Veggie Check Test Document

CA326 Third Year Project

Project Title:	Veggie Check
Student 1:	Róisín O'Rourke - 19360491
Student 2:	Amelia Grigoriev - 19348241
Supervisor:	Paul Clarke
Date Completed:	04/03/2022

Table of Content

1. Introduction **3**

1.1 Scope 3

1.2 Quality Objective 3

2. Testing Strategies **4**

2.1 Ad hoc testing 4

2.1.1 Ad hoc testing definition 4

2.1.2 Ad hoc testing use 4

2.2 Unit testing 4

2.2.1 Unit testing definition 4

2.2.2 Unit testing use 5

2.3 Performance testing 5

2.3.1 Performance testing definition 5

2.3.2 Performance testing use 5

3. Test Cases **6**

3.1 Test the Accuracy of the API 6

3.2 Test the API against the Database 6

3.3 Test Conversion of Text into a String for the URL 7

3.4 Test the Text Recognition Accuracy 7

3.5 Test the Performance of the Text Recognition 8

3.6 Test the Performance of the API 8

4. References **10**

1. Introduction

Veggie Check is an iOS application that uses the camera on a smartphone to capture the ingredients listed on a given food product and then determines if the product is suitable for vegans to consume. Text recognition is used to process the ingredients, which are then passed to an API to check if they are vegan safe.

This documentation will chronicle the testing that has taken place for the Veggie Check app to ensure that the application ran smoothly and without errors. It will describe the scope, testing strategy, the test cases and the results.

1.1 Scope

The features tested are the key features that are required for the application to function properly and efficiently. These include:

- How efficiently the isVegan API works.
- How efficiently the database works compared to the API.
- How accurate the Vision framework's text recognition is.
- The conversion of the recognised text into a string the API can accept.

1.2 Quality Objective

The objectives of the testing completed is to guarantee that the application operates to an efficient level and to confirm that the different functionalities of the application work properly and at a high performance.

2. Testing Strategies

The three main methods of testing that we focused on while creating the application were ad hoc testing which was used throughout, unit testing on several key functionalities of the system, and performance testing on functionalities that may slow down or decrease the overall performance of the application.

2.1 Ad hoc testing

2.1.1 Ad hoc testing definition

Ad hoc testing is testing completed without formal documentation or planning [1].

2.1.2 Ad hoc testing use

Ad hoc testing was used to test for and locate any bugs or defects while writing the code for the application. It was used throughout the beginning stages of creating the application to ensure that the various functionalities of the app were working as intended to. An example of its use was manually checking that the isVeganSafe status returned by the API was in fact accurate.

2.2 Unit testing

2.2.1 Unit testing definition

Unit testing involves breaking down the code for a system into the smallest possible units of code and then testing them individually and completely independently from the rest of the system [2].

2.2.2 Unit testing use

Unit testing was used to ensure that the individual functions of the system work properly and return the expected results. It was used in this project the verify that:

- The isVegan API will return the expected results.
- The local database will return the same results as the API.
- The recognised text can be converted into a string that can be appended to the URL for the API.
- The Vision framework's text recognition is accurate.

2.3 Performance testing

2.3.1 Performance testing definition

Performance testing examines the speed, reliability, and response time of a system, or a component of the system [3]. It identifies any performance issues so that the system can be improved upon and operate at a more satisfactory level.

2.3.2 Performance testing use

The application must be able to quickly verify the suitability of a food product and display the result to the user, so ensuring that Veggie Check performs to a high standard is crucial. The performance testing completed involved running the same piece of code ten times and then getting an average result for how long it took to run the code. It was used in this project to analyse:

- The performance of the text recognition.
- The performance of the isVegan API.

3. Test Cases

3.1 Test the Accuracy of the API

Test case name:	Test API
Pre-conditions:	There is an internet connection
Steps taken:	● A string of ingredients where the vegan safety of the ingredients is known. ● The string is appended to the URL. ● The API returns if the list of ingredients are vegan safe or not. ● The status of the check is compared with the known status of the ingredient list.
Expected Result:	The API will return the same result on whether the ingredients are suitable with what the actual suitability is.
Actual Result:	The API returned the same result as what was expected.
Test Results:	Pass.

3.2 Test the API against the Database

Test case name:	Test API Against Database
Pre-conditions:	There is an internet connection.
Steps taken:	● An array containing the database's information is created to mock the database. ● A string of ingredients are passed to the database and to the API. ● The isVegan status of the API check and the database check are compared.
Expected Result:	The status of both checks are the same.

Actual Result:	The status of both checks were the same.
Test Results:	Pass.

3.3 Test Conversion of Text into a String for the URL

Test case name:	Test Ingredients to URL String
Pre-conditions:	n/a
Steps taken:	● The list of ingredients is passed to the createString function that makes it suitable to be passed to the API. ● The string is converted to a URL.
Expected Result:	The string will be successfully converted into a URL.
Actual Result:	The string was successfully converted into a URL.
Test Results:	Pass.

3.4 Test the Text Recognition Accuracy

Test case name:	Test Text Recognition Accuracy
Pre-conditions:	n/a
Steps taken:	● An already uploaded image showing the ingredient list of a product is selected for testing. ● The text in the image is manually turned into a string. ● A Vision image request handler is created to extract the text from the image. ● The text extracted is compared with the string previously created.

Expected Result:	The extracted text and the typed out string will assert they are equal to each other.
Actual Result:	The recognised text and the actual text were the same.
Test Results:	Pass.

3.5 Test the Performance of the Text Recognition

Test case name:	Test Performance Recognition
Pre-conditions:	n/a
Steps taken:	● An already uploaded image showing the ingredient list of a product is selected for testing. ● A Vision image request handler is created to extract the text from the image. ● The text recognition is run ten times. ● The average time taken is calculated.
Expected Result:	The text recognition process should take less than a second.
Actual Result:	The process took on average 0.1 seconds.
Test Results:	Pass.

3.6 Test the Performance of the API

Test case name:	Test Performance API
Pre-conditions:	There is an internet connection
Steps taken:	● A list of ingredients is passed to the API. ● The process is run ten times.

	• The average time taken is calculated.
Expected Result:	The API should return a result in under two seconds.
Actual Result:	The average time taken was 0.4 seconds.
Test Results:	Pass.

4. References

[1] Tutorialspoint.com. 2022. *Adhoc Testing*. [online] Available at: <https://www.tutorialspoint.com/software_testing_dictionary/adhoc_testing.htm> [Accessed 3 March 2022].

[2] SearchSoftwareQuality. 2022. *What is Unit Testing? Definition from WhatIs.com*. [online] Available at: <https://searchsoftwarequality.techtarget.com/definition/unit-testing> [Accessed 3 March 2022].

[3] Guru99. 2022. *Performance Testing Tutorial: What is, Types, Metrics & Example*. [online] Available at: <https://www.guru99.com/performance-testing.html> [Accessed 3 March 2022].