# Software Testing Plan

# SmartTalk

# March 7, 2021

Sponsor: Dr. Okim Kang

Mentor: Fabio Santos

Version: 1.0

Team members: Joseph Vargovich (Lead), Andrew Munoz, Kehan Cao, Christian Bito-on, Malik Jones

# Table of Contents

# 1.0 Introduction

SmartTalk has been working on more implementations of ‘iSpeak,’ a Gamified Mobile Language Learner Application, under the guidance of team mentor Fabio Santos and the project’s sponsor, Dr. Okim Kang. Dr. Kang is the director of the Applied Linguistics Speech Lab at Northern Arizona University and has asked SmartTalk to help her make a system that will function not only as a powerful research tool for her graduate students but also as a tool for assisting foreign language learners with their pronunciation.

We will begin our analysis of our testing plan with brief unit testing, which is necessary to ensure that everything is functioning as intended and to locate any program errors. SmartTalk will be creating test cases to ensure that our business rule has been achieved by working with our client and as well to validate that each unit of the software performs as designed. Secondly, for integration testing, it is necessary to examine the modules when integrated to verify that they work as expected to test the modules which are working fine individually and to expose any issues when combining these components together to make sure that the interactions between the different components of the software run smoothly without any dilemmas. Lastly, usability testing determines the success of the system. Which is necessary to indicate if the application is user-friendly or not and was comfortably used by an end-user or not. The main focus of this testing is to check whether the end-user can understand and operate the application easily or not.

Throughout this document, we discuss the plan for testing the iSpeak project in detail and explain how these testing methods offer suitable test coverage of the entire project. Information on the architecture of the system and other components implemented into this application is detailed in our Software Design Document. We will begin our analysis of our testing plan with unit testing.

# 2.0 Unit Testing

Unit testing is a method in which we will test each part of our application individually. Using controlled input, we try to ensure that the specific portion of our application that we are testing will perform as expected, i.e. correct calculations, correct page setup. To enhance our testing process, we will be using Flutter's unit testing package. This package works as a dependency to our flutter application and allows us to create test files to work on our classes. Furthermore, our testing will mostly be black-box testing, as we will mainly focus on ensuring that data received from the database translates directly to a desired output. However, we will still have some white-box testing to look into some conditional paths that our application employs when setting up a page. Here are the list of sections we will be testing:

- Mobile Application
  - Lesson Setup
  - ASR Download and Run
  - Achievement Reception
- Website
  - Course creation/modification
  - Student analysis

Each section will contain a number of units pertaining to that function.

## Mobile Application:

This section outlines the main functionality of the mobile application. Since outside of some functions, the main purpose of the application is to take data and translate it into a complete course that is able to be taken and completed.

### Lesson Setup:

Lesson setup has these units, and data will be provided directly rather than through a database:

**Login:**

For this, we will simply login using a valid account as well as an invalid account. Expected results are if the account is valid, it will continue on from the login page, while an invalid account will stay on the page until a valid account is entered.

**Course Setup:**

In this scenario, we will provide the application with preset course information. This information will contain data at the course, module, lesson, and task-level, including every type of task. Expected results can be verified visually; The application should be able to click into the course, then into the modules, then into the lessons, and lastly through the tasks. At each task, the information provided, i.e. question prompts, videos, pictures should be formatted and displayed according to our design.

### ASR Download and Run:

For these units, we will focus on the verification on the downloading and use of the ASR models

**Download:**

Using Flutter's built-in storage viewer, we can verify that the ASR model has been downloaded. Furthermore, due to how the ASR package requires a downloaded model when loading a model to use, we can also verify the download process this way as well.

**Run:**

As for running the model, we will look at functionality rather than correctness, as ASR technology will not always be able to recognize the right word. However, once the package does recognize words, we can verify it to be working properly.

### Course Run and Achievement Reception:

In this section, we will focus on the correctness of the achievement system in regards to course progress.

**Achievements:**

In this test, we will proceed through a predesigned course with relevant tasks, i.e. tasks that count towards total correctness such as multiple choice tasks. We will do a number of runs through this course with varying levels of correctness. This will allow us to verify the different achievements awarded based on correctness percentage. Finally, upon completion of this run, an expected result will be that we will get awarded a completion achievement.

## Website:

To complete the test of web units, we will create a complete course with some related functions interspersed. The whole test is divided into two parts: creating courses and observing the results of student tasks.

## Creating courses:

Creating a complete course includes the following parts:

### Log in to the teacher account:

To open the web program, we need to enter the login credentials and click to log in when the input is complete. The program will compare the entered account and password with the account password stored in Firebase. When the two are the same, we can log in successfully. After the login is completed, the web program will read and display the course information of the currently logged-in account. If the account password does not match, it will stay on the login page without change.

### Create a course:

Click the Create Course button. The program will open a new page, we need to name the course and describe it. After the course is created, we need to create a module. The process of creating a module is similar to creating a course. After the Module is created, we can now assign tasks to students. Click Create task, and then we select the type of task we need to create (pronunciation task, video task, multiply choose task...). Gradually add the tasks that we want students to complete (in different tasks, the corresponding functions are different. For example, multiply choose task, we need to select the image corresponding to each option to upload. For the video task, we need to upload a video link). After setting up a complete task, click Create task. At this point a task has been created, we can open Firebase to check. In Firebase, click on the teacher account you just logged in to, and click on the created course, module, and task in turn. At this point, the task data just created should have been stored in Firebase.

### Modify Course information:

What needs to be tested now is the modified function of the program (whether it is possible to modify the course data and add students to the corresponding content). The test of this part of the function can be compared with the data changes of Firebase for testing. The specific steps are to add student information in different modules (Course, module, tasks) and observe whether the Firebase data changes accordingly. If every added or deleted student can find the corresponding

change in FIrebase, the module test of the added student is successful. Then, starting from the task, gradually delete and rename the created content. If the Firebase data changes accordingly, the functional test for modifying the course data is successful.

## Student Analysis:

**Observing single student:**

We can check the result of this student's task by selecting him (for example, the task completed by the student is ENG105-Module1-Task1, then click on this task, and we can see all the completed students). If you click on the student, the scores obtained by the student and the recorded audio for the student to complete the test are displayed, then the unit test for viewing the results of the student task is successful. At this point, there is an input box below the student's result. This input box can enter the teacher’s feedback, which can be seen in the mobile app.

**Observing students result populated:**

We can see the distribution of student test results in Firebase. The specific step is to click on the task you want to view in Firebase, Firebase will display the distribution of all student results in the form of pictures, and there will be a table corresponding to it.

# 3.0 Integration Testing

## Overview

Integration testing is the process of combining and testing the core modules of a software system, focusing specifically on their interactions with each other. Integration testing ensures that each component of the system can be effectively combined in a fluid manner free from major issues preventing their usage as a cohesive unit. The overall strategy of the integration testing is to ensure that the proper values inputted to each module are correct and indicate optimal functionality of the three core components of the software system.

## **Integration point 1**: Website Dashboard and Database integration

The website dashboard must be integrated with the Firebase backend services in order to send and receive data to/from the mobile application. Thus, the website is responsible for completing CRUD operations by utilizing the FlutterFire plugins integrating Google Firebase services with a Flutter Web application. The core method of testing this integration is by utilizing the Google Firestore web console to ensure that data pushed to Cloud Firestore is correctly formatted and ready to be parsed by the mobile application.

There is no formal test harness for this process, as much of the testing is easily handled by ensuring proper values are pushed to the Firestore database via the Firebase console and associated error/informational messages. One challenge with testing the Firestore database is

## **Integration point 2:** Mobile Application and Database integration

The mobile application is responsible for communicating with two key modules: the website dashboard and the ASR library, Vosk. The first integration is covered in this section, as the mobile application will receive and send data through the Google Firestore database in order to properly communicate with the website dashboard.

Since the website dashboard is responsible for the creation of the course hierarchy, the mobile application is responsible for loading these created courses and associated objects for the learner users to complete. The first step of testing the integration of these components is to verify that

the course structure created by the designer user on the website dashboard is accurately displayed to the learner. The tests will show any obvious errors in communication through the Firestore database as tasks will be missing features inputted by the designer if the integration fails. The loading and displaying of created courses thus makes up the first major point of integration between these two modules.

The second mobile to web integration is the uploading of learner responses to completed lessons and tasks. The mobile application must upload the audio files and selected answers to the website dashboard via the Google Firestore database. This communication will be tested by reviewing the responses from the learner on the website for a given lesson. If the integration fails, the learner responses will not be displayed properly to the course designer for review on the website dashboard.

## **Integration point 3:** Mobile Application and Vosk ASR library integration

The final core integration is between the mobile application and the ASR library, Vosk. These modules are connected via Dart channels which allow the mobile application to communicate with the Vosk speech recognition functions, which are programmed primarily using C++. These Dart channels are implemented separately for both the Android and iOS platforms and must be tested to ensure that proper ASR data is sent to the mobile application, which will then upload the speech recognition results to the website through the Firestore database.

The integration of Vosk into the Flutter mobile application will be achieved by calling the speech recognizer functions that will record and analyze learner speech as they answer production (speaking) questions. The Vosk recognizer functions will communicate the recognized words and sounds back to the mobile application to be recorded as part of the learner's response to the presented task. These responses will be sent to the website for review in order to analyze the ASR data and words recognized by Vosk. Additionally, the integration of these modules will allow for the mobile application viewcontrollers to display the recognized words and sounds to the user, so they can learn from repeated attempts to pronounce the word or phrase presented to them. The accuracy and efficient integration of ASR into the mobile application is the primary goal of testing these modules.

# 4.0 Usability Testing

Usability is defined as the amount of time it takes users to complete a task using mobile and web applications. Usability testing will be conducted via user testing of the software product. This will be quantified by recording the amount of time it takes user testers to perform key use cases of the system. For the mobile application, we will ask users to perform several tasks:

- ➢ Please create an account through the mobile application. Time expected: 2 minutes.
- ➢ Please log in to the mobile application. Time expected: 1 minute.
- ➢ Please navigate to the course module you have been assigned to by a course designer. Time expected: 1 minute.
- ➢ Please click on the first module listed in the course: Time expected: 30 seconds.
- ➢ Please navigate to the first lesson of the module: Time expected: 10 seconds.
- ➢ Please start the lesson and complete the first production task: Time expected: 2 minutes.
- ➢ Please move on to the next task, which is a perception task. Complete it and submit your answer. Time expected: 2 minutes.
- ➢ Please continue and finish the remaining 8 questions in the lesson. Time expected: 5 minutes.
- ➢ Navigate back to the lesson page. Time expected : 10 seconds
- ➢ Click on feedback and view the given feedback for lessons with feedback. Time expected: 1 minute
- ➢ View associated achievements for tasks. Swipe left for badges and right for achievements. Time expected: 1 -2 minutes
- ➢ Press home to return to the main screen. Time expected: < 1 minute

These tasks outline the primary use cases of the mobile application. We will accept the state of each use case if 80% of users can perform the task listed in the expected time. If less than 80% of users can complete the task, we will need to either adjust the timing goal or improve the user interface and tutorials to make sure users understand how to perform each task efficiently. These timing windows will likely be refined and discussed after early user testing.

For the website dashboard, a similar process will be used. This time, there will be different tasks that are specific to course designers and their use cases. The desired tasks are as follows:

- ➢ Please create an account through the website dashboard. Time expected: 2 minutes.
- ➢ Please log into the website dashboard. Time expected: 1 minute.
- ➢ Please create a course titled ENG 400. Time expected: 2 minutes.
- ➢ Please navigate to your new course ENG 400. Time expected: 30 seconds
- ➢ Please create a new blank module within your course. Time expected: 2 minutes.
- ➢ Please add a new blank lesson to the module. Time expected: 1 minute.
- ➢ Please add a blank production task to the lesson. Time expected: 1 minute
- ➢ Please add a question to the production task. Highlight and underline two words of the question. Time expected: 1 minute.
- ➢ Please add an expected answer to the production task. An audio recording of the word. Time expected: 2 minutes.
- ➢ Please add a new perception task. Time expected: 1 minute
- ➢ Please add an expected answer to the perception task. This could be a multiple-choice selection, free text answer, or short answer to be matched. Time expected: 1-2 minutes.
- ➢ Please add a new instruction task, detailing some information that the user should learn about. Time expected: 1 minute.
- ➢ Please confirm your lesson with the three tasks. Time expected: 1 minute.
- ➢ Please toggle the visibility of the lesson to the registered users of the course. Time expected: 1 minute.
- ➢ Please access the associated ASR feedback data from the production task. Time expected: 1 minute.

These tasks outline the primary use cases of the web application. We will accept the state of each use case if 80% of users can perform the task listed in the expected time. If less than 80% of users can complete the task, we will need to either adjust the timing goal or improve the user interface and tutorials to make sure users understand how to perform each task efficiently. These timing windows will likely be refined and discussed after early user testing

# 5.0 Conclusion

SmartTalk is devoted to providing our client and her team with a functional, convenient tool for their research purposes. In order to make sure that our product is working as intended with all the implementations that have been made. We are thinking unit testing would be the best option to test an individual component to make sure it functions properly and does not have a large number of bugs to the best of our ability. For example, for the mobile and web applications, we are going to ensure that the business logic makes sense of all the data being exchanged through the database. As for integration testing, we will test the main three components which are Automated Speech recognition (ASR), mobile & web framework to make sure that these components are fully connected to each other as a unit. This testing will allow for a fully interactive learning experience for all foreign language learners, with a focus on pronunciation; learners can verbally interact with the mobile application through ASR, and once modules are finished, the designer will be able to give constructive feedback to the learner based on data user response data. Lastly, we are aiming to have our client test this product to ensure everything is functioning properly and as well to get the product they were looking for.

With these criteria as a guide, we plan to do exhaustive testing of the “iSpeak: a Gamified Mobile Language Learner Application” project in order to deliver a functional and impressive product to our client. Once tested, we are confident that we will be able to provide Dr. Kang with a product she will be satisfied to give to her graduate students as a tool for research, and that will function also as a more entertaining and gamified learning experience for future language learners.