*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Ultimate QC 1.0

Team # 17

**Team Members**

Marcine Kelly

**Product Owner**: Rodrigo Morales

**Instructor**: Masoud Sadjadi

Not to be published or publicly distributed without permission by the publisher. Publisher contact information: mkell008@fiu.edu, Product Owner contact information: rmorales@produceinspectors.com.

***Abstract***

*This document introduces PIA’s current process for inspecting produce and it limitations and describes the system being developed in this project to help with that problem. The user stories for the system are listed and the development of this system is documented.*

**Table of Contents**

**Introduction ………………………………………………………………………………………………..5**

Current System …………………………………………………………………………………………….5

Purpose of New System …………………………………………………………………………………6

**User Stories …………………………………………………………………………………………………7**

Implemented User Stories …………………………………………………………………………….7

Pending User Stories ……………………………………………………………………………………8

**Project Plan ………………………………………………………………………………………………..9**

Hardware and Software Resources ……………………………………………………………..9

Sprints Plan ……………………………………………………………………………………………….10

*Sprint 1 ……………………………………………………………………………………………………….11*

*Sprint 2 ………………………………………………………………………………….……………………11*

*Sprint 3 ……………………………………………………………………………………..………………..12*

*Sprint 4 ………………………………………………………………………………………………..……..13*

*Sprint 5 ………………………………………………………………………………………………………14*

*Sprint 6 ……………………………………………………………………………………………….……...15*

*Sprint 7 …………………………………………………………………………………………………..…..15*

**System Design ………………………………………………………………………………….…………17**

Architectural Patterns …………………………………………………………………………….17

System and Subsystem Decomposition …………………………………………………………19

Deployment Diagram ………………………………….………………………………………………21

**System Validation ………………………………………………………………………….…………..22**

**Glossary …………………………………………………………………………………………………….27**

**Appendix …………………………………………………………………………………………………….28**

Appendix A - UML Diagrams …………………………………………………………………………28

*Static UML Diagrams ………………….……………………………………………………………….28*

*Dynamic UML Diagrams ………………………………….…………………………………………..30*

Appendix B - User Interface Design ……………………………………………………………...34

Appendix C - Sprint Review Reports ……………………………………………………………..35

Appendix D - Sprint Retrospective Reports ………………………………………………….38

**References ………………………………………………………………………………………………….40**

# Introduction

This section gives a description of the current system and the new system to be implemented. The Current System section describes how the QC inspectors currently inspect produce and the disadvantage and limitations of that process. The Purpose of the New System section describes how the new system intends to improve upon the current system and gives a description of the functionalities that the new system would provide to the QC inspectors. The remainder of the document.

In the remainder of the document, you will find a comprehensive list of all the user stories for this project. There is also a list of the hardware and software used to develop this system. Sprints Plan lists the user stories worked on for each sprint. The architectural patterns used in the system and a deployment diagram showing the associations between the subsystems and hardware explained. There are test cases for each user story implemented and in the Appendix are diagrams, sprint reviews and retrospectives.

## Current System

The current system for inspecting produce is a visual process. QC inspectors examine crates of produce in a refrigerated environment. They inspect every packet of produce in the crates. They manually count the produce in the packets, look for visible defects, take pictures of the produce and then enter their findings into a report.

Inspecting produce is a slow process and it takes a lot of time to complete a QC report. Spending long periods of time in a refrigerated environment can take its toll on the inspectors. The inspectors can miscount the number of produce that are in the packets, overlook defects, and enter the wrong information into their reports. Also because its takes a great deal of time to complete the QC reports the quality of the produce worsens and most of the produce is already sold before the QC Control is placed [1].

## Purpose of New System

The purpose of the new system, Ultimate QC 1.0, is to alleviate some of the human error that can occur during the inspection process. This new system will help QC inspectors to reduce the time spent on the inspection process so that the can complete their reports in a more timely manner. QC inspectors will be able to use Ultimate QC 1.0 to upload a picture of the produce and then quickly analyze the image. First it will identify the produce in the image. Then it will give an accurate count of the number of produce that is in the image. It will identify defects and give a count of the number of defective produce. After which it will give a percentage of what portion of the produce is defective.

# User Stories

This section contains a comprehensive list of all the user stories gathered from the product owner. Some of these user stories unfortunately were not implemented this semester. The user stories that have been implemented in this version of the system are listed under implemented user stories. The user stories that have been generated for this version, but not yet implemented or completed are those listed under pending user stories. The user stories reflect what functions are required of the new system and how it is to be packaged.

## Implemented User Stories

**User Story # 01 - Count**

Program counts the produce in the image.

Required

**User Story # 02 - Contour**

Program outlines the counted produce in the image.

Required

**User Story # 03 - Recognize**

Program can recognize the produce.

Required

**User Story # 04 - Defects**

Program should detect defects and count them.

Required

**User Story # 05 - Calculate**

Program calculates the percentage of good produce, individual defects, and total percentage of defects.

Required

## Pending User Stories

**User Story # 06 - Camera**

Program is able to use the camera of the device to take pictures of the produce.

Optional

**User Story # 07 - Report**

Program takes data calculated and puts it in a report.

Optional

**User Story # 08 - App**

Program is put together in an app.

Required

# Project Plan

The Hardware and software Resources section gives a list of the hardware and software that will be used to develop this system and why they were chosen. The Sprint Plan section lists the user stories that are to be worked on for each sprint. The user stories are listed in descending order of priority. There are a total of seven sprints.

## Hardware and Software Resources

A list of the hardware and software that I will use to develop this system:

Hardware: PC laptop/computer

Software: Windows OS – because this is the operating system I have running on my PC laptop.

MathWorks: It has documentation, examples, videos and support.

MATLAB – It is a prerequisite for all other toolboxes [2].

Image Processing Toolbox – this toolbox can be used to perform image analysis [3].

Computer Vision Toolbox – this toolbox can perform feature detection, extraction, matching, object detection and tracking [4].

## Sprints Plan

### Sprint 1

(08/31/2015 - 09/11/2015)

**User Story # 06 - Camera**

***Tasks***

* Connect to the camera of the device and capture images with the camera.

***Acceptance Criteria***

* Be able to use the camera of a device and see it on the computer screen.

***Modeling***

Refer to UML diagram in Appendix A figure#5.

### Sprint 2

(09/14/2015 - 09/25/2015)

**User Story # 01 - Count**

***Tasks***

* Program counts the blueberries in the image.

***Acceptance Criteria***

* Program counts each blueberry.
* Program displays the count.

***Modeling***

Refer to UML diagram in Appendix A figure#1.

**User Story # 02 - Contour**

***Tasks***

* Program outlines the blueberries being counted in the image.

***Acceptance Criteria***

* Outline is in a visible color.
* Outlines each blueberry individually.
* Outlines only the blueberries

***Modeling***

Refer to UML diagram in Appendix A figure#4.

### Sprint 3

(09/28/2015 - 10/09/2015)

**User Story # 01 - Count**

***Tasks***

* Program counts the blueberries in the image.
* Fix the count.

***Acceptance Criteria***

* Program counts each blueberry.
* Program displays the count.

***Modeling***

Refer to UML diagram in Appendix A figure#1.

**User Story # 02 - Contour**

***Tasks***

* Program outlines the blueberries being counted in the image.

***Acceptance Criteria***

* Outline is in a visible color.
* Outlines each blueberry individually.
* Outlines only the blueberries

***Modeling***

Refer to UML diagram in Appendix A figure#4.

### Sprint 4

(10/12/2015 - 10/23/2015)

**User Story # 04 - Defects**

***Tasks***

* Create program that can detect defects.

***Acceptance Criteria***

* Each defect has a different contour color.
* Program counts the defects.
* Program matches blueberries against samples of defects in to find defects in the image.

***Modeling***

Refer to UML diagram in Appendix A figure#2.

**User Story # 03 - Recognize**

***Tasks***

* Program can recognize blueberries in an image.

***Acceptance Criteria***

* Doesn’t mistake another fruit for a blueberry.
* Outlines the blueberries.

***Modeling***

Refer to UML diagram in Appendix A figure#3.

### Sprint 5

(10/26/2015 - 11/06/2015)

**User Story # 04 - Defects**

***Tasks***

* Create program that can detect defects.
* Detect immature blueberries

***Acceptance Criteria***

* Each defect has a different contour color.
* Program counts the defects.
* Program matches blueberries against samples of defects in to find defects in the image.

***Modeling***

Refer to UML diagram in Appendix A figure#2.

**User Story # 03 - Recognize**

***Tasks***

* Program can recognize blueberries in an image.

***Acceptance Criteria***

* Doesn’t mistake another fruit for a blueberry.
* Outlines the blueberries.

***Modeling***

Refer to UML diagram in Appendix A figure#3.

### Sprint 6

(11/09/2015 - 11/20/2015)

**User Story # 04 - Defects**

***Tasks***

* Create program that can detect defects.
* Detect immature blueberries

***Acceptance Criteria***

* Each defect has a different contour color.
* Program counts the defects.
* Program matches blueberries against samples of defects in to find defects in the image.

***Modeling***

Refer to UML diagram in Appendix A figure#2.

**User Story # 03 - Recognize**

***Tasks***

* Program can recognize blueberries in an image.

***Acceptance Criteria***

* Doesn’t mistake another fruit for a blueberry.
* Outlines the blueberries.

***Modeling***

Refer to UML diagram in Appendix A figure#3.

### Sprint 7

(11/23/2015 - 12/04/2015)

**User Story # 04 - Defects**

***Tasks***

* Create program that can detect defects.
* Detect immature blueberries

***Acceptance Criteria***

* Contour color is red.
* Program counts the defects.
* Program extracts the immature color to find defects in the image.

***Modeling***

Refer to UML diagram in Appendix A figure#2.

**User Story # 03 - Recognize**

***Tasks***

* Program can recognize blueberries in an image.

***Acceptance Criteria***

* Doesn’t mistake another fruit for a blueberry.
* Outlines the blueberries.

***Modeling***

Refer to UML diagram in Appendix A figure#3.

**User Story # 01 - Count**

***Tasks***

* Reduce outliers in images.

***Acceptance Criteria***

* Program counts each blueberry.
* Program displays the count.

***Modeling***

Refer to UML diagram in Appendix A figure#1.

**User Story # 5 - Calculate**

***Tasks***

* The program gives the percentage of how much of the produce are immature blueberries.

***Acceptance Criteria***

* Returned as a float with no more than one place after the decimal ex: 8.9%

***Modeling***

Refer to UML diagram in Appendix A figure#2.

# System Design

In this section, you will find the architectural patterns used in the system. There is an explanation of the subsystems of the system as well as a deployment diagram showing the associations between the subsystems and hardware.

## Architectural Patterns

Three-tier architecture – The interface layer includes the InspectorInterface subsystem because it deals with everything the user interacts with like making a request to get the count of the berries. The application logic layer includes the DefectManagement and BerryManagement subsystems. This layer processes the images, determining if there are blueberries, counts them and then looks for defects. The storage layer includes the DataStorage subsystem which deals with the storage and retrieval of images.

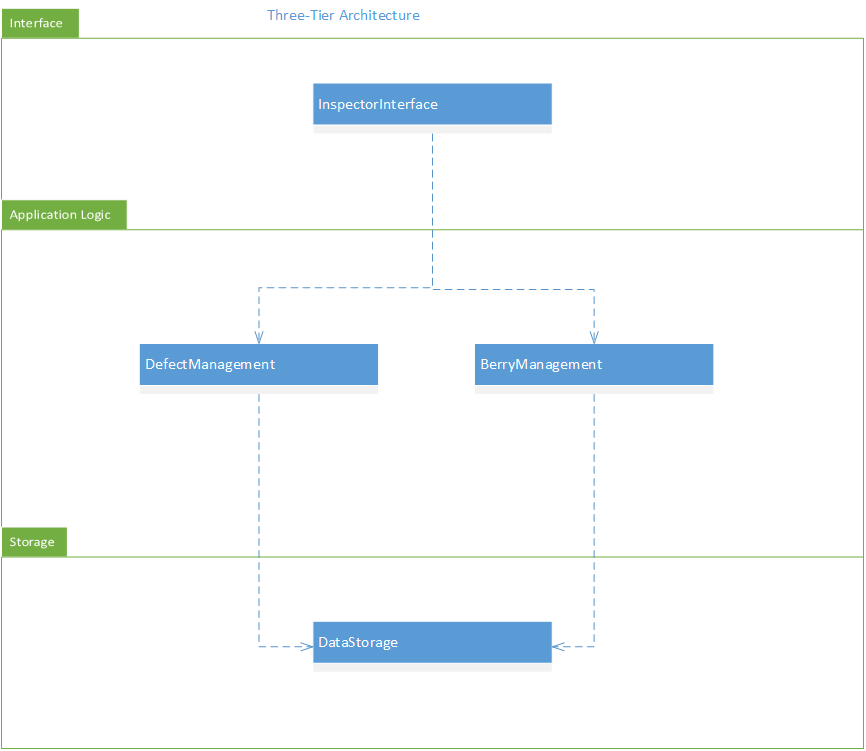


Figure #1 - Three-tier architecture

Pipe and filter architecture – The InspectorInterface subsystem gets an image and sends it to the DataStorage subsystem or gets images from the DataStorage subsystem, and then it outputs that into the BerryManagement subsystem. The BerryManagement subsystem processes the image and sends its output to the InspectorInterface subsystem. The InspectorInterface sends information from this output to both the DefectManagemant and DataStorage subsystems. The DefectManagement subsystem processes the input and sends its output to the InspectorInterface which sends information to the DataStorage subsystem.

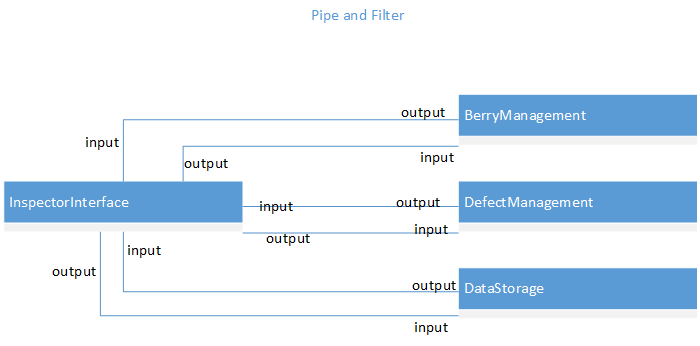


Figure #2 - Pipe and filter architecture

## System and Subsystem Decomposition

The InspectorInterface subsystem is responsible for realizing the user interface for the inspector. The BerryManagement subsystem is responsible for the detection of blueberries and its counting. The DefectManagement subsystem is responsible for the detection of defects and calculating the percentages. The DataStorage subsystem is responsible for the creation and storage of photos.

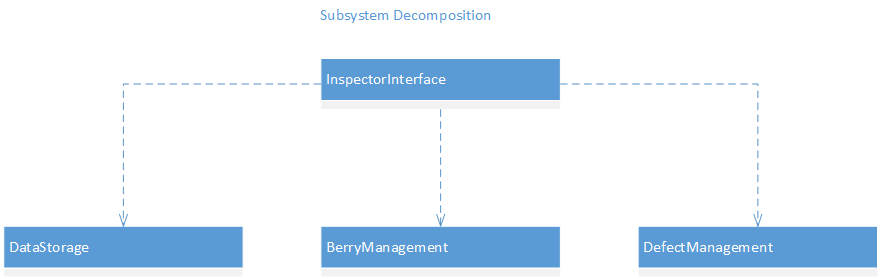


Figure #3 – Subsystem Decomposition

## Deployment Diagram

The InspectorInterface subsystem will reside on any computer that the system is installed on and it is connected to the DataStorage subsystem that resides on a server so that any computer that has the system installed on it will have access to all of the same photos and reports. The BerryManagement subsystem and the DefectManagement subsystem also resides on any computer that the system is installed on and they interact with the InspectorInterface subsystem.

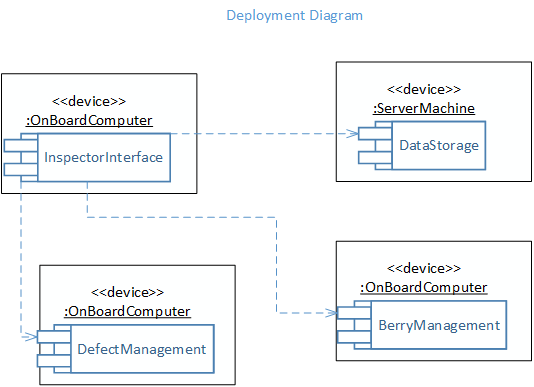


Figure #4 - Deployment Diagram

# System Validation

In this section are the system tests for each user story relating to the use cases. For each test the system preconditions, inputs to the system, expected outcomes of the test, and the results of the tests are described.

**User Story # 1 - Count**

System Tests

* Test 1 - To test that the system counts every blueberry in an image.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.

2. The user clicks on the run button.

* Input The user selects a blueberry image.
* Expected Output The system displays the image with all blueberries counted.
* Results The system will count all of the blueberries in a good quality image.
* Test 2 - To test that the system doesn't count objects that are not blueberries in an image.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.
* 2. The user clicks on the run button.
* Input The user selects a blueberry image.
* Expected Output The system displays the image with only blueberries counted.
* Results The system will count all of the blueberries in a good quality image and some outliers if the background is in the same color range as the blueberries.

**User Story # 2 - Contour**

System Tests

* Test 1 - To test that the system outlines blueberries in an image.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.

2. The user clicks on the run button.

* Input The user selects a blueberry image.
* Expected Output The system displays the image with all blueberries outlined in cyan.
* Results The system will outline the blueberries in a good quality image and some of the outlines are larger than the blueberries.
* Test 2 - To test that the system returns the number of objects outlined equal to the number of blueberries counted in an image.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.
* 2. The user clicks on the run button.
* Input The user selects a blueberry image.
* Expected Output The system displays the image with the same amount of blueberries outlined equal to the count.
* Results The number of objects outlined equal the count.

**User Story # 3 - Recognize**

* Test 1 - To test that the system detects blueberries in an image.
* Precondition 1. MATLAB must be running and detector program must be open.

2. The user clicks on the run button.

* Input The user selects a blueberry image.
* Expected Output The system displays the image with all blueberries detected.
* Results The system displays the image with most of the blueberries detected.
* Test 2 - To test that the system halts execution of the program when it doesn't detect any blueberries in the system.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.
* 2. The user clicks on the run button.
* Input The user selects an image with no blueberries.
* Expected Output The system displays the image saying that there are no blueberries detected and halts execution.
* Results The system halts execution if it doesn’t detect blueberries.
* Test 3 - To test that the system doesn't detect blueberries in images with no blueberries.
* Precondition 1. MATLAB must be running and detector program must be open.
* 2. The user clicks on the run button.
* Input The user selects an image with no blueberries.
* Expected Output The system displays the image with no objects detected.
* Results The system detects blueberries in images that have similar features to blueberries. The system tends to not detect blueberries in blackberry images.

**User Story # 4 - Defects**

System Tests

* Test 1 - To test that the system counts the detected immature blueberry defects, outlines them in red, and gives the percentage in an image with defects.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.

2. The user clicks on the run button.

* Input The user selects a blueberry image with immature blueberries.
* Expected Output The system displays the image with immature blueberries outlined in red and returns the count and percentage.
* Results The system detects immature blueberries in a good quality image.
* Test 2 - To test that the system does not detect immature blueberry defects in an image with no defects.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.
* 2. The user clicks on the run button.
* Input The user selects a blueberry image with no immature blueberries.
* Expected Output The system displays the image with no blueberry outlined in red and the immature defect count and percentage equal to zero.
* Results The defect count and percentage are equal to zero.

**User Story # 5 - Calculate**

System Tests

* Test 1 - To test that the system calculates the percentage of the blueberries that are immature.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.

2. The user clicks on the run button.

* Input The user selects a blueberry image with immature blueberries.
* Expected Output The system displays the image with immature blueberries outlined in red and returns the percentage as a float to the tenth place ex: 0.8%.
* Results The system returns the percentage to the tenth place.
* Test 2 - To test that the system calculates the percentage of the blueberries that are immature as zero in an image with no immature blueberries.
* Precondition 1. MATLAB must be running and Ultimate QC main program must be open.
* 2. The user clicks on the run button.
* Input The user selects a blueberry image with no immature blueberries.
* Expected Output The system displays the image and returns the percentage as a float to the tenth place equal to zero ex: 0.0%.
* Results The percentage is equal to zero.

# Glossary

PIA – Produce Inspectors of America

MATLAB – matrix laboratory

QC – quality control

# Appendix

## Appendix A - UML Diagrams

### Static UML Diagrams

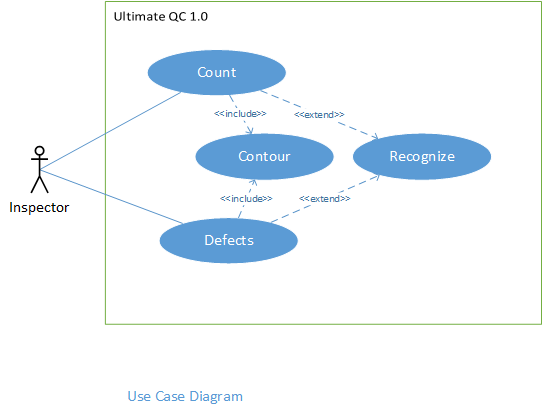


Figure #1 - Use Case Diagram

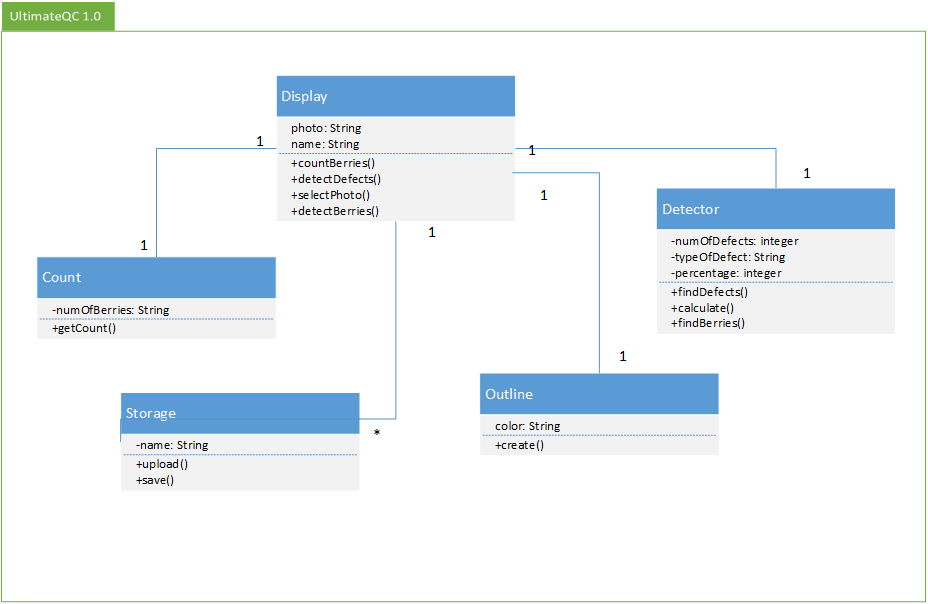


Figure #1 - Class Diagram

### Dynamic UML Diagrams

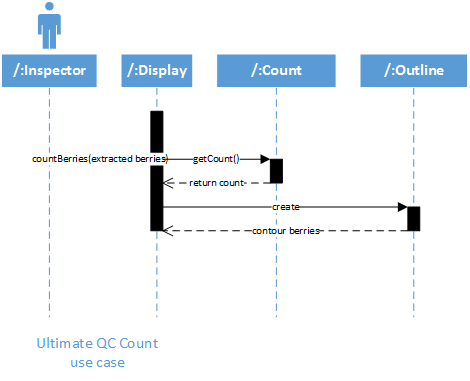


Figure #1 - Count Sequence Diagram

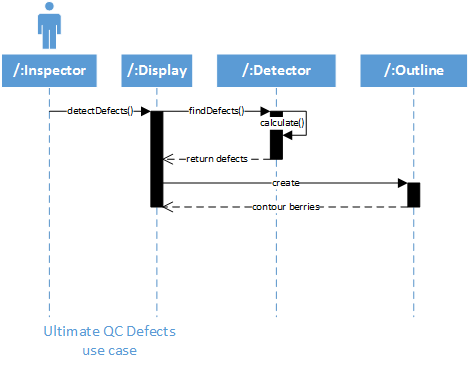


Figure #2 - Defects Sequence Diagram

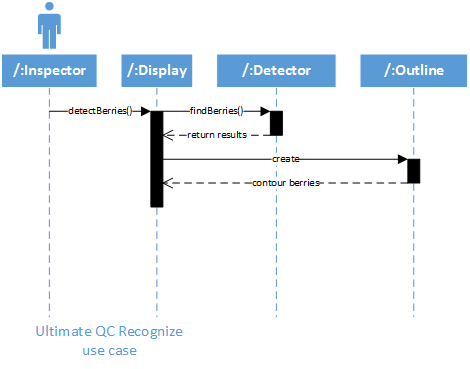


Figure #3 - Recognize Sequence Diagram

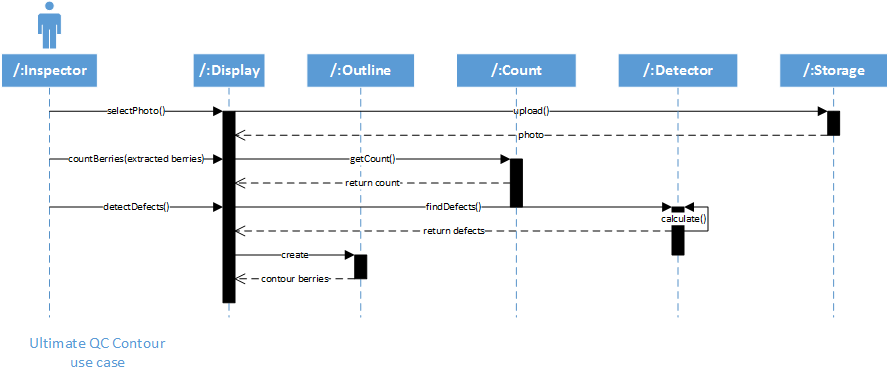


Figure #4 - Contour Sequence Diagram

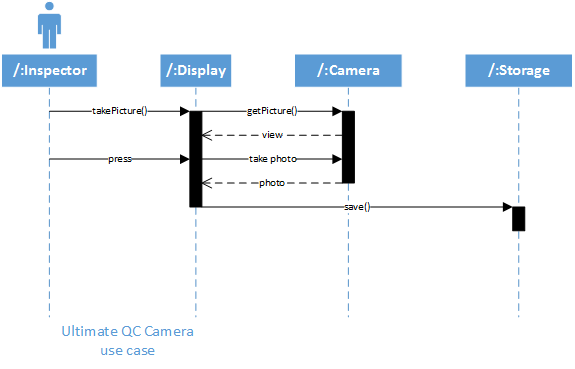


Figure #5 - Camera Sequence Diagram

## Appendix B - User Interface Design

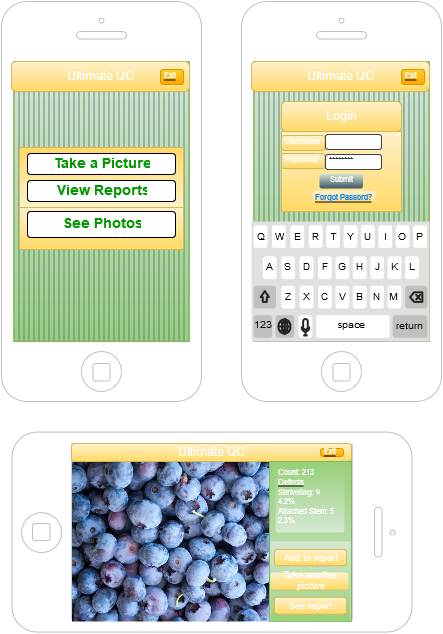


Figure #1 - Mockup of User Interface

…

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** September 11, 2015

**Attendees:** Marcine Kelly

**Discussed Topics:**

Considering the initial goals, cost estimates, and acceptance criteria, briefly explain what was achieved and what was not achieved in this sprint. Specify the reasons for not being able to finish all the work that was initially planned for this sprint. Specify if the product backlog was modified as a result of this meeting and if so, how.

Was not able to get OPENCV to work so, I was not able to work on any user story for this project.

**Sprint 2 Report**

**Date:** September 25, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

I was able to begin implementing the count and contour user stories. These user stories are not complete because it does not count and outline all of the blueberries. It also detects circles that are not there so the settings need adjust because it is too sensitive. Need to define the berries more so that the program can detect the blueberries (find there edges/detect them as circles).

**Sprint 3 Report**

**Date:** October 09, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

I was able to improve the count program. It now counts the blueberries in the images that I have and outlines them. The program does detect outliers (circles) outside of the tray that are not there in some of the images because the background that the tray is on is dark like the blueberries so not all of the background gets extracted.

**Sprint 4 Report**

**Date:** October 23, 2015

**Attendees:** Marcine Kelly

**Discussed Topics:**

I was able to begin implementing the defects and recognize user stories. These user stories are not complete. The program can detect some blueberries but not all blueberries. It also detects places on the tray where there are no blueberries. The defect I am trying to detect is the immature blueberries but it detects any kind of blueberry and things that are not blueberries.

**Sprint 5 Report**

**Date:** November 06, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

Worked on implementing the defects and recognize user stories. These user stories are still not complete. The program seems to be able to recognize more blueberries than before but not all blueberries. It stills detects empty places on the tray. The immature blueberries hasn’t improved much.

**Sprint 6 Report**

**Date:** November 20, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

Continued to work on implementing the defects and recognize user stories. The program recognizes a lot of blueberries but not all. Empty spots are still detected and immature blueberry detection has not improved.

**Sprint 7 Report**

**Date:** December 04, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

I was able to make the program crop the images to the size of the tray to reduce the outliers detected in the count program. When the program is executed it allows the user to select an image. The program halts execution if it does not recognize blueberries in the image. I was also able to detect immature blueberries and return the count and percentage.

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** September 11, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

I was not able to get OPENCV to work and therefore was not able to begin working on any user stories for this project. We agreed that I would do more research and look into more alternatives for the next sprint. My product owners decided to look into MATLAB and contact MathWorks.

**Sprint 2 Retrospective**

**Date:** September 25, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

In this sprint my product owners decided to try a trial version of MATLAB so I was able to begin implementing the count and contour user stories. I was able to detect some blueberries in the images that I have, outline and count them. But, I was not able to count all of them so the program still needs more work.

**Sprint 3 Retrospective**

**Date:** October 09, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

In this sprint I was able to count and outline the blueberries in the images I have by extracting as much of the background as possible. Unfortunately if any part of the background is in the same color range as the blueberries it will show up in the foreground. Therefore there can be outliers present in the count.

**Sprint 4 Retrospective**

**Date:** October 23, 2015

**Attendees:** Marcine Kelly

**Discussed Topics:**

In this sprint I was able to detect some blueberries but some parts of the tray were also being detected. I also tried to detect immature blueberries but the detector detected any kind of blueberry and parts of the tray and background. For the next sprint I am going to include a picture of the tray itself as part of the negatives and try to change the colormap of the images in hopes of detecting immature blueberries better.

**Sprint 5 Retrospective**

**Date:** November 06, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

In this sprint I was able to improve the blueberry recognition detector a bit but it still needs more work. I changed the colormap of the images but the detector still doesn’t detect immature blueberries accurately. For the next sprint my product owners will try to get in touch with MathWorks to see if they can give me some advice/direction on how to improve my recognition and detection programs.

**Sprint 6 Retrospective**

**Date:** November 20, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

Recognition program seems to have improve a bit but the immature blueberry detection program detects all kinds of blueberries and non-blueberries. Was not able to get in touch with MathWorks for advice on how to improve my recognition and detection programs. For the next sprint we hope to hear back from MathWorks and get some direction.

**Sprint 7 Retrospective**

**Date:** December 04, 2015

**Attendees:** Marcine Kelly, Rodrigo Morales, Carlos Villabona

**Discussed Topics:**

In this sprint I was able to detect immature blueberries. I reduced the outliers detected in the counting by cropping the images. The program will halt execution if it doesn’t detect any blueberries in the image. Unfortunately, the recognition program needs a lot more work because it still tends to detect blueberries in images that have none.

# References

[1] PIA’s PowerPoint presentation

[2] http://www.mathworks.com/products/matlab/

[3] http://www.mathworks.com/products/image/

[4] http://www.mathworks.com/products/computer-vision/