Software Testing Plan

Version 1.0

04/02/19

SciKids

Sponsor: Elizabeth Glass

Mentor: Austin Sanders

Samantha Earl, Claudia Coronel, Gwen Morris

Table of Contents

1. Introduction

According to the U.S. Department of Commerce, STEM occupations are growing at 17% annually while other occupations are growing at 9.8% annually. These statistics show that there is a growing need for workers with a STEM-related education. Science, technology, engineering, and mathematics play an important role in bettering the future for the U.S. A study from the PEW Research Center, a nonpartisan American fact tank, has reported that most Americans believe the k-12 STEM education in the United States is mediocre at best. Some problems associated with this view on STEM education include disinterested students, outdated curricula, and outdated materials. There is an overall lack of engaging systems that help promote STEM careers. To help solve this issue, the team has created a gesture-based engagement program with a focus on STEM-related content. The program includes a navigational system that includes modules that are used to hold STEM-related content.

The team is currently finalizing our system with software testing that will allow the team to evaluate the functionality of the software created. This will include unit, integration, and usability testing. Unit testing will break up the system into its main components and test them separately. We will then ensure that all of the main components are compatible with each other through integration testing. Finally, we will have potential users interact with the system to make sure that the system can be easily learned and used by people of different age groups.

These tests are important to this system as our client plans to have a wide range of people use it for educational purposes. If a failure occurs while a user engaging in the product, the user can become disinterested and the system would have failed at its overall purpose in engaging and entertaining users. This system will also serve as the foundation for future programmers to add more content to, so we must ensure that it works properly before others

can add more to it. To begin testing, we will discuss how we plan to split the system up into smaller units and test those units individually.

2. Unit Testing

Unit testing looks at individual components of the software to confirm that each unit performs as expected. To ensure the functionality of our system, we will be creating several unit tests. We will be creating these tests using Unity's built-in Test Runner module. This will allow us to write tests in C# and run them in the Unity editor. We have decided to split up the tests into 3 main categories: the local SQLite database, gesture recognition, and methods used for the activities we have included in the system. Functionalities provided by Unity and Nuitrack, such as clicking buttons, changing scenes, object collisions, etc., will not be tested. This decision is based on the assumption that these methods have already been thoroughly tested by the providers before public distribution.

2.1 SQLite Database

The SQLite Database in our system is responsible for storing users' personal profiles, information for personalized available modules, personal progression, and new high scores. These features are accessed through the Registration, Login, and DBManager classes.

2.1.1 Registering a New Account: This test will make sure that a database is created if it does not already exist, and that a new user is successfully added to the "user" table. The following are functions that will test methods found in the Registration class.

Test	Equivalence Partitions	Boundary Values	Invalid Inputs	Example Input	Expected Output
Make-Database	N/A	N/A	N/A	N/A	A new local database is created with the “user,” “score,” and “module” table.
VerifyInputs	Username:a string of 8-30 ASCII characters. Password: a string of 5-30 ASCII characters.	Username: A string of 7, 8, 20, 30 and 31 ASCII characters. Password: A string of 4, 5, 20, 30, and 31 ASCII characters.	Usernames under 7 characters or over 30 characters. Passwords under 5 characters and over 30 characters.	Username: testname1 Password: testpass1	True if inputs are considered valid. False otherwise.
AddAccount	Username: a string of ASCII characters. Password: a string of ASCII characters.	N/A - VerifyInputs must return true before AddAccount is called.	N/A - VerifyInputs must return true before AddAccount is called.	Username: testname1 Password: testpass1	Username and password are added to the “users” table in database.

2.1.2 Login to an Existing Account: The Login class checks the database to see if an account exists. If the account does exist, the user is logged in and other scenes will change to reflect that. When a user is logged in, a temporary file is made with the user’s ID. This will allow the DBManager (discussed in the next subsection) to access the account’s information when needed.

Test	Equivalence Partitions	Boundary Values	Invalid Inputs	Example Input	Expected Output
Account-Exists	Username: a string of ASCII characters. Password: a string ASCII characters.	Username: 0, 1, 8 characters. Password: 0, 1, 5 characters.	Username: NULL Password: NULL	Username: testname1 Password: testpass1	True if a match is found with both the username and password. False otherwise.
GetID	Username: a string of ASCII characters.	Username: 0, 1, and 8 characters.	Username: NULL	Username: testname1	An integer with the user's ID.

2.1.3 Database Management: The DBManager is used by other classes (other than Register and Login) as a middleman to interact with the database.

Test	Equivalence Partitions	Boundary Values	Invalid Inputs	Example Input	Expected Output
GetStatus	N/A -	N/A	N/A	N/A	True If a temporary file with the user's ID exists. False otherwise.
WriteID	Integer >= 1	-1, 0, 1	Integers < 1	1	A temporary file is created and contains the user ID.
Get-Username	Integer >= 1	-1, 0, 1	Integers < 1	1	A string with the username associated with the ID.

AddScoreTo DB	Game ID: Integer >= 1 User ID: Integer >= 1 User Score: Integer >= 1	Game ID: -1, 0, 1 User ID: -1, 0, 1 User Score: -1, 0, 1	Game ID: Integer < 1 User ID: Integer < 1 User Score: Integer < 1	Game ID: 5 User ID: 1 User Score: 20	If a user ID is already in “score” table, the user’s score is updated if it is higher than the previously saved score. Otherwise, a new row is added with all three values.
GetHigh-Score	User ID: Integer >= 1 Game ID: Integer >= 1	User ID: -1, 0, 1 Game ID: -1, 0, 1	User ID: Integer < 1 Game ID: Integer < 1	User ID: 1 Game ID: 5	The user’s score for the correspon-ding game.

2.2 Gesture Recognition

The Gesture recognition portion of the system is responsible for ensuring that the user will be able to use gestures as a means of navigation through the system, as well as interact with the different modules available. Therefore the items that we will be testing will be focusing on the backend’s ability to recognize and interpret these gestures.

Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input/Action**	**Expected Outputs**
UserFound()	N/A	N/A	User stands in front of the depth sensing camera.	User’s body is interpreted with Nuitrack’s SDK. A text will appear on the screen saying “User found”

HandTracking()	N/A	N/A.	User stands in front of the depth sensing camera and waves their right hand around the screen	User's hands are interpreted with Nuitrack's SDK. A text will appear on the screen showing coordinates of hands at a specific moment in time
ItemDestroyed()	N/A	N/A	User stands in front of the camera and touches objects on the screen.	User's contact with the objects will cause them to disappear, triggering a text on the screen displaying "Destroyed"

2.3 Content

As proof of feasibility, the team has created a couple of games to represent content in the module systems. Two of the content based games that we will be testing include an interview game as well as a math based game. Described in this section are methods we plan to implement regarding those specific activities.

Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input/Action**	**Expected Outputs**
ButtonClick()	N/A	N/A		Checks if counter correctly totals the points acquired for those interviewing.

EndGame()	N/A	N/A.	User finishes the interview game	Game provides feedback based on the person that the individual has chosen for the job.
MathValidation()	N/A	N/A	Randomized math equations made from a function.	Correct math equations

3. Integration Testing

The purpose of integration testing is to make sure all of the units are capable of interacting and working together. As the user navigates through our system, we want to ensure that data is not lost across different aspects of our project. We want to make certain that when the different modules or our system are combined, they deliver optimal results. In order to prove the effectiveness of our system, we will perform integration testing. During our testing, we will be incorporating the different modules combined as to exhibit functionality. These will be done within the Unity editor. In the following sections we will describe our integration testing plans.

3.1 Content Accessing Database:

As stated in section 2.1, the database stores users' high scores, personal progression, and information for personalized profiles. This contributes to the user's overall interest in the system, as it inspires a level of competitiveness with either themselves or with others. Because of these reasons, it is important that the database can be manipulated by the activities implemented.

The team has created two activities as a proof of feasibility: an interview activity for college and/or community members, and a simple math game for children from grades K-5. The interview game serves as a thought exercise, so no score system is implemented. However, the math game does implement a score system. The math activity updates the UserScore field in the "score" table when the user earns a higher score than the one previous.

Integration Test	**Sample Input**	**Expected Output**
Math activity can edit database through DBManager.	A user who is logged in sets a new high score in activity.	The value for userScore in the "score" table is updated.

3.2 Navigation with Gesture Recognition:

Gestures are one of the ways users can access scenes and interact with module content. The GUI must respond to gestures from the user that allows the user to navigate the program. The exception of not using gestures in the menu scenes would be to log in to an account as this requires keyboard input for the email and password.

Integration Test	**Sample Input**	**Expected Output**
Click on buttons using gestures, starting at the main menu.	User stands in front of camera and hovers hand over a button for 3 seconds.	The button is clicked and the scene changes.

3.2 Content with Gesture Recognition:

The content that the team has provided are capable of reading gestures. Some content has more gesture involvement then others. The mathematics game is heavily gesture based and involves the entire users body. The interview game is simple and requires basic gesture interactions with a series of buttons.

Integration Test	Sample Input	Expected Output
Math activity can be used with gestures	User will stand in front of a depth sensing camera for body recognition. As equations are shown on the screen in a timely manner the user will need to destroyed the game object with the right answer.	As game objects with the correct answer are destroyed. The system will validate the answer and if it is the correct one, it will increment the score.
Community-College activity can be used with gestures.	User can select available options by hovering over a button for 3 seconds	Buttons in activity perform what they are expected to do.

4. Usability Testing

In order to decide how the usability for our system should be tested, we have to take into consideration who will be using it. The purpose of this phase will be to ensure that users can easily learn how to use our product with little to no difficulty. Because our system is meant for users of almost any age, the learning curve should be as simple as possible. Our usability testing will primarily consist of test groups of different ages to interact with our system. This will include functions like navigating the menus, creating a new account, and playing at least one of the activities provided. Another requirement that is important to our system is that the user finds it engaging and pleasant.

Some of the modules and content that have been created targets individuals who are in K-12 and college/community levels. Hence, our population for this study will include individuals in K-12 and community/college students. We will be taking individuals from both groups both male and female. After they have completed the tasks described in this section we will be conducting a survey in order to better our system. The responses will be analyzed by the team and critics made to the system that will benefit users. By conducting multiple studies on the

different groups our system will be targeting we hope to gain insight on how to better our system.

4.1 Navigability

For our system we have provided the users with easily readable text and appropriated sized buttons for them to interact with. These navigation buttons can be found in the corners of every scene in the project. We have made these buttons simple and intuitive. To navigate throughout our web application we have incorporated a 3 second hover that will be registered as a button click.

Purpose:	The user will be able to easily navigate throughout the different scenes in system using gestures/mouse
Task:	Navigate through the entire system using gestures. Ensure that you render each page at least once
Plan:	Taking at least two individuals for each of the modules we have developed and ensuring that there is no confusing with how to navigate the system.
Survey:	● On a scale of 1-10 how easy was it to navigate the system? ● Did you enjoy using your hands to move through the system? ● Could you reach all of the pages described?

4.2 Overall satisfaction

One of the biggest requirements of our system is for it to be engaging and pleasant to use. For this, we have developed an easy to learn system, that shouldn't take users a long time to

establish how to navigate around the menu screens with gestures that are appropriate. We have also chosen a color scheme that is visually appealing and doesn't have issues with text readability. The gestures we have chosen are not strenuous on the individual, and have a timed limit. The timed limit ensures that the user will only be interactive for the amounts of time they choose, as to strain away from causing headaches or fatigue.

Purpose:	The user will find the system engaging and enjoyable
Task:	Run through the respective module in your age group, try at least one of the activities
Plan:	Taking at least two individuals for each of the modules we have developed and asking them their opinions on the content. Posing the right questions based on graphics and gestures.
Survey	● Did you find the material engaging and informative? ● On a scale of 1-10 how enthusiastic were you to continue? ● Would you use this system again?

5. Conclusion

In conclusion, SciKids has finished developing our gesture-based learning system focused on STEM recruitment. The next steps towards our finished product are completing the tests as described in this document. We will be running hard-coded tests to assure that everything is working correctly in respect to our sponsor's requirements. By planning out unit, integration, and usability testing we will prove all possible workflows of our system. Unit testing will be focused on proving the operational portions of our system, while the integration testing will focus on the

organization of data throughout our system, Finally, our usability testing will focus on giving the user a satisfactory experience with our system. In creating these several testing phases, we will be certain that our end product will be fully functioning and engaging for future users.