Team Wow

Food evaluation/Testing Software

Client: Prof. David Gee

Members:

Geoff Worley

Shaina Greer-Short

Frank Senseney

Connor Taylor

Advisor: Szilard Vajda

Table of Contents

Introduction

1. Purpose

This document lays out the general design of the standalone webapp for recording and creating tests for evaluating food samples. This will explain the intended users for the app, how the it will be used, and the tools and resources the developers will use in order to fulfill the software requirements. It will include high level overviews of user's using the app and what they will see when interacting with the app, as well as low level explanations of the tools and programming concepts used in order to facilitate the functions of the app.

2. Scope

There are main software features that need to be implemented. There needs to be a system to keep track of users, test users and admin users. There needs to be the main testing system that accepts the input of the tester and sends ties the result to that test to later be converted into an Excel sheet. There needs to be a system for the admin to create tests, to manage user groups and tests. There needs to be a system to take all of the user inputs for a test and create an excel sheet for that test with the results.

The software won't be doing any analytics or augmentation of the data submitted, only the translation from collected data into the excel sheet. Because this is for educational purposes, the checking of patterns and data analysis is to be done by the students rather than the software.

This implementation will speed up the process from recording data into an excel sheet. Before this implementation was devised, students had to manually record their tests on food samples, and then record them manually into an excel sheet.

3. References

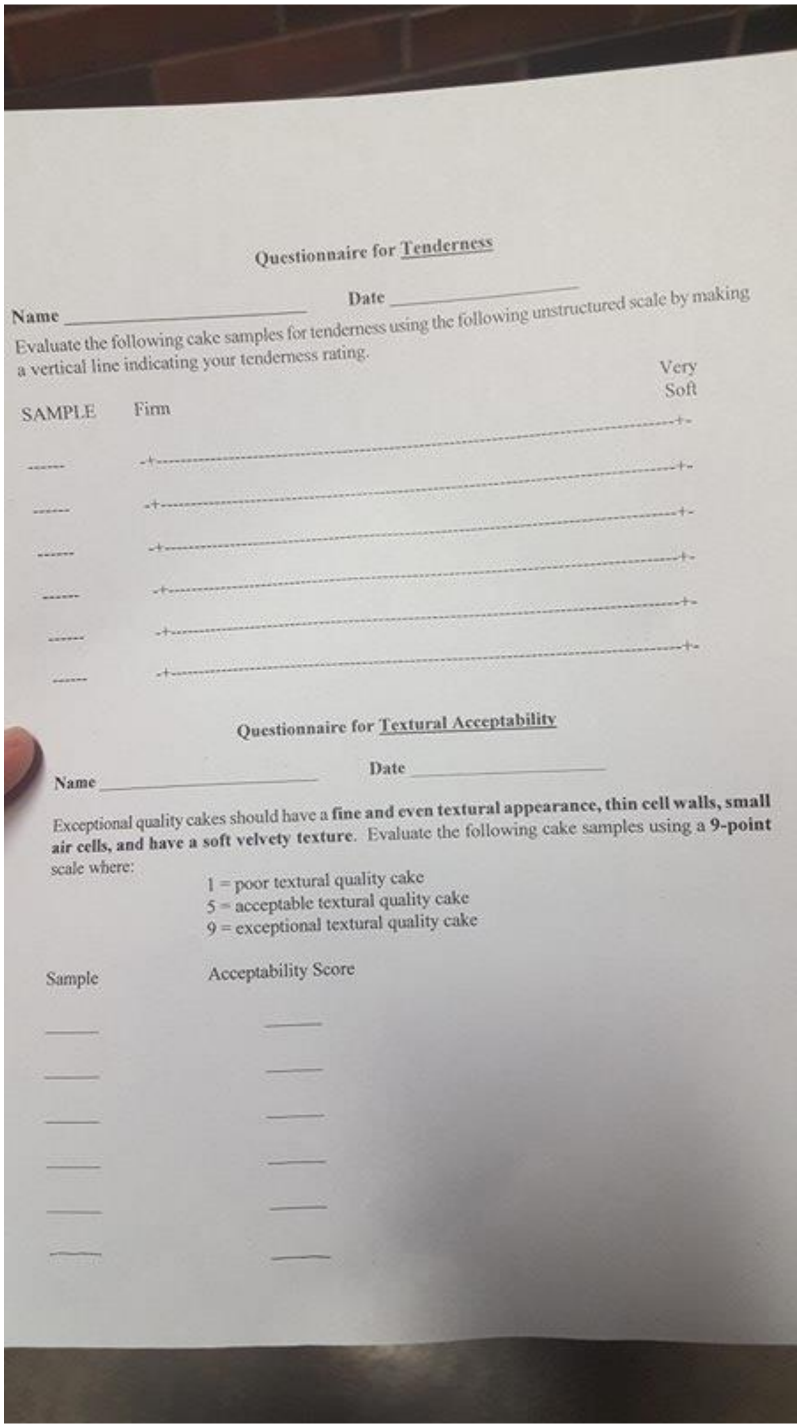
https://www.schoolhousetech.com/test/

4. Overview

The rest of the SRS will go over the high level overview and the low level requirements of the software requirements. I've also included UI mockups and diagrams that demonstrate the processes performed by users and workflows. The mockups provide a visual and conceptual understanding of what the users are going to be doing and seeing. This is displayed first to create a frame of reference for the rest of the document

Workflow and UI Mockups

Original Test Input Form



Questionnaire for <u>Tenderness</u>

Name ______________________ **Date** ________________

Evaluate the following cake samples for tenderness using the following unstructured scale by making a vertical line indicating your tenderness rating.

SAMPLE	Firm	Very Soft
------	-+------------------------------------	------------------------------------+-
------	-+------------------------------------	------------------------------------+-
------	-+------------------------------------	------------------------------------+-
------	-+------------------------------------	------------------------------------+-
------	-+------------------------------------	------------------------------------+-
------	-+------------------------------------	------------------------------------+-

Questionnaire for <u>Textural Acceptability</u>

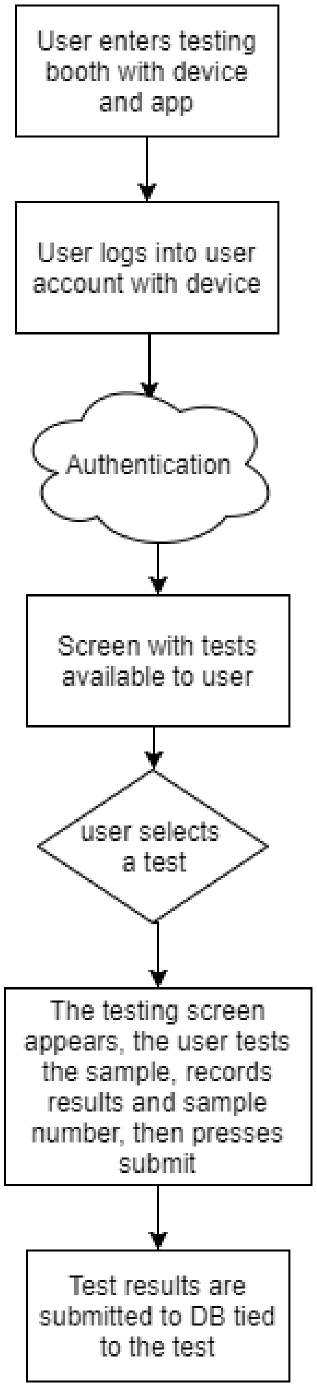
Name ______________________ **Date** ________________

Exceptional quality cakes should have a **fine and even textural appearance, thin cell walls, small air cells, and have a soft velvety texture.** Evaluate the following cake samples using a **9-point** scale where:

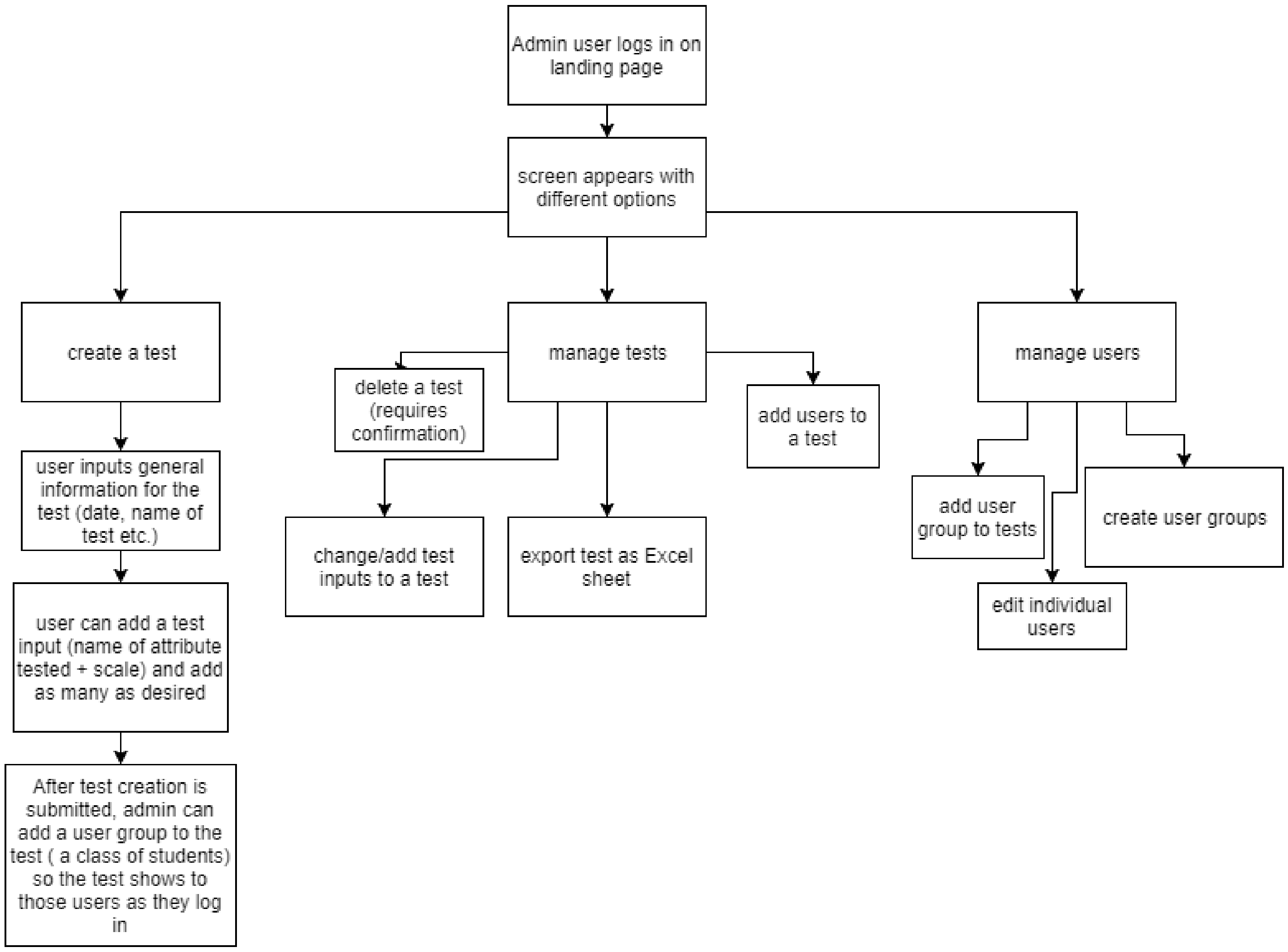
1 = poor textural quality cake
5 = acceptable textural quality cake
9 = exceptional textural quality cake

Sample	Acceptability Score
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Testing User Workflow

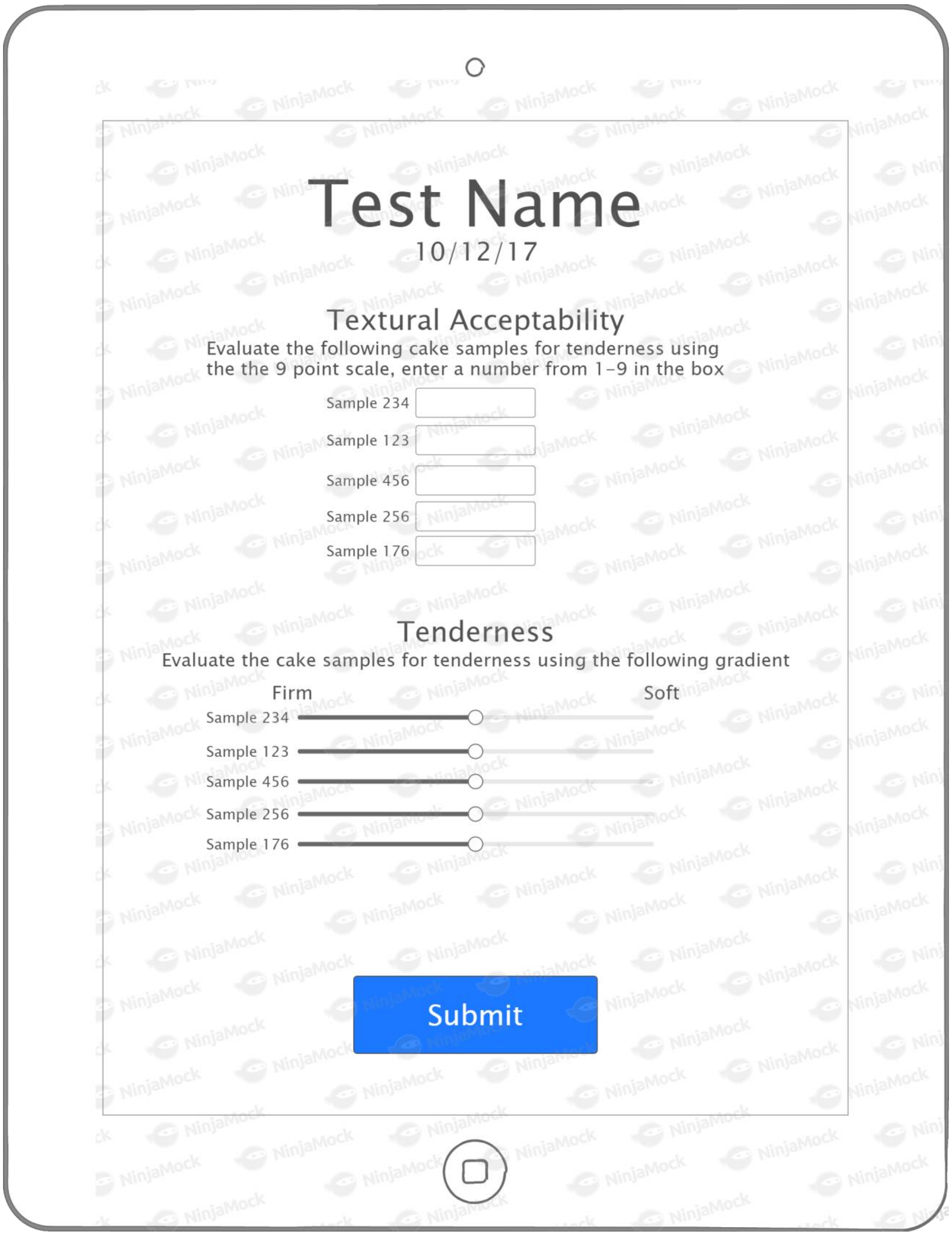

User enters testing booth with device and app
User logs into user account with device
Authentication
Screen with tests available to user
user selects a test
The testing screen appears, the user tests the sample, records results and sample number, then presses submit
Test results are submitted to DB tied to the test

Admin Workflow



Admin user logs in on landing page
screen appears with different options
create a test
manage tests
manage users
delete a test (requires confirmation)
add users to a test
user inputs general information for the test (date, name of test etc.)
change/add test inputs to a test
export test as Excel sheet
add user group to tests
create user groups
edit individual users
user can add a test input (name of attribute tested + scale) and add as many as desired
After test creation is submitted, admin can add a user group to the test (a class of students) so the test shows to those users as they log in

Test UI



Test Name

10/12/17

Textural Acceptability

Evaluate the following cake samples for tenderness using the the 9 point scale, enter a number from 1-9 in the box

Sample 234	
Sample 123	
Sample 456	
Sample 256	
Sample 176	

Tenderness

Evaluate the cake samples for tenderness using the following gradient

Overall Description

1. Product perspective

This software is a standalone product. There are a few interconnected systems. At it's core, the software is meant to create and administer tests. There are a few other systems that will need to be introduced in order for the implementation to work in the academic setting, and with the exporting capabilities required. There will need to be systems for user authentication, a database to hold results from tests, a way to connect out user database to a test. These systems can be focused into 4 categories, user, exporting, recording and creating.

Some software suites exist with this kind of functionality, but they are more geared towards creating academic tests, rather than testing inputs. One example is Schoolhouse Test 4, created by Schoolhouse Technologies. While the software suite would fulfill the function of creating tests, there wouldn't be a linkable online component to export the collected data.

User Interfaces

There are two main interfaces within the software, The test interface and the admin Interface. The test interface is mainly utilized by students that are going to be testing and recording samples. The admin interface is going to be used by the instructor of a class to create tests, moderate user groups, and to manage and export tests

Hardware Interfaces

The software is mainly going to be used on tablets and PC's. The testing interface and functions will mainly be used on the tablet while the PC will be used when the admin is making and managing tests.

Software Interfaces

The software is going to be developed as a webApp, so a hosted web page with a URL to access it. We will need to use backend linked to a database, as well as many front-end interactions to translate sliders and user inputs into data to be recorded. I believe a noSQL database will be the easiest to work with this solution. There won't be a huge need for complicated query patterns, and most data can be manipulated in the excel sheet anyway. The use of a front-end framework like Bootstrap will also be used. This makes managing the different screen sizes easier between mobile, tablet, and PC screens.

Communication interfaces

There will need to be communication between user input and the database, as well as network protocols in place in order to access the URL. Users will also have to communicate with the backend in order to authenticate their logins.

Memory Constraints

Issues with memory should be rare. With regular testing, there would be at most for or five submissions to the back end at the same time.

Operations

I'll separate this section into testing user and admin user

Admin – The admin user will have a lot of operations and control. While managing tests, they should be able to initiate the creation of a test, delete an existing test, tie user groups to tests, and export existing tests into excel sheets. With test creation, there are a multitude of functions. The addition of scales and descriptions, which user groups will initially be added to the test. While managing users, the admin should be able to create and delete individual users and user groups. For any act of deletion, confirmation should be required.

Testing- The main interaction a testing user will have with the software is typing to log in, manipulating scales, and submitting their evaluations. They possibly can export the results of tests they have participated in as well.

2. Product function

The main product functions include:

- Recording test results
- Creating evaluation tests
- User functions (log-in, tie users to tests)
- Exporting tests in Excel format

3. User Characteristics

This section will be separated into testing user and admin user

Testing user

These users will be college students in nutritional classes. The will most likely have some computer experience, enough to do the fucntions required of them. They'll only really be logging in, performing and submitting their tests, and then exporting their tests into an excel sheet.

Admin User

An admin user will most likely be a university professor or class TA. They will have sufficient computer experience in order to create tests. The goal is to make it as simple and pain-free as possible so that even the computer illiterate would easily be able to create a test, and have testing users tied to that test.

4. Constraints

1. Regulatory policies

There are certain limitations that are might occur with hosting the application on the CWU network. These are currently to be determined. We anticipate there may be database limitations.

2. Interfaces to other applications

There may be some issues with our software interfacing with Excel. We will need to make our software flexible for tests with a small number of inputs to appear in the same format as a test with a large number of inputs. We'll need to be able to export our database data and translate it into an excel filetype as well.

3. Criticality of the application

This software will be essential to the fucntions of the class. While there is a backup manual solution, this solution will make the manual process become obsolete. It needs to be up and running for students when they input tests, and for whenever an admin is performing admin functions.

4. Safety and security considerations

The website will most likely need to be https compliant at the least. There is some concern with false log-ins from users, so a password will need to used. Two factor authentication for the admin is a possibility. If we do use CWU to host, there may be a possibility to use Student logins from the MYCWU service. Using these most likely comes with increased security concerns as well. These are to be determined at the moment, it will be discussed with CWU.

5. Assumptions of dependencies

The main assumptions for this software come from the devices that each part of the software will be accessed on.

Requirements

1. Interfaces

The interface requirements will be separated into testing user interface and admin user interface

Testing User Interface

The testing user interface will include a log in screen, some screen to select the test to take, the actual test screen, and some screen to select an option to export a test as a spread sheet. The user experience flow most likely will happen as a user navigating to the URL that hosts the app, and then it would show the landing page that would include a log in screen. After the user logs in, a screen would appear that have the tests available to that user. This would most likely take the form of a list that covers most of the screen, it can use a lot of screen real estate because there isn't a lot of functionality that the user needs. After the user selects a test, there will be an option menu that appears to either export a copy of the test, or to record their samples for that test. When a user is submitting test results, there will be their various tests they need to record and then a submit button at the bottom of the screen. Some error or notifications would appear during certain errors. These will most likely appear as centered windows, and the page will fade to slightly darker to accent the error message.

Admin User Interface

The admin user interface will start on the log in page as well. After the admin logs in, there will be a page with 3 separate sections. There will be a top section with some links related to certain functions that are performed most often. Links that say "create a test", "export most recent test" will be some of the available options. Below there will be 2 columns. One column will display an ordered list of the most recent tests created, listed by date. The identifier for each test will be either a code, or a date. More deliberation with the user of the admin functions is required to see how they want the recent tests ordered. The other column will include buttons for the other admin functions. These include managing users and managing tests. The UI for test creation will be very malleable. Each addition of a condition will follow the same format. It will be a faded button to add an additional test condition, and then after they add a description for the condition, and select a scale, then that same faded button will appear under it to add additional conditions for a single test.

2. Functions

1. Validity checks on the inputs

The main validity checks will be the user log-in. We will have to compare the user input on the log in screen with the user data in our database. These will need to line up in order to allow a log in, otherwise there will be an error message. For test inputs that allow text input, like a number scale, we'd have to check the input to make sure it's within the correct input. It's possible to negate this by using a drop down menu in order to prevent users from erroneous inputs. This system depends on the amount of precision required. Admin will also have a lot of confirmation for certain actions like deleting of users and tests.

2. Exact sequence of operations

The regular test user workflow will involve the following:

1. Navigates to URL
2. Types log in and submits
3. Selects a test to perform
4. Inputs info to test form
5. Submit

The regular admin workflow to create a test will involve the following:

1. Navigates to URL
2. Types log in and submits
3. Selects link to create a test
4. Fills in general test information
5. Click + to add a test condition
6. Add description and select scale
7. Add user group to the test
8. Submit and create test

3. Error handling and recovery

The main errors to consider will be erroneous inputs to tests. These will mostly be handled by the system. In the event of an error that isn't handled, there will be a bad request URL that we'll redirect the user to. It would be nice to include the nature of the error as well, but it extrapolating that information from the error may be impossible since it will be more related to the server side operations rejecting the input instead of our actual software. If there is an input that doesn't comply and it is actually saved to the DB, there should be a way to reflect that in the exported excel sheet. The error input can be represented in a number away such as "XXXX"

3. Performance Requirements

The app should be able to support 4-5 test users submitting tests concurrently. The size of each recording input is to be determined, but if we implement a noSQL database, it should be a rather small JSON object for each input submitted.

4. Database Requirements

The database will need two main tables. There will need to be a table for the users that will be creating and submitting tests. It will have to support a number of entries equal to the number of students in the classes using the software to submit tests. The other table will be for the tests that are created.

User Table Design

Username	char
Password	char
Group_ID	number

Test table design

Date_Created	Date
User_Group_ID	number
Number_of_entries	number
Entries	Test_Entry

Test entry

Test_Attribute	char
Input_Scale_Type	char
Value	number
Username	char

5. Design Constraints

These main design constraints come from the nature of using a webapp. We will come across problems with transmitting the data from the user and utilizing a shared table. Each table will need to contain the entry data for every user that submitted info for the test. There may be issues with creating a flexible solution for exporting excel sheets as well. There will be a performance issue with a large amount of inputs for test. Depending on how our database is implementing, it may take time to grab each entry and translate it to the excel sheet. We might have to make a flexible user interface as well. While tablets are the main design platform for the testing users, it the input system worked on mobile, it would allow for a lot more freedom for testers, they could potentially not even need tablets to input the data, and could access the site and input their tests on a smart phone.

6. Standards compliance

The main standards compliance will come be networking related. HTTPS should be sought after and authentication involving user log ins as well. If we use CWU as a host, there will be additional guidelines that we will have to adhere to as well. These are currently TBD as we have not fully decided if we are going to use them for hosting.