Combined-SRS-SDD-Test Document

for

I – COWBOY

Version 1.0

Ebru Ipek AKTUKMAK

Ekrem DEMIRHAN

Ilker Tuna TUZCU

Bugrahan YALCINKAYA

Revision History

Date	Version
27.05.2017	1.0

Table of Contents

A. System Requirements

The purpose of the product “I-Cowboy” is counting human in indoor places with thermal cameras to secure privacy protection and providing useful information to users. The product has a big database, because it stores all thermal camera outputs with the date which outputs are taken at. The information is sent to a web server via Raspberry Pi. At the end, the information is shown on a website to users. Everyone who has a user login and password can use the product. I-Cowboy enables these users to obtain human density for a target area, total human count for a period, and rearrange their businesses with given information. A user can filter the results with target camera ID’s, or time period.

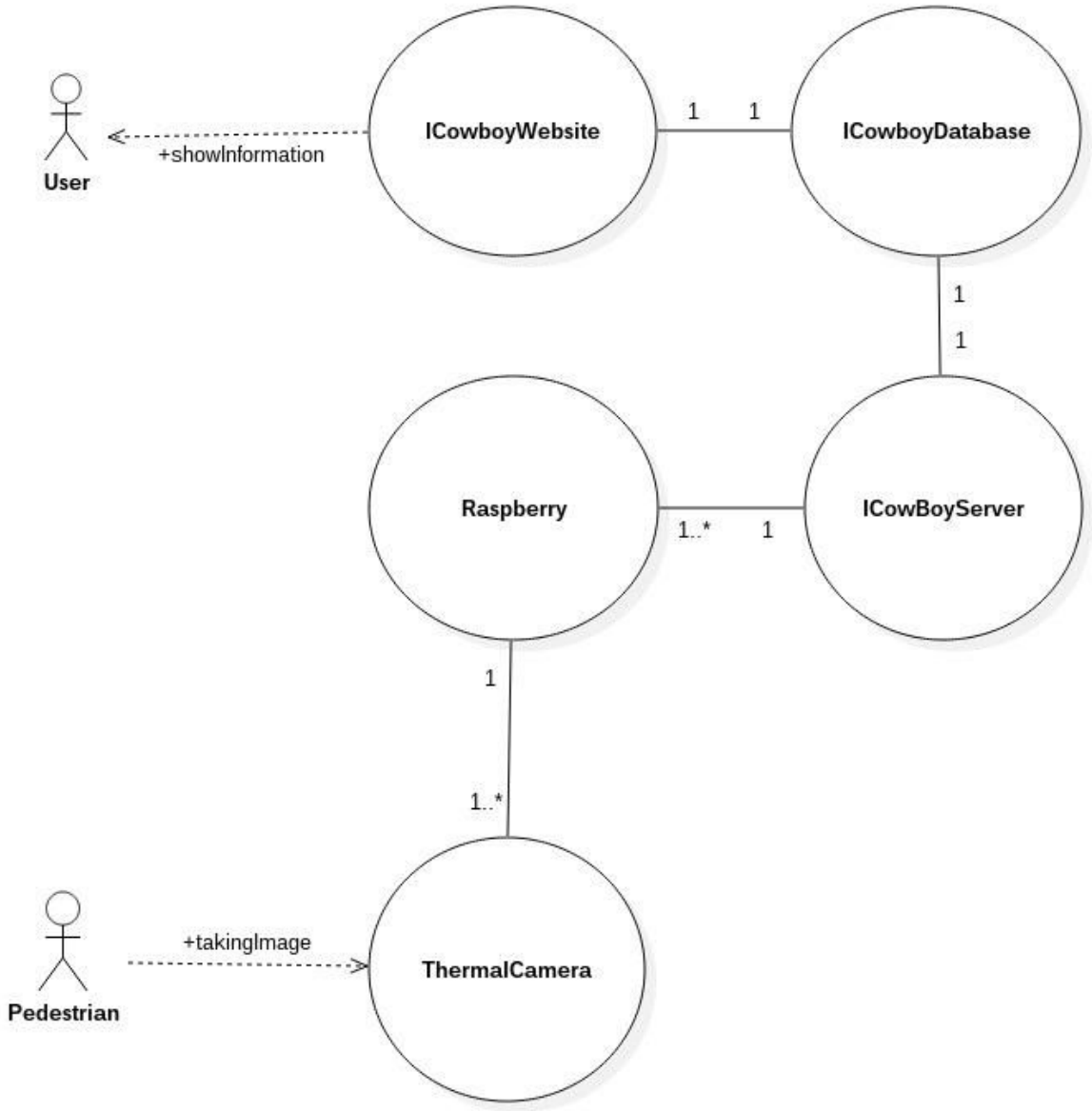

Figure 1: Structure of I-Cowboy

As it can be seen in Figure 1, the product is taking images with a small periods. The images are made meaningful output on Raspberry and the outputs are sent to server. At the server people are counted with machine learning model and all information is stored in the database. When a user would like to see the results, the website is pulling the intended time period or camera ID results.

The project plan of I-Cowboy consists of several steps. At first, getting image from a thermal camera has been done. After that, these images have been enchanced in order to be useful at next steps. With taking thousands of these images, a machine learning model has been created in order to make counting human possible. At the next step, a web server has been built up and machine learning part was placed on that server. The communication between thermal camera and

server is provided with TCP. At that time, image enhancing has been done at Raspberry Pi. After increasing the number of thermal cameras, image processing part has been transferred to processors which are on thermal cameras. The connection with server and website is provided with mongoDB. Server stores the data which it takes from Raspberry Pi (date, camera id, and etc.) and human count. At the end, website shows frames at real time. In addition to that, it can meaningful graphics to the end user.

The product should keep database backups in case of failure because it has a huge and important data in its database. When system crashes, system should recover system information from database backups. Moreover, the reliability of the system mostly depends on internet connection.

The website must be available on a device whenever there is internet connection. It should be available 24 hours a day, 7 days a week.

In order to use website, one needs to have login information. To make the product secure, all login information should be stored on database as encrypted. No one can reach these information.

I-Cowboy uses thermal cameras that are still developed by the manufacturers, because of that, I-Cowboy should be clearly coded and documented in case of big changes.

Because of that end user can reach to the product via website, I-Cowboy can be called portable. It is reachable all by platforms and mobile phones with an Internet connection.

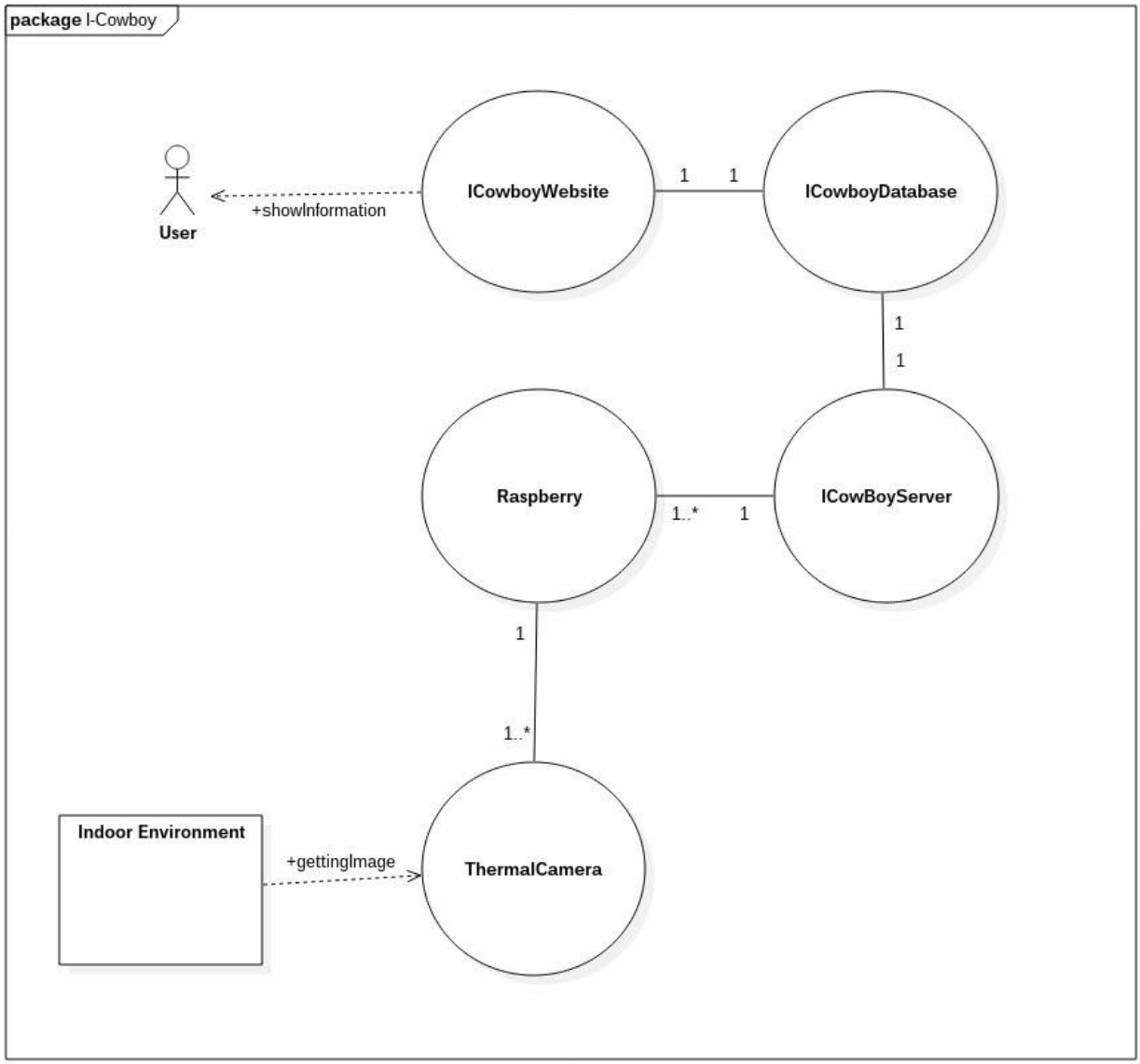

Figure 2: Use Case Diagram

B- System Design

B.1- Module Structure

The system consists of a client, a server and a website front end. Client is the part that collects the raw data from sensors, processes it and sends the simplified image and entrance-exit results to the server. Server is where the counting by machine learning is done and data is stored. Additionally, website backend, which is separated from the server program that communicates with the client part of the System, is run on the server machine.

B.1.1- Client

Client is a program that runs on a Raspberry Pi in the area of interest. It is comprised of 4 modules that form a pipeline. The program is designed to work with a varied number of sensors plugged to the Raspberry Pi. Every time the program is executed, it checks how many cameras are plugged in and runs a separate instance of this pipeline for each camera concurrently. Currently, these operations execute with the speed of approximately 4 frames per second.

In the image acquisition step, the program takes a single raw frame from a sensor via USB, does deinterlacing and fixed pattern noise(FPN) removal. In the enhancement step, it makes further enhancements and simplifications by applying filters to the image, turning it into a simple black-or-white binary image. In the detection step, the first human counting and entrance-exit detection is done. Entrance-exit results are eight numbers, which are numbers of the people entered or exited the camera's field of view from the four edges. Finally, in the last step, binary image is compressed and sent to the server alongside counting results through a TCP connection.

B.1.2- Server

The server accepts TCP connections from multiple client threads receives data. Data consists of a compressed binary frame and entrance-exit counts from four edges of a camera. Since camera data was very often noisy in the earlier stages of the development, counting with image processing was not always relieable. Therefore we developed a machine learning model for this purpose. Server thread decompresses the frame it receives, runs it through the machine learning

model and gets the number of people in it. This result and the entrance-exit results from the clients are saved in different databases with time stamps.

Backend of the website runs on the same machine but is separated from the program that is mentioned in the previous paragraph. The backend handles requests from the website users, such as authentication requests and queries to the database.

B.1.3- Website

Website is where end users can see the results collected from the cameras in the form of visual graphs and predictions based on these results.

B.2- Source Code Structure

B.2.1- Client Side Code Structure

Client side codes reside in the Camera_USB_Reader_Module directory. For transferring data through USB, the program uses libusb, a user-space C library that provides generic access to USB devices.

Below is a list of source files under this directory:

components.h
Contains struct definitions for use in image processing.

fundamental_functions.h
Contains function definitions for image processing.

fundamental_functions.c
Contains definitions of some of the functions declared in the header file with same name. In the current version, this file is unused.

jpeg_reader.h
Contains function declarations for reading & writing .jpeg image files. It was used for testing when we were not able to get real-time data from the camera.

jpeg_reader.c
Contains definitions for the functions declared in the header.

usb.h
Contains USB data transfer-related functions and declarations of some constants that are specific to the protocol used by the particular camera the system uses.

main.c
The main program that starts as many threads as cameras connected, and communicates with the server.

GUI/
This sub-directory contains a program with a graphical user interface that uses the header files from its' parent directory but has its' own main file. This version is used for testing purposes and recording samples to train the machine learning model. We used the Qt library for the interface and OpenGL for the displaying the video footage from the cameras.

Below is a list of source files in this directory:

GLVideo.pro
Qt project file.

mainwindow.ui
Interface file created by the Qt Creator IDE. It is used to generate C++ code while building.

mainwindow.h
Class declarations for the main window and the display widget.

mainwindow.cpp
Function definitions for the main window and the display widget.

main.cpp
The main program. Creates a main window instance for each camera and runs until all windows are closed.

B.2.2- Server Side Code Structure

VPS_MAIN_PROGRAM_WITH_DB+PICTURE+TOTAL_COUNT_V3.py

This Python script accepts TCP connections from client threads and receives data. After receiving a frame, decompresses it and runs it through the machine learning model for counting. Then it saves the counting result from the model and entrance-exit results from client to the database.

website_folders/

This folder contains website files such as HTML documents and server-side scripts written in javascript.

B.3- Diagrams

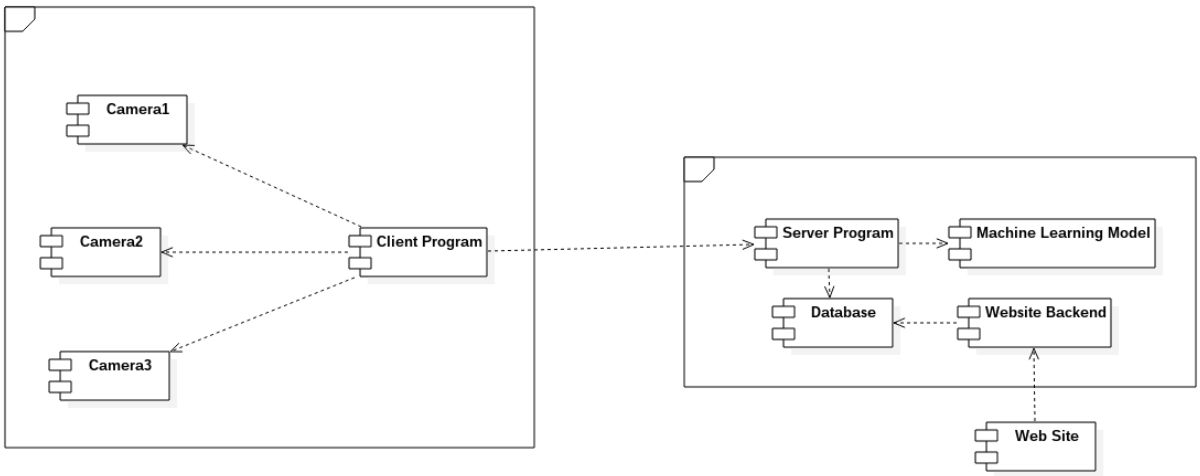

Figure B.1- Component Diagram

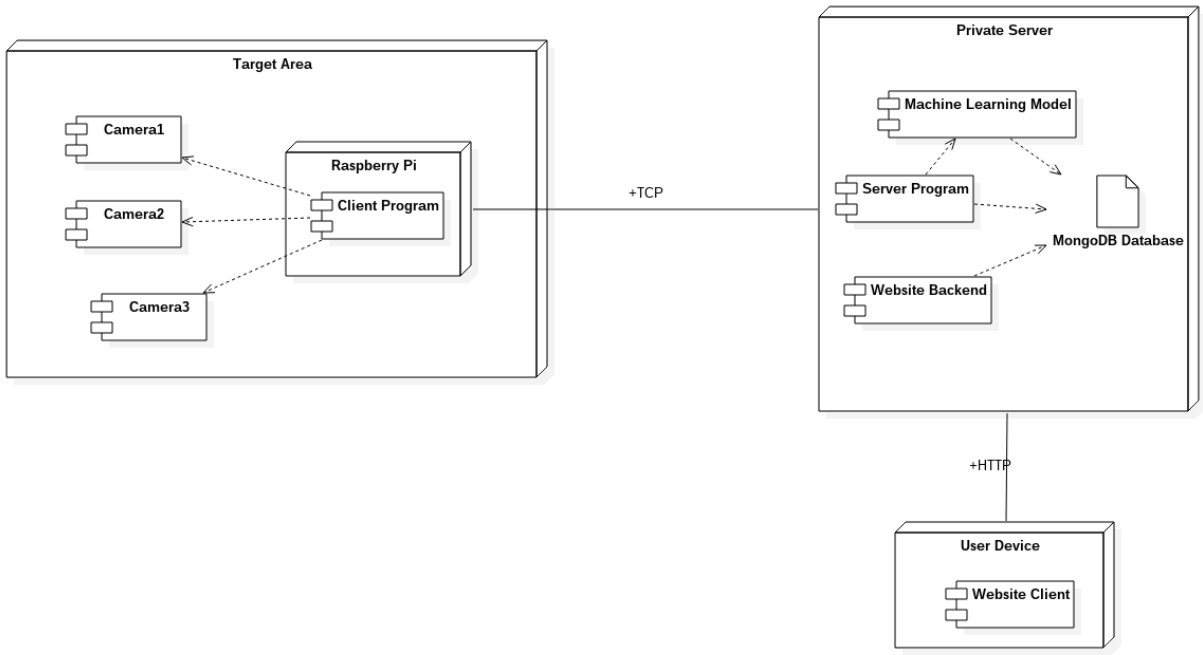

Figure B.2- Deployment Diagram

C. Testing

1-Instant Human Count Testing Plan

These tests are executed with the purpose of determining the accuracy of human count results in a certain time and place given by our system.

1.1-Test Scenerio

Our system provides us human count in a certain time and place momentarily. While doing this functional tests, we compared the instant real human counts and the ones that our system provides.

1.2-Hardware preparation

For executing these tests, we set up our system in three meters high from the floor. The system consists of three thermal cameras and one raspberry pi. The cameras are placed in a way that they all have non-intersecting but close field of views which approximately covers … square meters in total.

1.3-Software preparation

We all configured our software part for working with three cameras and specified height.

1.4-Test Method

We are using machine learning methods to obtain instant human count. There is a method called ‘test classification’ that can be applied after training instant human count with machine learning. This means that we make the system test itself by providing the data that is already been used while executing machine learning.

1.5-Testing Code Structure

While executing these tests, we used a method called ‘test classification’ that we implemented ourselves inside our machine learning model.

1.6-Test Success

With these tests, we obtained eighty-seven percent success. This means that the accuracy percentage of the human count results that our system provides is eighty-five percent correct.

2-Instant Human Count and Human Density Flow Testing Plan

These tests are executed with the purpose of determining the accuracy of instant human count results and human density flow results in a certain time and place given by our system.

2.1-Test Scenerio

In this scenerio, we set up our system in a way that it watches the view of in front of the Ismail Abi Laboratory (one of the computer labs in Computer Engineering Department). We tested the how instant human count and human density flow results that our system provides is correct.

2.2-Hardware preparation

For executing these tests, we set up our system in three meters high from the floor. The system consists of three thermal cameras and one raspberry pi. The cameras are placed in a way that they all have non-intersecting but close field of views which approximately covers … square meters in total. The field of view of the cameras surrounds the entrance and exit of the laboratory.

2.3-Software preparation

We all configured our software part for working with three cameras and specified height.

2.4-Test Method

The cameras are located in a way that the system can observe all the entrance, exit, or directions that most of the people prefers to walk to while entering in or exiting from the laboratory.
While testing this scenerio, as a group, we observed in front of the laboratory from any direction and manually recorded the human count and directions used by each person entering or exiting the interested area.
After getting manual data, we compared the system results with our observation results.

2.5-Testing Code Structure

While executing these tests, we used our source code implementation as it is. This means that we didn't implemented new test codes but we made the source code run in real time and provide results.

2.6-Test Success

With these tests, we obtained eighty-five percent success. This means that the accuracy percentage of the human count and human density flow results that our system provides is eighty-five percent correct.

3-Total Human Count Testing Plan

These tests are executed with the purpose of determining the accuracy of total human count results obtained by merging human count data coming from all cameras in a certain time and place given by our system.

3.1-Test Scenerio

In this scenerio, we set up our system in a way that it watches the view of in front of the Ismail Abi Laboratory (one of the computer labs in Computer Engineering Department). We tested the how total human count results that our system provides is correct. In this tests, different from other tests, data is perceived as a whole by merging data coming from each camera. This means that the scenerio also includes the entegration of data of cameras to each other.

3.2-Hardware preparation

For executing these tests, we set up our system in three meters high from the floor. The system consists of three thermal cameras and one raspberry pi. The cameras are placed in a way that they all have non-intersecting but close field of views which approximately covers … square meters in total. The field of view of the cameras surrounds the entrance and exit of the laboratory.

3.3-Software preparation

We all configured our software part for working with three cameras and specified height.

3.4-Test Method

The cameras are located in a way that the system can observe all the entrance, exit, or directions that most of the people prefers to walk to while entering in or exiting from the laboratory.
While testing this scenerio, as a group, we observed in front of the laboratory from any direction and manually recorded the human count in the interested area.
After getting manual data, we compared the system results with our observation results.

3.5-Testing Code Structure

While executing these tests, we used our source code implementation as it is. This means that we didn't implemented new test codes but we made the source code run in real time and provide results.

3.6-Test Success

With these tests, we obtained eighty-two percent success. This means that the accuracy percentage of the human count and human density flow results that our system provides is eighty-five percent correct. But in these tests, we observed that if data with some unpredictable error comes to the system, the results may not be stable from time to time. We also observed that if the cameras are not located so carefully, when a person is staying at the intersection of the field of view of the cameras , the system may not be stable from time to time.