Test Plan & Results

Overall Test Plan

We will use a variety of testing strategies to measure and iterate the accuracy and efficiency of QSine. For the main tests, we will use several different images. Some of the strategies we will focus on are normal testing, abnormal testing, blackbox testing, and functional testing. Unit testing will be used for the overall development of the software but will not be done on the finished product.

Normal (e.g. case IAI 1) and abnormal (e.g. Case IAI 2) testing are a fundamental part of our process. Because we're using barcode scanning and machine learning, we need to train our software on a variety of images. The only way to ensure this algorithm is functional is by using normal and abnormal tests to make sure it can recognize the desired food items from a variety of angles, but that it won't recognize a food item where none exists if we take an arbitrary image of some unrelated object.

Blackbox (e.g. case BAI2) testing will be used after the algorithm is well trained to see what percentage of the time it correctly identifies an item and returns the relevant list of allergens. The functional (e.g. case IAI1) testing will primarily be for all non-critical features of the App such as the UI or any user settings.

Test Case Descriptions

Image Allergen Identification

IAI1.1	**Image Allergen Identification Test 1**
IAI1.2	This test will validate allergen identification from a clear image of food.
IAI1.3	The QSine App receives a well-lit, clearly defined image of food and only said food item. The App will then process the image and return potential allergens.
IAI1.4	Input: A well-lit, clearly defined image of image of a cheeseburger
IAI1.5	Expected Output: "Contains: Dairy(cheese)"
IAI1.6	Normal
IAI1.7	Blackbox
IAI1.8	Functional
IAI1.9	Integration
IAI1.10	Results: The App correctly returned the allergen Dairy(cheese) from the supplied image
IAI2.1	**Image Allergen Identification Test 2**
IAI2.2	This test will validate that if an image is blurry/unrecognizable the App will not resort to a guess.

IAI2.3 The QSine App receives a poor-quality image of some food. The App will then process the image and return it is uncertain.
IAI2.4 Input: A poor-quality image of some food.
IAI2.5 Expected Output: “Uncertain”
IAI2.6 Abnormal
IA2.7 Blackbox
IAI2.8 Functional
IAI2.9 Integration
IAI2.10 Results: The Application can detect if the input image can be used for querying

Text Allergen Identification Test

TAI1.1 **Text Allergen Identification Test 1**
TAI1.2 This test will verify if the Application can correctly identify allergens of a dish using a text
TAI1.3 The Application will receive an ingredient list and a recipe of a dish. The Application will then return the allergens associated with that dish
TAI1.4 Input: An ingredient list and a recipe of a dish
TAI1.5 Expected Output: All the possible allergens associated with this dish
TAI1.6 Normal
TAI1.7 Blackbox
TAI1.8 Functional
TAI1.9 Integration
TAI1.10 Results: The Application can correctly identify the allergens associated with this dish

TAI2.1 **Text Allergen Identification Test 2**
TAI2.2 This test will verify that if text is unclear, then the QSine App will inform the user that a retake is required.
TAI2.3 We will use the QSine App to take a picture of a menu while moving the camera slightly. This will produce a blurry image that we will then send to the backend API with OCR (Optical Character Recognition) to identify the text. The OCR will be unable to read the characters and will inform the user accordingly.
TAI2.4 Input: A poor-quality image of a menu.
TAI2.5 Expected Output: “Uncertain – please retake image”
TAI2.6 Abnormal
TAI2.7 Blackbox
TAI2.8 Functional
TAI2.9 Integration
TAI2.10 Results: The QSine App is prevented from guessing as uncertainty in the image recognition will result in a request to recRPture any given information from the user.

TAI3.1	**Text Allergen Identification Test 3**
TAI3.2	This test will check to see if the web Application can correctly identify the allergens associated with this dish if the text description contains spelling or grammatical mistakes
TAI3.3	We will use the QSine App to input the text description of a dish. This description will contain spelling and grammatical errors
TAI3.4	Input: A description of a dish with spelling and grammatical errors.
TAI3.5	Expected Output: All the possible allergens associated with this dish
TAI3.6	Abnormal
TAI3.7	Blackbox
TAI3.8	Functional
TAI3.9	Integration
TAI3.10	Results: The QSine App can detect the correct allergens despite spelling and grammatical errors in dish description.

Information Search Test

IS1.1	**Information Search Test 1**
IS1.2	This test will test the accuracy of the search function when using text
IS1.3	The QSine App will receive the description of a dish with ingredients and recipe. The App will retrieve the correct name of recipe and image of recipe
IS1.4	Input: Recipe and Ingredients list of a dish
IS1.5	Outputs: The correct image and name of dish
IS1.6	Normal
IS1.7	Blackbox
IS1.8	Functional
IS1.9	Integration
IS1.10	Results: The correct dish is being retrieved

IS2.1	**Information Search Test 2**
IS2.2	This test will test the accuracy of the search function when using images
IS2.3	The QSine App will receive the image of a dish. The App will retrieve the correct name of the recipe and description of recipe
IS2.4	Input: Image of a dish
IS2.5	Outputs: The correct description and name of dish
IS2.6	Normal
IS2.7	Blackbox
IS2.8	Functional
IS2.9	Integration
IS2.10	Results: The correct dish is being retrieved.

IS3.1	**Information Search Test 3**

IS3.2	This test will test the accuracy of the search function when using text with pronunciation or grammatical errors
IS3.3	The QSine App will receive a description of a dish with ingredients and recipe with grammar and pronunciation errors. The App will retrieve the correct name of recipe and image of recipe
IS3.4	Input: Recipe and Ingredients list with grammar and pronunciation errors
IS3.5	Outputs: The correct image and name of dish
IS3.6	Abnormal
IS3.7	Blackbox
IS3.8	Functional
IS3.9	Integration
IS3.10	Results: The correct dish is being retrieved

Barcode Allergen Identification Tests

BAI1.1	**Barcode Allergen Identification Test 1**
BAI1.2	This test will validate barcode scanning accuracy.
BAI1.3	For this test, we will use the QSine App to scan a UPC (Universal Product Code). Will then ensure that the App is successful in retrieving the UPC number corresponding to the scanned product.
BAI1.4	Input: Barcode for a jar of peanut butter.
BAI1.5	Expected Output: “12345678910”
BAI1.6	Normal
BAI1.7	Whitebox
BAI1.8	Functional
BAI1.9	Unit
BAI1.10	Results: Barcode scanning functionality is accurate and returns the expected number for further Application use.

BAI2.1	**Barcode Allergen Identification Test 2**
BAI2.2	Validate valid barcode item fetching.
BAI2.3	For this test, we will use the QSine App to scan a valid barcode. Then we will ensure that the backend API will return the corresponding product information.
BAI2.4	Input: Barcode for a jar of peanut butter.
BAI2.5	Expected Output: “Contains: Peanuts”
BAI2.6	Normal
BAI2.7	Blackbox
BAI2.8	Functional
BAI2.9	Integration
BAI2.10	Results: Barcode retrieval system is accurate and returns the expected item information back to the QSine App.

BAI3.1 **Barcode Allergen Identification Test 3**

BAI3.2 This test will benchmark the speed of barcode allergen identification.

BAI3.3 For this test, we will scan a UPC with the QSine App and time the system from the scanning of the barcode to the update time of the user's screen. This will simulate normal App use and ensure that the user experience is not hindered by performance.

BAI3.4 Input: Barcode for a jar of peanut butter.

BAI3.5 Expected Outputs: Time < 2 seconds, "Contains: Peanuts"

BAI3.6 Normal

BAI3.7 Blackbox

BAI3.8 Performance

BAI3.9 Integration

BAI3.10 Results: The barcode allergen identification is efficient enough to deliver positive user experience. Short delays prevent the user from getting frustrated especially in cases where successive uses (such as at a grocery store) are needed.

User Experience Tests

UX1.1 **User Experience Test 1**

UX1.2 This test will validate that any user will be able to perform and understand an identification task (image, text, barcode).

UX1.3 For this test, we will give the QSine App to 3 non-developers and ask them to use the App with instructions to check if there are potential allergens in an item and to verbalize their thoughts. We will then observe their steps taken to see if they are able to accomplish this task. This will ensure that the QSine App is user friendly for identification, even when unfamiliar.

UX1.4 Input: The user actions taken on the QSine App to identify allergens in a bag of salami.

UX1.5 Expected Output: User can scan the bag of salami and understand that it may contain processed meats.

UX1.6 Normal

UX1.7 Blackbox

UX1.8 Performance

UX1.9 Unit

UX1.10 Results: The user can utilize the QSine App even when unfamiliar or in a pinch.

UX2.1 **User Experience Test 2**

UX2.2 This test will validate that any user will be able to input their own allergens into the App.

UX2.3	For this test, we will give the QSine App to 3 non-developers and ask them to use the App with instructions to input their own allergens and to verbalize their thoughts. We will then observe their steps taken to see if they are able to accomplish this task. This will ensure that the QSine App is user friendly and can be used to prioritize displaying of the users' own allergens upon identification of an item.
UX2.4	Input: The user actions taken on the QSine App to input their own allergens, in this case dairy and soy.
UX2.5	Expected Output: User can input both their dairy and soy allergens.
UX2.6	Normal
UX2.7	Blackbox
UX2.8	Performance
UX2.9	Unit
UX2.10	Results: The user can personalize the QSine App in a manner that allows them to see a more accurate representation of their risks.

Retrieval Performance Test

RP1.1	**Retrieval Performance Test 1**
RP1.2	This Application will check if the application can retrieve the results from a text query in a reasonable amount of time
RP1.3	For this test, we will input 100 descriptions of dishes into the application. We will calculate the average time it takes to retrieve the name of the dish with the allergens to see if it is under 10 seconds
RP1.4	Input: 100 dishes text descriptions
RP1.5	Expected Output: 100 dish names and their corresponding allergen retrieved with each query taking under 10 seconds
RP1.6	Normal
RP1.7	Blackbox
RP1.8	Performance
RP1.9	Integration
RP1.10	Result: The web application can retrieve the name of a dish and its allergens in less than 10 seconds
RP2.1	**Retrieval Performance Test 2**
RP2.2	This Application will check if the application can retrieve the results from an image query in a reasonable amount of time
RP2.3	For this test, we will use 100 example images and clock their time from submission to the API. The response will then be gathered, and their times will be averaged. This will ensure that the user never has to wait for too long to retrieve an individual result to guide a decision.
RP2.4	Input: 100 example images

RP2.5	Expected Output: average time < 10s and their corresponding allergen identifications
RP2.6	Normal
RP2.7	Blackbox
RP2.8	Performance
RP2.9	Integration
RP2.10	Result: The QSine application can retrieve an image allergen identification with an average time of less than 10 seconds.

Test Case Matrix:

	Normal/ Abnormal	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
IAI1	Normal	Blackbox	Functional	Integration
IAI2	Abnormal	Blackbox	Functional	Integration
TAI1	Normal	Blackbox	Functional	Integration
TAI2	Abnormal	Blackbox	Functional	Integration
TAI3	Abnormal	Blackbox	Functional	Integration
IS1	Normal	Blackbox	Functional	Integration
IS2	Normal	Blackbox	Functional	Integration
IS3	Abnormal	Blackbox	Functional	Integration
BAI1	Normal	Whitebox	Functional	Unit
BAI2	Normal	Blackbox	Functional	Integration
BAI3	Normal	Blackbox	Performance	Integration
UX1	Normal	Blackbox	Performance	Unit
UX2	Normal	Blackbox	Performance	Unit
RP1	Normal	Blackbox	Performance	Integration
RP2	Normal	Blackbox	Performance	Integration