**CITS3200N Acceptance Test Portfolio**

I, Grant Landers have reviewed and approve of the Acceptance Test Portfolio, where the information outlined is accurate and suited to my needs. I accept this document as the standard by which I will accept the final product.

Client Signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Objectives**

The purpose of this document is to outline the testing environment around which our project will be based. The acceptance tests below are objectives with which this project may be broken down, reflecting the main functional requirements as specified by the hundred dollar test (RAD section 3.2). Passing these tests is an indication of the successful functionality of the project.

**Document References**

This test makes reference to the requirements analysis document (RAD). The sections on functional requirements (section 3.2) and non-functional requirements (section 3.3) were assessed in creating the tests

**Test Summary**

The system is a data collection and processing app. Users must interact with the iOS app user interface, which successfully collects lifestyle data. This information is then pushed to a data hosting service which stores the information in the long term. This server must then be able to interface with an administrative interface; a PC compatible application that allows the administrator to receive and review data.

**Testing Strategy**

The testing strategy must deliver four main goals. Firstly, each of the subsystems (user interface, server, and administrative interface) must be tested independently for acceptability. Each component must be able to perform its entire function to a rigid standard. Finally, the system will be tested in its entirety, ensuring that the components function cohesively with one another.

Testing will be aimed at system interfacing; ensuring that information is delivered in an appropriate, compatible manner between user and subsystem or between subsystems.

The testing for the independent subsystems will be performed by the groups testing manager(s) and the developers in charge of that respective subsystem. Testing for the entire system will be done by the testing manager(s) alone.

## Tests are ordered from most important to least important.

## **Test A**

This test is to ensure that the server can correctly communicate and interact with the mobile application. It will check the functionality of normal operation procedures, as well as how it handles unexpected or malformed input.

### **Test Specification**

The following criteria are required to pass this test:

* The server can be reached over the internet (ping test)
* The API is running and accessible
* User surveys (MTDS/training volume/sleep) can be successfully sent to the server, stored, and then retrieved with the survey data remaining in-tact (idempotence)
* The server only responds to users of the system; it does not return any useful information without a proper login
* The server will reject malformed input appropriately (sufficient error-handling)

### **Test Description**

This test will be conducted remotely to the server (over the internet). Part of the test will involve manually interacting with the server interface (API) via a normal web browser. This is to ensure that the interface is robust and can handle various forms of input. Subject to time constraints, and if deemed necessary, automated unit testing of the API may be an option.

Input data:

* Sample user (email/password)
* Sample survey data (MTDS/training volume/sleep)

Output data:

* The server response (JSON formatted)
* Examination of the data stored on the server

Testing procedure:

* Determine the URL or IP address of the server.
* Ping the server.
  + No response → test failed
* Enter the URL of the API (e.g. <http://server/api>) and check that it responds as expected. This includes checking the HTTP return code.
  + No response or invalid response → test failed
* Login to the API with the sample credentials. Check that the server accepts the credentials and responds as appropriate.
  + Rejection of credentials or invalid response → test failed
* Simulate the submission of various sample surveys to the server. Check that the server accepts this data. Check the response.
  + Rejection of sample data or invalid response → test failed.
* Submit various sample surveys that are known to have invalid responses. Check that the server rejects this data. Check the response.
  + Accepting the data or invalid response → test failed.
* Logout from the server. Check the response.
  + Invalid response → test failed.
* Attempt to request data specific to the sample user (private data). Check that the server rejects the request (already logged out).
  + Returns user specific data → test failed
* Using the administrative interface, check that the saved responses are equal to what was submitted.
* Perform various permutations of requests against the API manually. Check that the response is as intended.

### **Test Analysis Report**

TBA (once test has been conducted)

**Test B**

Test B will assess the usability of the system at the user and administrative interfaces. It will test how the users will interact with the system and whether or not they are easy to understand.

**Test Specification**

For this test to pass, it is required that:

* All aspects of the app are properly labelled and can be navigated easily
* The app specifies all tasks which must be completed, reminding the user when they have missed something
* All functions of the app require little effort to learn and complete
* The administrative interface delivers info in a readable fashion and is easy to understand

**Test Description**

This test assesses the usability of the user components of the system. It will involve people who have no prior knowledge of the project attempting to use all of its functionality based on intuition alone. This is known as beta testing

Output Data:

* Feedback from random users, detailing parts of the system which:
  + Are acceptable and straightforward
  + Cause confusion and/or incorrect usage
  + Do not function properly or as expected
  + Take too much time to perform or are inefficient

Procedures:

* Give random users access to the system via iPhone or emulator (user interface) or PC (administrative interface)
* Allow them to use the applications without intervening
* Once they have used the applications to their entirety, ask them feedback questions based on the output data listed above

**Test Analysis Report**

TBA (once test has been conducted)

## **Test C**

This test is to ensure that the administrative interface has basic functionality.

### **Test Specification**

The following criteria are required to pass this test:

* The interface only allows authorised users access
* Users can be added/removed
* Survey data can be downloaded and saved appropriately (e.g. CSV format)

### **Test Description**

This test will be conducted remotely to the server (over the internet).

Input data:

* Sample administrator login (email/password)

Output data:

* Exported survey data

Testing procedure:

* Navigate to the administrative interface (e.g. <http://server/admin>). Ensure that it is accessible.
  + Not accessible → test failed
* Attempt login with invalid credentials.
  + Accepting invalid credentials → test failed
* Attempt login with the sample administrator credentials
  + Rejecting the credentials –> test failed
* Check that the interface is laid out as intended. Check that there are navigation options to add/remove users, to view survey data and to export that data.
  + Malformed interface → test failed
* Check the add/remove user functionality. After adding/removing a user, attempt login with those credentials on the API
  + Login possible with deleted credentials or login not possible with added credentials → test failed
* Check the ability to view survey results and the ability to export that data
  + Cannot view or export data → test failed
* Logout of the interface. Check that it is not possible to access any part of the administrative interface.
  + Able to access restricted areas without credentials → test failed

### **Test Analysis Report**

TBA (once test has been conducted)

**Test D**

Test D is to ensure that sufficient error checking is in place during user interface so that the system is not given false values or incompatible data

**Test Specification**

The requirements for test A will be tested for acceptance:

* The system errors and alerts the user when data has not been entered and must be. This will prevent users from sending incomplete data
* The system errors and alerts the user when data has been incorrectly entered into a field. This will prevent any incorrect data from entering the server, reducing the chance of errors later in the system

**Test Description**

This test is conducted by manually inputting data into the app and attempting to send this to the server. This will test that the system recognises the distinction between correct and incorrect data and displays error messages in the case of incorrect data.

Input Data:

* Correct data to test that data is pushed to the server
* Incorrect data to test that the system displays an error message and doesn’t push data to the server
* Lack of data entered (blank fields) to test that the system displays an error message and doesn’t push data to the server

Procedures:

* Exercises are entered into the ‘diary’ in a correct manner, progress is saved
  + Expected result: no error
* Exercise is entered into the ‘diary’ in an incorrect manner (e.g. letters in number field, special characters in field, infinite characters in field)
  + Expected result: an error message is displayed next to the relevant field before input can be completed
* Exercise is entered into the ‘diary’, only some fields filled out
  + Expected result: an error message is displayed next to the relevant field before input can be completed
* Repeat for all other tabs in app (goals and schedule)

**Test Analysis Report**

TBA (once test has been conducted)

**Test E**

Test E will assess how many users may use the system at any time. The hosting server may only facilitate a finite amount of users at any one time. This test will quantify this value and ensure that it surpasses an appropriate threshold

**Test Specification**

As an upper limit, the server must be able to facilitate at least 60 people at any one time, it required that:

* Multiple people may use the database simultaneously (users and administrators)
* Multiple people may submit data at the same time (users and administrators)

**Test Description**

In this test we will analyse the systems response to multiple access attempts at a single time to replicate high data traffic. This test will be done in two ways: firstly by emulating the user and administrator input by automating instructions to the database. And secondly, by using real devices at the same time to create a real world scenario

Input Data:

* Server input commands replicating those of the user and administration data pushes
* Actual pushes of data from user and administrator applications

Procedures:

* Use an automated environment to send commands to server, Starting with small data size and frequency and progressing to larger, faster tests. (test various types of commands and frequencies)
* Use iPhone/emulators and PC to send data from user and administration interfaces to database at the same time. Continually input exercises etc. and save progress

**Test Analysis Report**

TBA (once test has been conducted)

**Test Materials**

In order to perform all of the tests to satisfaction we will require access to multiple computers and mobile devices that have access to the system and the internet. These tools will be provided by the development group and will not require any external resources or expenditure.

We will also require access to a sample group of people to beta test the system. The development group will also provide applicants for testing unless the client provides their own sample group and believes that to be more appropriate