**Test Plan**

Team 2 - Soft Skills Application

Nathan Anderson

Jonathan Delira

JuSeung Park

Flor M. Lopez

1. Introduction:

In order to ensure the correct functioning of the Soft Skills Training System the team has developed strategies to test each one of the components of the System. The system is composed of an Android application and a Website/Server. Each component will have it’s individualized description of how it will be tested.

1.1 Major Testing Issues:

1. Application

Since the Android application is the most interactive part of the system, the major issue that the testing will present is the development of a suitable unit test that works at the moment of integrating the components (the modules and their Java code). The separate internal HTML modules will need to be tested outside the app with stand alone JUnit tests.

1. Server

A major testing issue for the server is to test the network connection of the application-server and website-server to prove that it is stable and continuous and provides the proper access to the data. Apart from that the server needs to have its access and authorization restrictions tested.

1. Website

The website is a well known domain, and the Soft Skills Training Website does not have any particular function that uses anything new. The website will be able to run on any web browsers, so the browser testing needs to be done. Other than browser testing, there will not be any serious issue for the website.

2. Testing Strategy/Responsibilities:

For the Android application, Flor & Jonathan will check how the app handles all close states. In applications build in android studio with standard android API, the entry points and flow of how the app behaves is standard. This can be done by an operator using a debug deployed app. The tester will leave the app at various major parts and attempt to return to the app. If it fail to continue at the previous point the app needs to restart at applicable stable points.

We will test the functionalities that are specific to the system’s logic with JUnit test methods. We will design a method that sends in predetermined information to component and verify its result. Also JUnit test can be used to test the functionality of server communication. The server should provide a back end API just for testing the android app. The functionality of this will be that in normal operation of the server there is authentication and authorization restriction to data, and what data user sees is dynamic. This situation is not ideal for testing so an API that allows for tests to be ran on all externally networked devices with generic data. The system is slightly more complex in this regard due to the fact that HTML modules are dynamically ran. This is a situation where later developers may design a HTML module that has functionality not anticipated by the current design team. Out documentation will have to specify what functionality of HTML we can and can not support. The training module that the Android app runs will be HTML and the functionality supported for the modules in the system will be tested by Nathan & Juseung on a stand alone Java JUnit test.

The android applications functional testing will be done via Monkey tool, which supplies pseudo-random streams of user events.

The Web server, which will be tested by Nathan & Juseung, will primarily use JUnit test methods to check functionality. The testing of the web server can easily go beyond the methods ran from the Eclipse IDE where the server is developed because the JUnit test can be ran in a stand alone android app and other remote PC’s. This remote PC JUnit test can simulate any function of web service and access to the server as a whole. The web test tool HTMLUnit has a capability to simulate a full web browser experience and can attempt to check access and authorization of the system. The process will be to simulate logging in as an unauthorized user and attempt to access pages and information that the given privilege would not allow. Similarly, a second test will be ran that has privilege and that test will attempt to access what it should be allowed to access and verify it is successful.

The concept of using JUnit and HTMLUnit in testing is meant to facilitate regression testing. These test tools will be able to be set up and ran repeatedly as development progresses. They will attempt to ensure that new versions of the code will be just as functional and reliable as the previous versions. Only minor additions to the unit test will be needed to accommodate testing of new functionality.

2.1 Testing Features:

Android app:

* User input per activity
* Frame rate per activity
* Client-server connection
* Android versions that support the application.

2.2 Non-Testing Features: Comprehensive video list

3. Test Environment:

Android app:

* Component tests:
  + JUnit allows us to create individual units of code and test their functionality.
* Performance tests:
  + Dumpsys tool outputs performance information relating to animation frames.
* Stress tests:
  + Monkey generates pseudo-random streams of user events. Allows to run tests in a physical device connected to the pc pr in the emulator.
* Server OS: cs448 provided Linux server.
* Web Server: Apache
* Database: MySQL. For testing purposes, a PHP script was created a to verify the connectivity and access to the database to gather the required data in every activity.

4. Performance and Stress Testing:

The performance tests will require the simulation of slow data connections and data connection breaks. We will provide in the server testing API the access to data that will intentional break, or download at various predetermined speed. Of which the slowest speed should trigger a failure in the android app. The application will have to handle gracefully the failure of data connections gracefully and return the user to stable points in the app experience. Likewise the JUnit tests that simulate connecting with the server will simulate broken connections and make sure the server does not hang up or crash. Also standard protocol-based mobile load testing by recording network traffic, with proxy based recorder or alternative, between device and server, replaying network requests for large number of virtual users and analyzing results.

5. Project Results:

With the android app, our main goals for testing were to improve UI performance, through review and repair of frame rates, and ensure robustness, by supplementation and examination of an arbitrary flow of user events. For example, Dumpsys, an android tool, supplied frame rates per activity, above and below the desired rate. Furthermore, the data provided, in CSV format, was effortlessly applicable to display and analyze in a spreadsheet. Another helpful android tool known as Monkey supplied random user actions for every activity within the app and results detailing actions enforced per activity and the app’s response(s) to such actions.