	team to play a game against	Database		
3.1	If no other teams are available, the system shall generate a fake team	Game Engine, Random Number Generator	Game, Team, Player	Case 3
4	The system shall allow the user to play a game	Game Engine, Random Number Generator, Game Mechanics, Front End Graphics	Team, Game, Player, SpecialObjects	Case 3
4.1	The system shall allow the user to choose when to start a turn	Game Engine	Game	Case 3
4.2	The system shall allow the user to end a game via completion	Game Engine	Game, DatabaseModel	Case 3
4.3	The system shall allow the user to concede a game before its conclusion	Game Engine	Game, DatabaseModel	Cases 3, 9
5	The system shall allow a user to view their collection of cards	Front End Graphics, User Interface	User, Team, Player, Collection, DatabaseModel	Cases 4, 9
6	The system shall allow the user to create and manage teams	Front End Graphics, User Interface	User, Team, Player, Collection, DatabaseModel	Cases 4, 9
7	The system shall allow the user to view and purchase available packs with acquired in game currency	Game Engine, Front End Graphics, User Interface	DatabaseModel, Store	Cases 5, 9
8	The system shall allow the user to exit the application	User Interface	MainMenu	Case 6
9	The system shall be able to retrieve user information from the	Database	DatabaseModel	Cases 3, 4, 9

	database			
9.1	The system shall be able to retrieve team information from the database	Database	DatabaseModel	Cases 3, 9
9.2	The system will be able to retrieve the score of a game.	Database	DatabaseModel	Cases 3, 9
9.3	The system will be able to retrieve store information.	Database, Game Mechanics	DatabaseModel, Store	Cases 5, 9
9.4	The system will be able to retrieve game information.	Database	DatabaseModel	Cases 3, 9
10	The system will operate on an Android version above version 4.4	Game Engine	Game, DatabaseModel	All Cases
11	The system will have an active data connection with the database.	Database	DatabaseModel	Cases 1, 2, 3, 4, 5, 9
12	The system will be able to adjust text size.	Database, User Interface, Front End Graphics	DatabaseModel, Options	Case 7
13	The system will be able to retrieve forgotten user info for users.	Database, User Interface	DatabaseModel	Cases 2, 9
14	The system will allow users to view the starting manual.	Database, User Interface, Graphic Generator	DatabaseModel	Case 7
15	The system will determine who will have access to the game documentation.	Game Engine, Database	DatabaseModel	Case 9
16	The system will have a central documentation repository	Database	DatabaseModel	Cases 8, 9
17	The system will use	Game Mechanics,	DatabaseModel, Game	Case 3

	calculations to play games.	Database, Random Number Generator		
17.1	The system will gather player stats before games.	Database, Game Engine	DatabaseModel, Game	Cases 3, 9
17.2	The system will use calculations to determine the success of certain actions.	Game Mechanics, Database, Random Number Generator	DatabaseModel, Game, Team, Player, Special Objects	Case 3
17.3	The system will decide who the winner is if the two teams end up with the same score.	Game Mechanics, Database	DatabaseModel, Game, Team, Player, Special Objects	Case 3
18	The system will display a winner to the user using a graphic after the game.	Game Mechanics, Graphic Generator, Database	DatabaseModel, Game	Case 3
19	The database shall be built by 5 individuals with some database experience	Database	DatabaseModel	Cases 8, 9
20	The core game application shall be built by 5 individuals using the Unity engine, C#, and Visual Studio	User Interface, Graphic Generator, Game Mechanics, Random Number Generator, Game Engine	All Classes	All Cases
21	Product Testing shall be performed on Android emulators and Android phones	NA	NA	All Cases
22	The system shall be accessed only by authenticated users	User Interface	User, Admin	Cases 1, 9
23	The system shall use cryptographic protocols	NA	NA	Cases 8, 9

	such as SSL and HTTPS for network communications			
24	The system shall end the session automatically when an open session is not used for 10 minutes	NA	NA	Case 6
25	The system shall have an uptime of 99%	NA	NA	Case 8
26	The system shall present information to the user with a visually pleasing interface	User Interface, Graphics Generator	All Classes	Case 8
27	The system shall be available to both customers and admins 24/7	NA	NA	Case 8
28	The system shall be restarted within 5 minutes of system failure	NA	NA	Case 8
29	The system shall alert an admin via email when the system has a failure	NA	DatabaseModel	Case 8
30	The system shall protect customer's information using SSP and HTTPS	NA	NA	Case 8, 9

II. Test Results

Description of the Actual Test Environment

The actual testing environment didn't differ much from what was expected, however we did ignore one tool in our initial description that was going to be extremely helpful during testing; the Unity Editor. Besides use of the editor to handle dependencies between objects and translations in the game world, the actual testing was still performed on android emulators during each partial build.

Results of Testing

Unit Tests

Throughout implementation of our system, our team has been performing unit testing on the components that they have been developing. Below is a short synopsis of some of those tests.

Daniel Carman - Core Game

My primary development task has been creating the core game application. Currently, the framework behind the game's functionality, consisting of teams, players, items, and the ball are implemented and functional, all that needs to be added is the logic that utilizes the stats along with some RNG to produce the math that drives matches.

Currently the system meets game related requirements 3.1, 4, 4.1, 17, 17.1, and 17.2. This means that the system is capable of generating a fake team to play against, starting up a new game with the active player team and an opposing team, and letting the user start their turn, thus causing the game to progress according to the currently implemented game logic. These requirements were tested by running code in the Unity editor throughout development, documenting which requirements were satisfied, and occasionally checking my work by running the full system on an emulator.

Currently, requirements, 3, 4.2, 4.3, 17.3, and 18 have not been implemented. This means that the system currently doesn't have any end conditions implemented for the game, which also implies that it has no winner's graphic, and the team search function is not currently implemented.

Moving forward, my top priority is to finish implementing the game logic and end conditions so that I may work with Ron to get the team search function properly working.

Ronald Marrero - Database Connection

My primary development task has been establishing the data connections from the app to the database. Connections to the database are only made with users after they have logged in. After that, each connection is made over HTTPS to the Firebase database. Testing ensured full functionality of the previously mentioned components as well as reading/writing to the realtime database. Further development is required to test game logic against the database as only standard CRUD (Create Read Update Delete) operations have been tested.

Kaleb Yangson - Options Menu/Common Assets

My primary development task has been building a visually pleasing interface that allows both the customization of the game (Options Menu) and the currency shop that allows the purchase of crates using in game currency (Common Assets). Testing ensured functionality of menu buttons and navigation. Further development is required for the actual in game currency system as well as universal variables that will be able to be edited from the options menu.

Brian Wengier - Card Collection/Main Menu

My primary development task has been building an interface that allows players to both navigate from the main menu to various other interfaces (such as Options Menu, Play Menu, and Quit) and to view the player's card collection. Testing ensured the functionality of menu buttons and navigation. It also ensured the ability to pull up a player's cards and view their stats. Further development is required on the storing and calling of cards with the server and a more visually pleasing interface for the main menu.

Malik Henriquez - Pause Menu / Animation Assets

My primary development task has been building a pause menu that would allow a user to interrupt a game in progress as well as adding the animation assets needed while the game is being played. For the pause menu, testing ensured that any background game processes would be put to sleep while the user decided which option to select on the pause menu. Testing of the animation assets in unity ensured that there were no glitches in how the animations were occurring and also ensured that the animations were occurring at the proper times.

This testing has not yet been done in conjunction with the database connection but has been confined to local debugging. The next and final step for my piece of integration testing is to ensure that there are no faults from pausing a game in progress and that once the pause menu is closed, the connection re-opens and the game resumes.

System Tests

Once our system has progressed to its first full build, we will begin performing the comprehensive system tests proposed in the previous section. Initially they will be performed on android emulators to ensure the system can operate under optimal conditions, and then the tests will be performed under the nonoptimal conditions presented by mobile hardware.

Remaining Tests

Our system is still in development and as such, there are some components, such as the core game functionality, that cannot be tested yet. As the functionality of the system expands, further tests will be conducted and included in this log.

Requirements Traceability Matrix (With Status)

Req ID.	Requirement Description	Architecture Reference	Design Reference	Test Case Reference	Status
1	The system shall allow the user to login to the system	User interface, Database	User, Database Model	Cases 1, 9	Pass (unit testing)
1.1	The system	User interface,	User, Database	Cases 1, 9	Pass (unit

	shall notify the user in the event of a failed login	Database	Model		testing)
2	The system shall allow the user to create a new account	User interface, Database	User, Database Model	Cases 2, 9	Pass (unit testing)
3	The system shall allow the user to search for a team to play a game against	User interface, Game Engine, Database	Main Menu, DatabaseModel	Cases 3. 9	In Development
3.1	If no other teams are available, the system shall generate a fake team	Game Engine, Random Number Generator	Game, Team, Player	Case 3	Pass (unit testing)
4	The system shall allow the user to play a game	Game Engine, Random Number Generator, Game Mechanics, Front End Graphics	Team, Game, Player, SpecialObjects	Case 3	Pass (unit testing)
4.1	The system shall allow the user to choose when to start a turn	Game Engine	Game	Case 3	Pass (unit testing)
4.2	The system shall allow the user to end a game via completion	Game Engine	Game, DatabaseModel	Case 3	In Development
4.3	The system shall allow the user to	Game Engine	Game, DatabaseModel	Cases 3, 9	In Development

	concede a game before its conclusion				
5	The system shall allow a user to view their collection of cards	Front End Graphics, User Interface	User, Team, Player, Collection, DatabaseModel	Cases 4, 9	In Development
6	The system shall allow the user to create and manage teams	Front End Graphics, User Interface	User, Team, Player, Collection, DatabaseModel	Cases 4, 9	Pass (unit testing)
7	The system shall allow the user to view and purchase available packs with acquired in game currency	Game Engine, Front End Graphics, User Interface	DatabaseModel, Store	Cases 5, 9	In Development
8	The system shall allow the user to exit the application	User Interface	MainMenu	Case 6	Pass
9	The system shall be able to retrieve user information from the database	Database	DatabaseModel	Cases 3, 4, 9	Pass (unit testing)
9.1	The system shall be able to retrieve team information from the database	Database	DatabaseModel	Cases 3, 9	In Development
9.2	The system will be able to retrieve the score of a	Database	DatabaseModel	Cases 3, 9	In Development

	game.				
9.3	The system will be able to retrieve store information.	Database, Game Mechanics	DatabaseModel, Store	Cases 5, 9	In Development
9.4	The system will be able to retrieve game information.	Database	DatabaseModel	Cases 3, 9	Pass
10	The system will operate on an Android version above version 4.4	Game Engine	Game, DatabaseModel	All Cases	Pass (self sufficient)
11	The system will have an active data connection with the database.	Database	DatabaseModel	Cases 1, 2, 3, 4, 5, 9	Pass (unit testing)
12	The system will be able to adjust text size.	Database, User Interface, Front End Graphics	DatabaseModel, Options	Case 7	In Development
13	The system will be able to retrieve forgotten user info for users.	Database, User Interface	DatabaseModel	Cases 2, 9	In Development
14	The system will allow users to view the starting manual.	Database, User Interface, Graphic Generator	DatabaseModel	Case 7	In Development
15	The system will determine who will have access to the game documentation	Game Engine, Database	DatabaseModel	Case 9	Pass (self sufficient)

	.				
16	The system will have a central documentation repository	Database	DatabaseModel	Cases 8, 9	Pass (self sufficient)
17	The system will use calculations to play games.	Game Mechanics, Database, Random Number Generator	DatabaseModel, Game	Case 3	Pass (unit testing)
17.1	The system will gather player stats before games.	Database, Game Engine	DatabaseModel, Game	Cases 3, 9	Pass (unit testing)
17.2	The system will use calculations to determine the success of certain actions.	Game Mechanics, Database, Random Number Generator	DatabaseModel, Game, Team, Player, Special Objects	Case 3	Pass (unit testing)
17.3	The system will decide who the winner is if the two teams end up with the same score.	Game Mechanics, Database	DatabaseModel, Game, Team, Player, Special Objects	Case 3	In Development
18	The system will display a winner to the user using a graphic after the game.	Game Mechanics, Graphic Generator, Database	DatabaseModel, Game	Case 3	In Development
19	The database shall be built by 5 individuals with some database experience	Database	DatabaseModel	Cases 8, 9	Pass (self sufficient)

20	The core game application shall be built by 5 individuals using the Unity engine, C#, and Visual Studio	User Interface, Graphic Generator, Game Mechanics, Random Number Generator, Game Engine	All Classes	All Cases	Pass (self sufficient)
21	Product Testing shall be performed on Android emulators and Android phones	NA	NA	All Cases	Pass (self sufficient)
22	The system shall be accessed only by authenticated users	User Interface	User, Admin	Cases 1, 9	Pass (unit testing)
23	The system shall use cryptographic protocols such as SSL and HTTPS for network communications	NA	NA	Cases 8, 9	Pass (self sufficient)
24	The system shall end the session automatically when an open session is not used for 10 minutes	NA	NA	Case 6	Pass (self sufficient)
25	The system shall have an uptime of 99%	NA	NA	Case 8	In Development

26	The system shall present information to the user with a visually pleasing interface	User Interface, Graphics Generator	All Classes	Case 8	In Development
27	The system shall be available to both customers and admins 24/7	NA	NA	Case 8	In Development
28	The system shall be restarted within 5 minutes of system failure	NA	NA	Case 8	In Development
29	The system shall alert an admin via email when the system has a failure	NA	DatabaseModel	Case 8	In Development
30	The system shall protect customer's information using SSP and HTTPS	NA	NA	Case 8, 9	Pass (self sufficient)

Conclusion

Our conclusions from the current testing we've done is that the system is coming along nicely. Testing commenced only relatively recently, and the system isn't fully implemented yet. However, the modules that are capable of being tested on their own and that are completed are passing unit tests flawlessly. We definitely have work to do with the implementation of our system, such as fully implementing the game logic and pulling assets from the database, but this test plan gives us a good way to gauge our current progress and the current functioning of our system.