Bilkent University

Department of Computer Engineering

# Senior Design Project

*Prelude*

## Analysis Report

Group Members: Samet Özcan, Osman Orhan Uysal, Osman Burak İntişah, Mehmet Alper Karadağ, Ziya Erkoç

Supervisor: Uğur Güdükbay

Innovation Expert: Veysi İşler

Jury Members: Hamdi Dibeklioğlu, Shervin Rahimzadeh Arashloo

Analysis Report

March 23, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/1.

# Table of Contents

# 1. Introduction

In the textile industry, the clothes produced in some factories are manually examined in a cloth inspection machine to detect anomalies. When an anomaly occurs, the personnel operating that machine is responsible for either fixing the error right away or marking the improper section so that it can be replaced later. Yet, since people need to scan the cloth flowing continuously, they might miss some anomalies which leads to faulty cloths due to fatigue and the limitations of human nature, therefore, decrease in the companies' product prices and loss in their reputation [1, 2, 3].

# 2. Current System

Some automated cloth inspection machines are developed to speed up and enhance the quality control process. They require small human intervention since anomaly detection is done by the machine itself. These machines use methods such as Fuzzy Logic, Artificial Neural Network (Uster Fabricscan) and sometimes deterministic methods [4]. Yet, these machines cost too much money since the machines are sold as a whole instead of just the automation part as an add-on. Because some companies - including *the* Company A (anonymized) - do not want to purchase new machines but improve theirs, they request an automated fabric anomaly detection product that is compatible with every cloth inspection machine [3]. These machines are also not much accurate probably because the methods they use depend on old technologies [3].

# 3. Proposed System

We are going to develop a software that will detect anomalies in the fabric using a special Deep Learning algorithm called Region-Based Convolutional Neural Network (R-CNN). This algorithm takes defect locations on an image as an input and gives location of the defect on recently seen images. Since we require real-time detection of defects, we will use Faster R-CNN algorithm which is improved version of R-CNN to be used in real-time applications [5]. Our software is also going to ease the image labeling process by providing textile experts with a user-friendly interface. It will also allow continuous improvement of R-CNN by providing a way to do periodical training so that R-CNN can improve itself even during the product is live. Our solution will only require a computer and cameras as much as the number of cloth inspection machines and thus no new inspection machines are needed. As a result, our software will solve the problem more accurately with more recent technology, with continuous improvement mechanism and cheaper.

## 3.1 Overview

We will use Regional Convolutional Neural Network algorithm to predict defects on the clothes. Our software consists of two stages. At the first stage, data will be collected to train our R-CNN model. At the second stage, defects of the cloths will be predicted by our software. During second stage, we will also allow labelling to correct false positives and false negatives so that the model can be improved.

## 3.2 Functional Requirements

- Workers can label data for training through a user interface.
- Workers can fix the false positives (wrong alarms) generated by the model.
- The system should stop the cloth inspection machine and signal warning when it encounters an anomaly.
- Workers can view in which cloth inspection machine the anomaly has occurred and the exact location of the error on the cloth.
- Workers can see which type of anomaly has occurred.
- Managers can generate a report for frequently occurring anomalies per batch and also across all batches processed so far.

## 3.3 Nonfunctional Requirements

### 3.3.1 Performance

Software must be able to detect anomalies within at most 1 second to match the speed of the fabric inspection machine [3].

### 3.3.2 Accuracy

Software must be able to detect anomalies with at least 70% of accuracy which is the rate of accuracy for a human [4].

### 3.3.3 Reliability

Software should not let any fabric to pass uninspected in case of an anomaly such as slow down in the computer or a software error.

### 3.3.4 Compatibility

Software will run on Windows operating system. Video cameras which can record in high definition will be used to capture the flow of the fabric.

### 3.3.5 Scalability

Software should be able to handle the input from at least the number of cloth inspection machines.

### 3.3.6 Maintainability

Operating system, frameworks, python language get updates in a while. Therefore, our software should be up-to-date as well.

## 3.4 Pseudo Requirements

### 3.4.1 Implementation Requirements

- GitHub will be used for version controlling of the project and collaboration among the team members.
- To be able to get the textile images, at least one camera will be placed on top of the machine.
- A Python application will be developed to be used by textile workers to label the images containing anomalies.
- Another Python application will be developed for training the system and detecting the anomalies of the textile.
- At first labeled photos will be stored in local storage, then while training the system the data will be stored in a cloud database.
- Open source libraries and frameworks such as Tensorflow will be used.
- Different libraries and frameworks can be added during the development process.
- Object Oriented Programming (OOP) principles will be followed while designing the applications.

### 3.4.2 Economic Requirements

We will need Line Scan Cameras for prediction and a DSLR camera for data collection both of which will be purchased by the factory.

### 3.4.3 Language Requirements

We will use Turkish for our user interface since the factory that this application will be realized in is in Turkey and all of its employees know Turkish, .

### 3.4.4 Ethical Requirements

The data collected from the factory will not be shared with third-parties.

### 3.4.5 Privacy and Security Requirements

Any information taken from the company to be used in the project -e.g the width and height information of inspection machines and fabrics, daily production, production capacity etc.- or related to the organization, structure, workers etc. of the company must not be shared with any third party for any purposes.

### 3.4.6 Confidentiality Requirements

The statistics and any data related to the faults in the fabrics must be shared only with the company when needed. Otherwise, they can be shared with the third party only if the company gives permission.

### 3.4.7 Maintenance Requirements

CNN will be continuously improved as more data is labeled to increase accuracy.

## 3.5 System Models

### 3.5.1 Scenarios

#### 3.5.1.1 Data Collection

The detection of faults for data collection will be under responsibility of a worker who already scans the fabrics for faults. He is assumed to stop the fabric flow as soon as he detects a fault(s), and later, take a photo of the fabric, label the area including the fault on the photo and select the class of the fault via the application on the computer. This process will continue until the sufficient number of data is collected.

#### 3.5.1.2 Prediction: Absence of Fault

When there is no fault in the fabric, the system just continues to scan the fabric, that is, it collects the image data, processes it, and checks whether there is a fault in the image or not. In this case, there is no need to pause the fabric flow or warn the worker for the fixation of the fault.

#### 3.5.1.3 Prediction: Presence of Fault

When there is a fault in the part of the fabric being scanned, the system is assumed to detect the fault after capturing and processing the image of the fabric. Later, the system should pause the fabric flow by sending a signal to the cloth inspection machine, and warn the worker via user interface so that he can fix the error. After the worker fixes the fault, he is assumed to press the button near the cloth inspection machine in order for the scanning process to continue. When he presses the button, the system will record the fixation information for statistical purposes as well.

#### 3.5.1.3 Training during Prediction

Occurrences of some fault types take a long time, therefore, collecting sufficient numbers of those type faults might result in interruption of the setup of the system. In addition, the number of data collected increases the accuracy of the system in Neural Networks in general. Thus, between the two stages -training and prediction- we are planning to conduct another stage, when training is continued while predictions are made. For instance, when the system detects a fault(s) but the worker decides that actually there is no fault or vice versa, the system will be updated accordingly.

## 3.5.2 Use Case Model and Descriptions

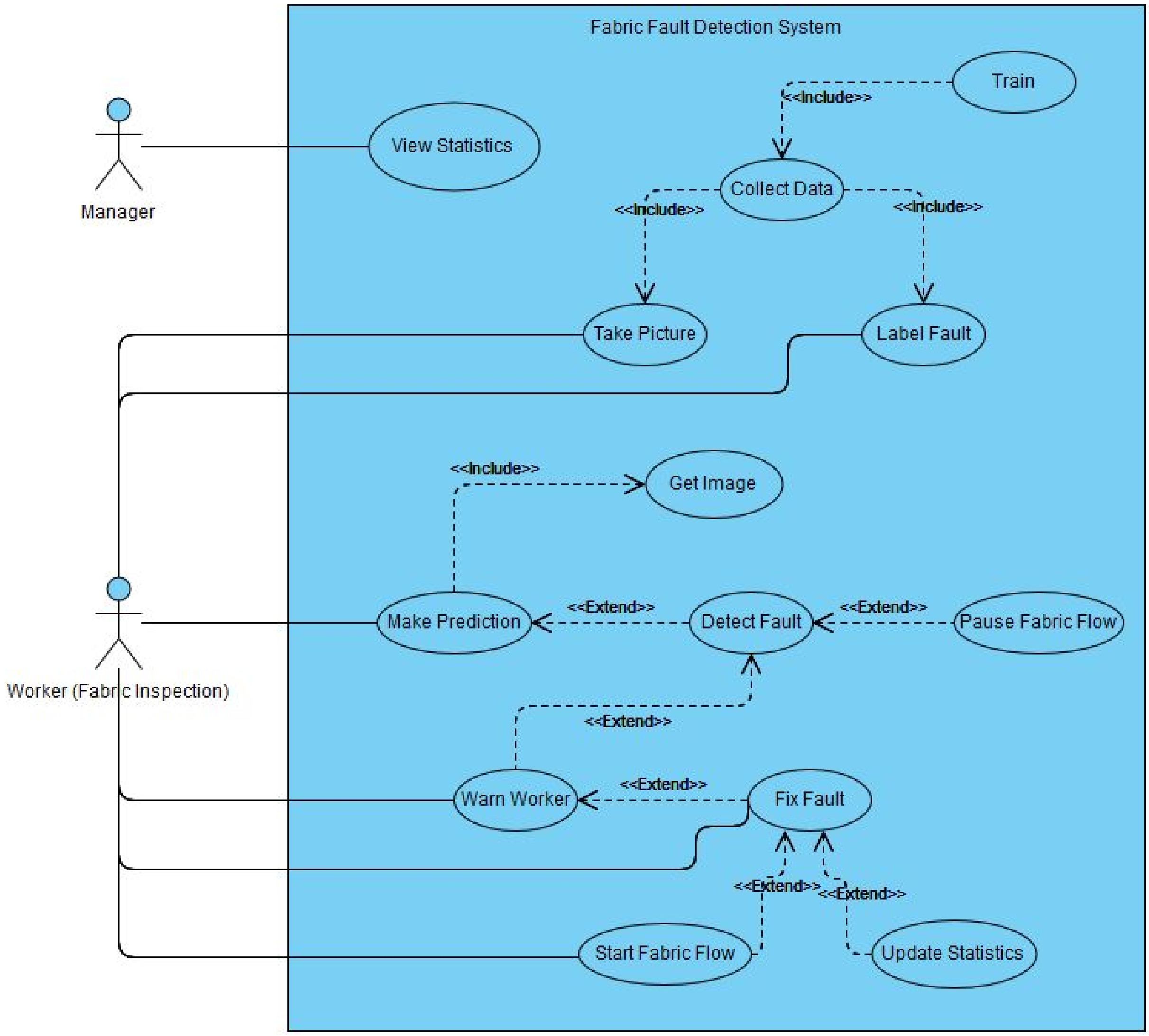


Figure 1: Use Case Diagram

| | |
|---|---|
| **Use Case Name** | Train |
| **Participating Actor** | |
| **Entry Condition** | The system is set up and started for data collection |
| **Flow of Events** | 1. Data is collected<br>2. Data is preprocessed<br>3. CNN works on the collected data |
| **Exit Condition** | Training processes of CNN on the data is finished |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Collect Data |

| | |
|---|---|
| **Participating Actor** | Worker (Fabric Inspection) |
| **Entry Condition** | The system is set up and started for data collection |
| **Flow of Events** | 4. The worker scans the fabric flow for detecting faults<br>5. The worker detects a fault<br>6. The worker takes picture of the fabric, labels the fault and its class, and saves the image -in all- using the application on the computer |
| **Exit Condition** | Sufficient number of fault image is collected |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Take Picture |
| **Participating Actor** | Worker (Fabric Inspection) |
| **Entry Condition** | A fault is detected by the worker |
| **Flow of Events** | 1. The worker clicks on the “Take Picture” button on the interface of the application on the computer |
| **Exit Condition** | The picture of the area including fault(s) is taken |
| **Special Requirement** | The worker must pause the fabric flow after fault detection in order to get the picture of the faulty area |

| | |
|---|---|
| **Use Case Name** | Label Fault |
| **Participating Actor** | Worker (Fabric Inspection) |
| **Entry Condition** | A picture of the area including the fault(s) is taken |
| **Flow of Events** | 1. The worker clicks on the faulty area<br>2. The worker selects the fault class |
| **Exit Condition** | The worker saves the labeled image |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Take Picture |
| **Participating Actor** | Worker (Fabric Inspection) |
| **Entry Condition** | A fault is detected by the worker |
| **Flow of Events** | The worker clicks on the “Take Picture” button on the interface of the application on the computer |

| | |
|---|---|
| **Exit Condition** | The picture of the area including fault(s) is taken |
| **Special Requirement** | The worker must pause the fabric flow after fault detection in order to get the picture of the faulty area |

| | |
|---|---|
| **Use Case Name** | Make Prediction |
| **Participating Actor** | |
| **Entry Condition** | Images are taken from the fabric continuously |
| **Flow of Events** | The system scans the fabric flow for detecting faults via the image taken |
| **Exit Condition** | The system detects a fault on the fabric |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Get Image |
| **Participating Actor** | |
| **Entry Condition** | The fabric flow is started |
| **Flow of Events** | 1. An image of the fabric is taken via the camera by the system |
| **Exit Condition** | The image is taken |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Detect Fault |
| **Participating Actor** | |
| **Entry Condition** | The system detects a fault(s) on the taken image |
| **Flow of Events** | 1. The system scans the taken image<br>2. The system detects a fault(s) and its location on the image<br>3. The system determines the class of the fault<br>4. The system labels the image accordingly |
| **Exit Condition** | The system detects a fault on the fabric |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Warn Worker |
| **Participating Actor** | Worker (Fabric Inspection) |

| | |
|---|---|
| **Entry Condition** | The system detects a fault on the image and determines the fault's class |
| **Flow of Events** | 1. The system labels the inspection machine where a fault(s) is detected on the screen<br>2. The system shows the labeled image of the corresponding fabric on the screen |
| **Exit Condition** | Warning processes are completed |
| **Special Requirement** | |

| | |
|---|---|
| **Use Case Name** | Pause Fabric Flow |
| **Participating Actor** | |
| **Entry Condition** | The system detects a fault(s) on the taken image |
| **Flow of Events** | 1. The fabric flow on the inspection machine where a fabric fault(s) is detected is paused |
| **Exit Condition** | The workers presses on the 'Start Fabric Flow' button |
| **Special Requirement** | |

| | |
|---|---|
| **Usa Case Name** | Fix Fault |
| **Participating Actor** | Worker |
| **Entry Condition** | The worker is warned about the fault, faulty fabric and the corresponding inspection machine |
| **Flow of Events** | 1. The worker goes to the inspection machine<br>2. The worker fixes the fault according to its type |
| **Exit Condition** | The fault is fixed |
| **Special Requirement** | |

| | |
|---|---|
| **Usa Case Name** | Start Fabric Flow |
| **Participating Actor** | Worker |
| **Entry Condition** | The worker fixes the fault |
| **Flow of Events** | 1. The worker presses on the 'Start Fabric Flow' botton near the inspection machine |
| **Exit Condition** | The fabric flow is started |

| | |
|---|---|
| **Special Requirement** | |

| | |
|---|---|
| **Usa Case Name** | Update Statistics |
| **Participating Actor** | |
| **Entry Condition** | The fabric flow is started on the inspection machine where a fault(s) is fixed by the worker |
| **Flow of Events** | 1. The system updates statistics which include fault information, faulty fabric information, date and time etc. |
| **Exit Condition** | Statistics are updated accordingly |
| **Special Requirement** | |

| | |
|---|---|
| **Usa Case Name** | View Statistics |
| **Participating Actor** | Manager |
| **Entry Condition** | The manager opens the statistics window of the application on the computer |
| **Flow of Events** | 1. Statistics related to the frequency of fault occurrences for each fault type, the fabrics and the weaving machines they belong to, the share of each faulty type in the fault occurrences etc. are displayed on the screen |
| **Exit Condition** | The manager closes the statistics window |
| **Special Requirement** | |

## 3.5.3 Object and Class Model

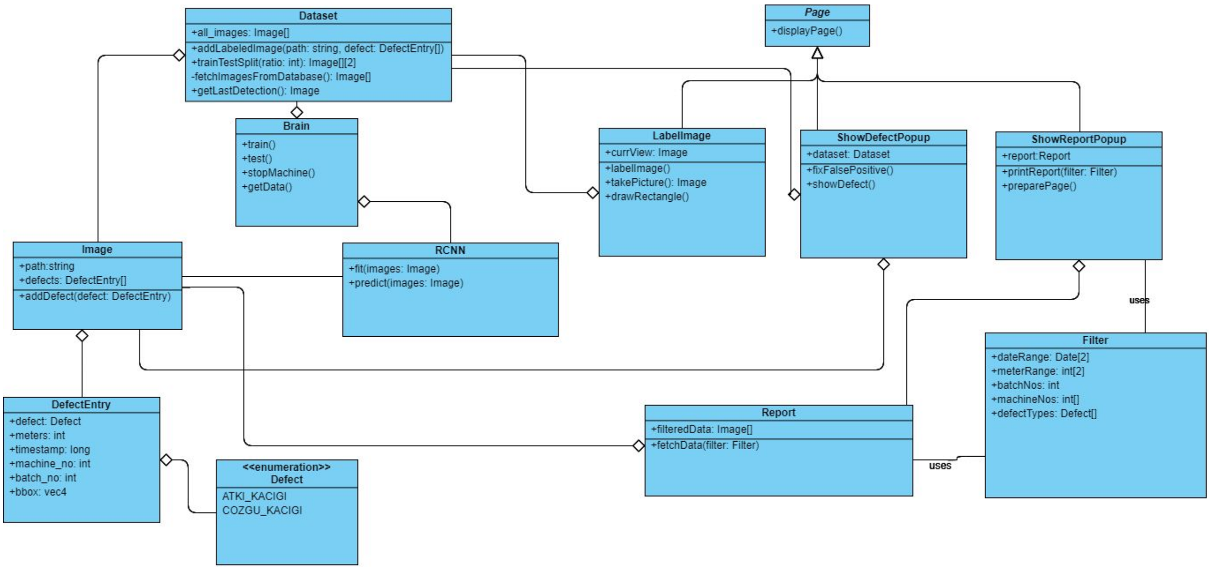


Figure 2: Class Diagram

### 3.5.3.1 Dataset Class

**Description:** This class contains all images in our database to be used for training and testing.

**Attributes:**

- *all_images*: Holds all the images captured so far

**Methods:**

- *addLabeledImage*: It adds new image data to the dataset
- *trainTestSplit*: It randomly splits all images into train and test sets based on the ratio given as parameter. It returns two disjoint train and test sets.
- *fetchImagesFromDatabase*: It synchronizes the local images with the database. If new data inserted into the database it pulls or if new data inserted locally it sends to the database. Here database can simply be a .txt file.
- *getLastDetection*: This method finds the last image that contains a defect to be used by other classes.

### 3.5.3.2 RCNN Class

**Description:** This class is responsible for training and testing the R-CNN model.

**Attributes:** -

**Methods:**

- *fit*: This method trains the R-CNN model. Its inputs are image itself, defect label and the bounding box of the defect.

- *predict*: This method finds the label and bounding box of the defect, if any, given an image.

#### 3.5.3.3 Image Class

**Description:** This class is responsible for keeping image information.

**Attributes:**

- *path:* Absolute location of the image in the computer
- defects: Defects associated with this image.

**Methods:**

- *addDefect:* Adds new DefectEntry instance to a particular image

#### 3.5.3.4 DefectEntry Class

**Description:** This class is responsible for keeping information about a defect.

**Attributes:**

- *defect:* Type of the defect
- *meters*: At which meter of the cloth the defect resides
- *timestamp*: The exact epoch time the defect seen
- *machine_no*: Identity number of the machine where the defect occurred
- *batch_no*: Identity number of the cloth batch where the defect occurred
- *bbox*: Bounding box of the defect

**Methods:** -

#### 3.5.3.5 Defect Enumeration

**Description:** It enumerates all types of defects. As new types are collected, this enumeration will enlarge.

**Values:** ATKI_KACIGI, COZGU_KACIGI

#### 3.5.3.6 Report Class

**Description:** It is responsible for generating comprehensive reports based on the filter.

**Attributes:**

- *filteredData*: It holds last fetched and filtered data to be used again and again.

**Methods:**

- *fetchData:* It fetches all data from the Brain class that satisfies the filter conditions.

### 3.5.3.7 Brain Class

**Description:** This class manages the whole inspection process

**Attributes:** -

**Methods:**

- *getData*: Gets train and test data from dataset
- *train*: It initiates training process
- *test*: It initiates prediction process
- *stopMachine*: It stops the machine in case defect occurred

### 3.5.3.8 Page Class

**Description:** It is an abstract class to show user interface

**Attributes:** -

**Methods:**

- *displayPage*: Abstract method that will be implemented by the children of the class. This method is responsible for populating the user interface

### 3.5.3.8 LabelImage Class

**Description:** This class is used for showing user-interface for labeling the images to train the model.

**Attributes:**

- *currView*: It is the live stream of the cloth inspection machine shown on the screen

**Methods:**

- *labelImage*: This method gets the bounding box and class of the image and stores it.
- *takePicture*: It takes picture of the defect and shows on the screen
- *drawRectangle*: It draws rectangle on the image to specify bounding box of the defect

### 3.5.3.9 ShowDefectPopup Class

**Description:** This class is responsible for visually showing the defects along with their locations on the image

**Attributes:**

- *dataset*: It holds a dataset instance to show last

**Methods:**

- *fixFalsePositive*: If the defect shown is not a defect but a false positive generated by the model, this method fixes it.

- *showDefect*: It shows last detected image to the user.

#### 3.5.3.10 ShowReportPopup Class

**Description:** This class is responsible for populating necessary user interfaces to filter, query and print reports.

**Attributes:**

- *report*: Current report for the detected defects

**Methods:**

- *printReport*: It prints the report with an applied filter.
- *preparePage*: It prepares user interface to show report and filtering options.

#### 3.5.3.11 Filter Class

**Description:** This class is responsible for holding filters/queries to generate reports for particular cases.

**Attributes:**

- *dataRange*: Defects occurred at between these dates
- *meterRange*: Defects occurred at between these meters
- *batchNoRange*: Defects occurred at these batches
- *machineNoRange*: Defects occurred at these machine
- *defectTypes*: Only these defect types

**Methods:** -

## 3.5.4 Dynamic Models

### 3.5.4.1 Activity Diagram

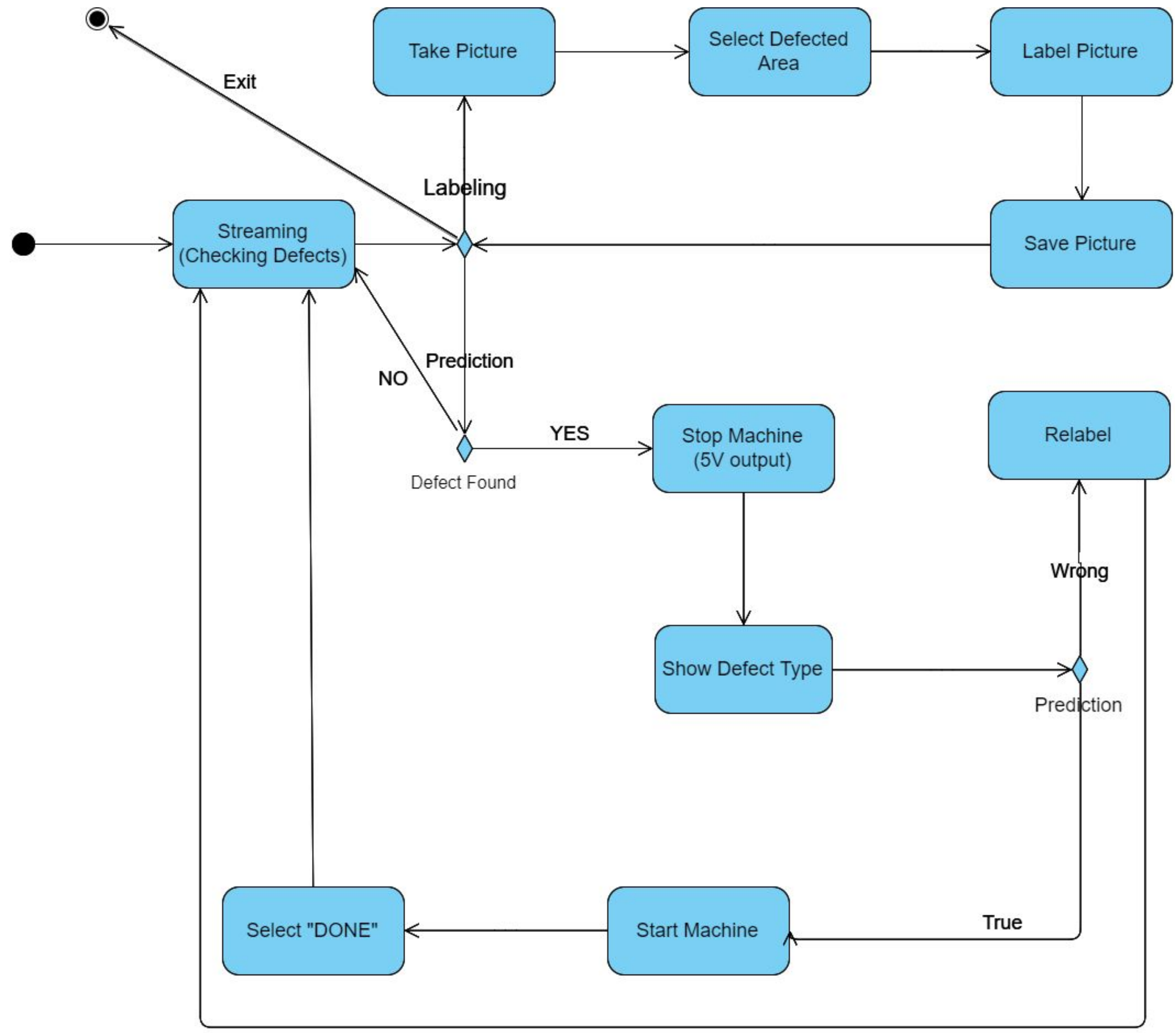


Figure 3: Activity Diagram

The activity diagram shows all the steps that could be taken by the system. The application starts with the streaming of the fabrics that is continuously moving on the machine. From this state the user can go to labeling activity, prediction activity and exit. From labeling activity users will be able to take the current picture of the stream, select the defect area and type and save the picture. After saving the image it goes back to streaming activity. If the system is continuously running and not labeling any photo it is trying to predict whether or not there is a defect in the photo taken from the stream. If it finds a defect, it gives an output as 5V to stop the machine, shows the defect type as popup, and asks the confirmation from the user. If the prediction is wrong, the user can relabel the taken image. Otherwise, if the prediction is true, after starting the machine and selecting the done button, it goes back to streaming activity.

### 3.5.4.2 State Diagram

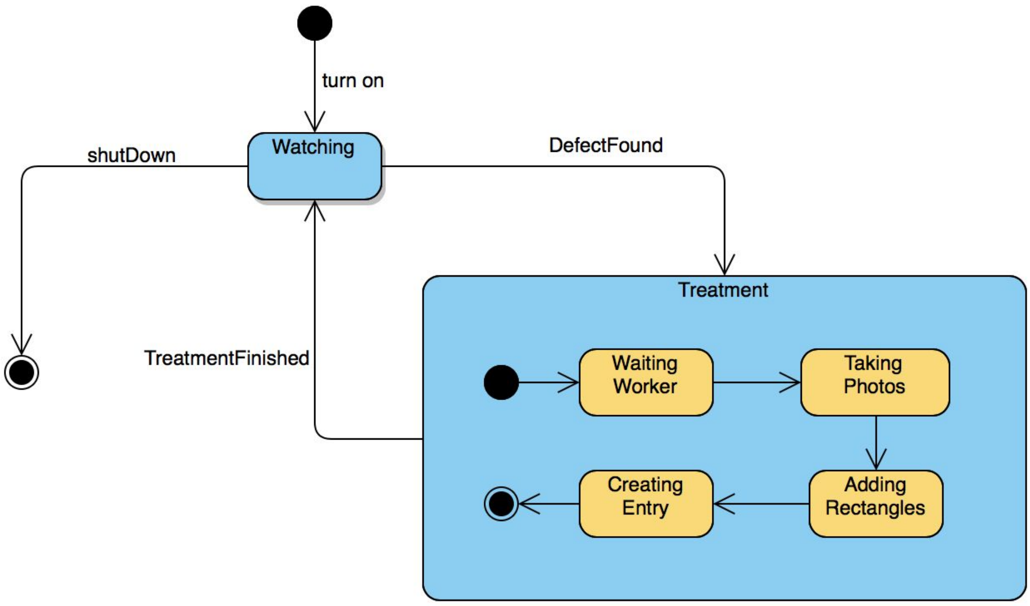


Figure 4: Predictor State Diagram

Program starts with a state where the program is watching for any defects. If any defect is found, the program goes to the treatment state. First it will wait for the quality control worker to take action. Then it will take the photos of the fabric where the defect is occured. Rectangles will be added and entry will be created with the worker’s commands. Then treatment will be finished and the program continue to watch for other defect until it is shut down.

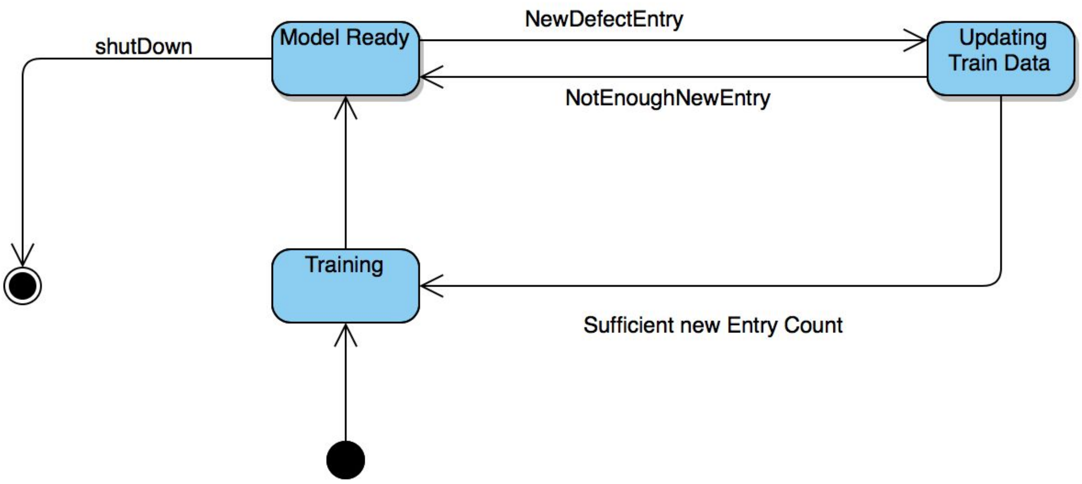


Figure 5: Trainer State Diagram

In the training side of the program, it will start with training the initial dataset. When the model is ready, program watch and predict if fabric contains any defect. Whenever the predictor program sends an entry, it will update the data pool and look if enough new entries have come to re-train the model.

### 3.5.4.3 Sequence Diagram

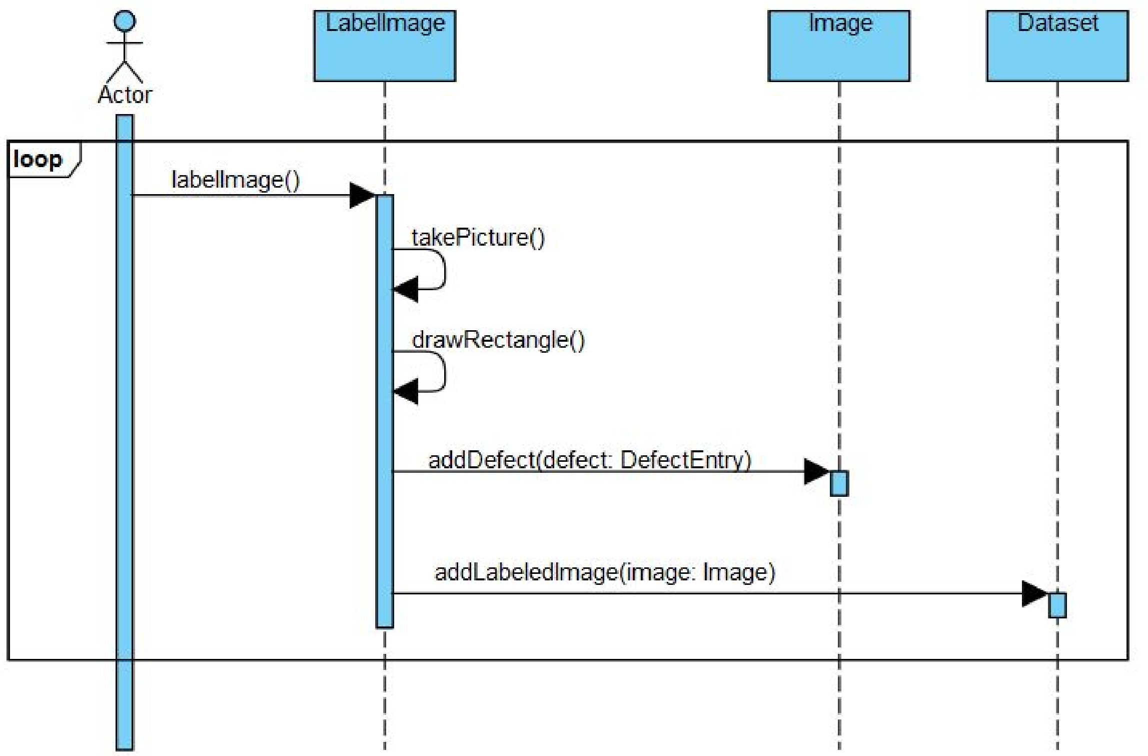


Figure 6: Label Image sequence diagram

If a worker detects a flaw in the system, that is, a defect is not detected or the machine stopped but there is no defect, he interrupts the fabric inspection machine and triggers the image labeling process. First, a picture of the fabric is taken then, worker marks the area with defects or marks the picture as defectless. If there is a defect, defect entry is added to the image. Finally the training dataset is updated for further improvements.

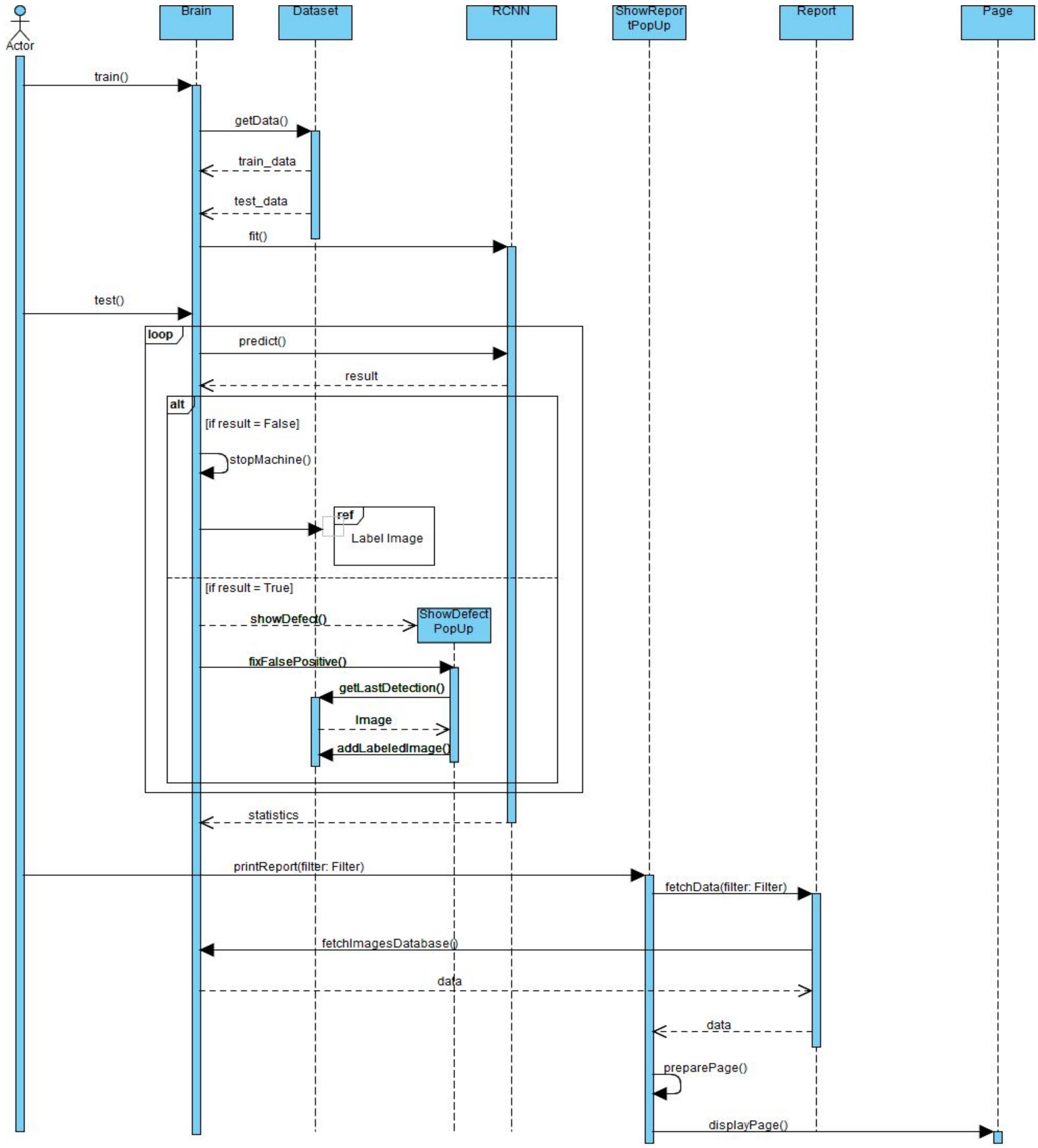


Figure 7: Main flow sequence diagram

First of all, we train our model with the dataset we have. Brain class controls the flow between Neural Network and Dataset. At the prediction phase captured pictures are constantly fed into the model to predict the labels. R-CNN returns the result of the prediction. If a worker detects a flaw in this prediction he stops the machine and Label Image sequence is invoked. If our system detects a defect in a flawless cloth then the worker will tell our software that it is a false positive. Users can also see the statistics and performance of the model which are kept in the Brain class. When the user clicks the show statistics button Show report class gets relevant information and prepares the page with filters given in the UI. Finally, Page class which is parent to UI classes displays the page.

## 3.5.5 User Interface - Navigational Paths and Screen Mock-ups

### 3.5.5.1 Streaming Page

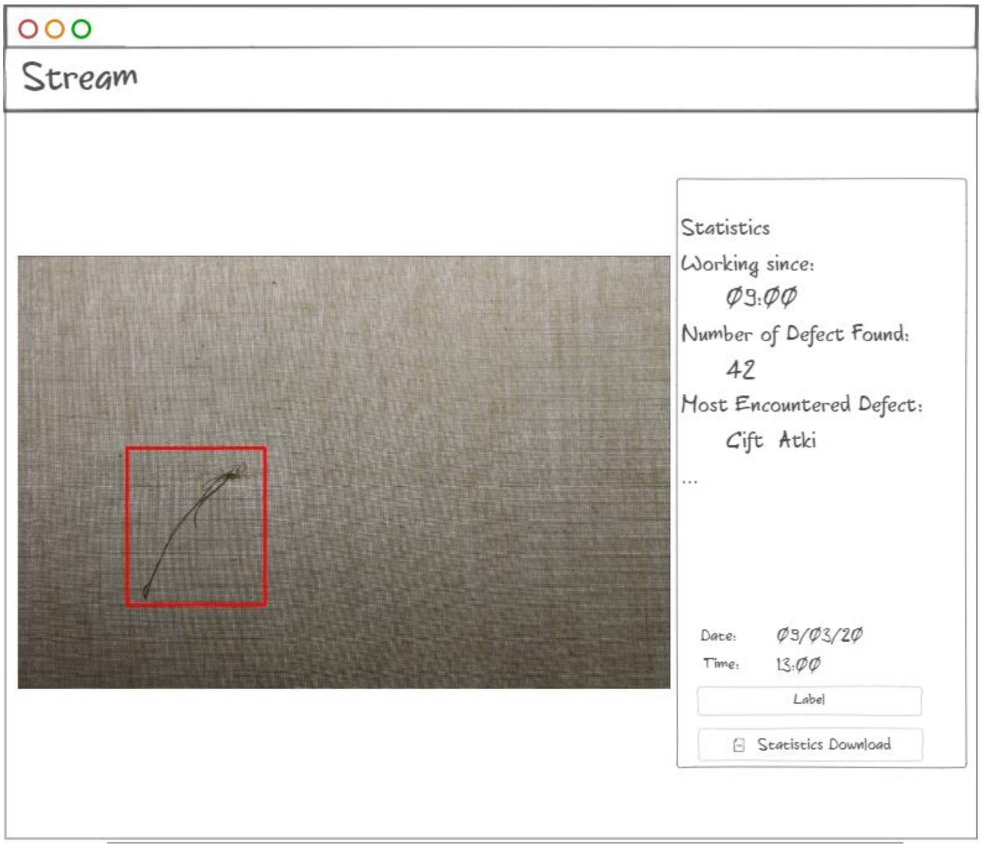


Figure 8: Image Labeling UI

This tool is going to be used for both data collection and after the machine learning stage is completed. The stream will be shown along with the statistics of the machine learning program. In the small statistics part working time of the program, number of defects found, most encountered defects etc. will be shown. Managers will still be able to download more comprehensive reports which will be mentioned in the *Statistics* section. Additionally, workers will be able to label the fabrics with the label button while searching for defects. When the label button is selected a photo is taken and popup is shown.

### 3.5.5.2 Labeing Pop-up

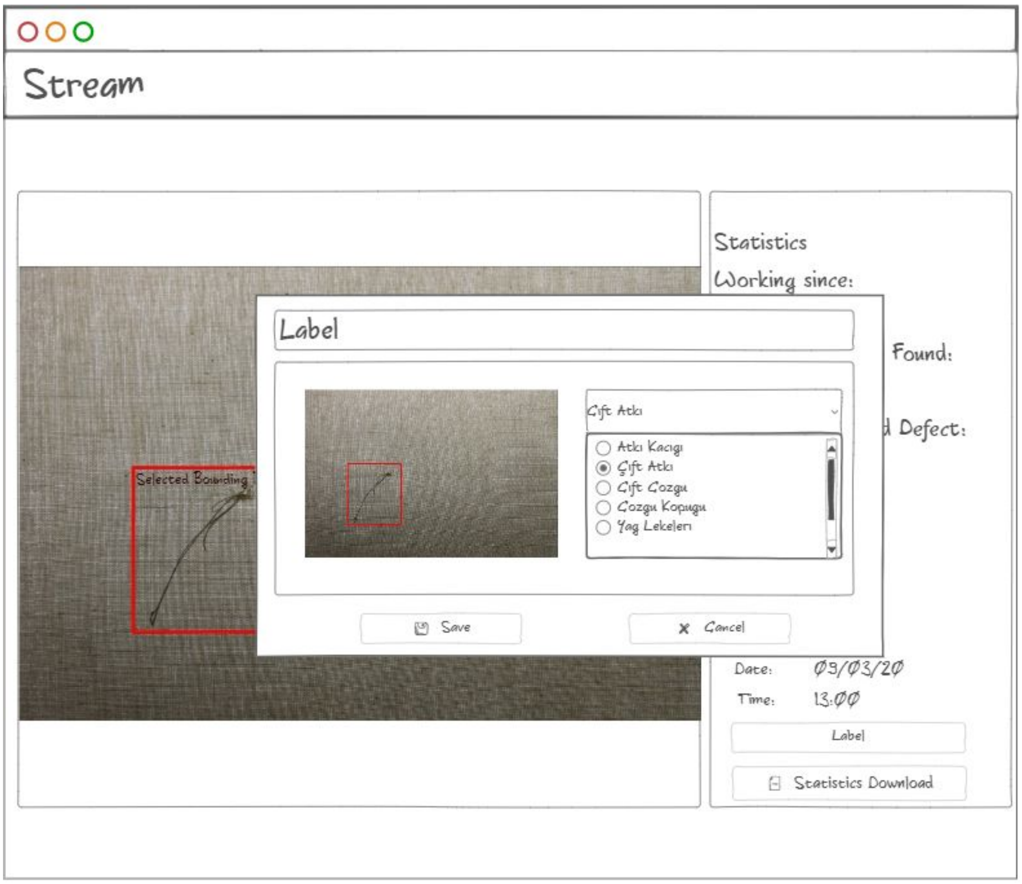


Figure 9: Name Label UI

Experts in the company will be able to take the picture of the defected fabric, add bounding box, and choose which defect it has. Then with clicking the save button the picture will be saved into the local hard disk along with the txt file which includes the name of the picture, type and bonding box of defect, date, time information.

### 3.5.5.3 Statistics Pop-up

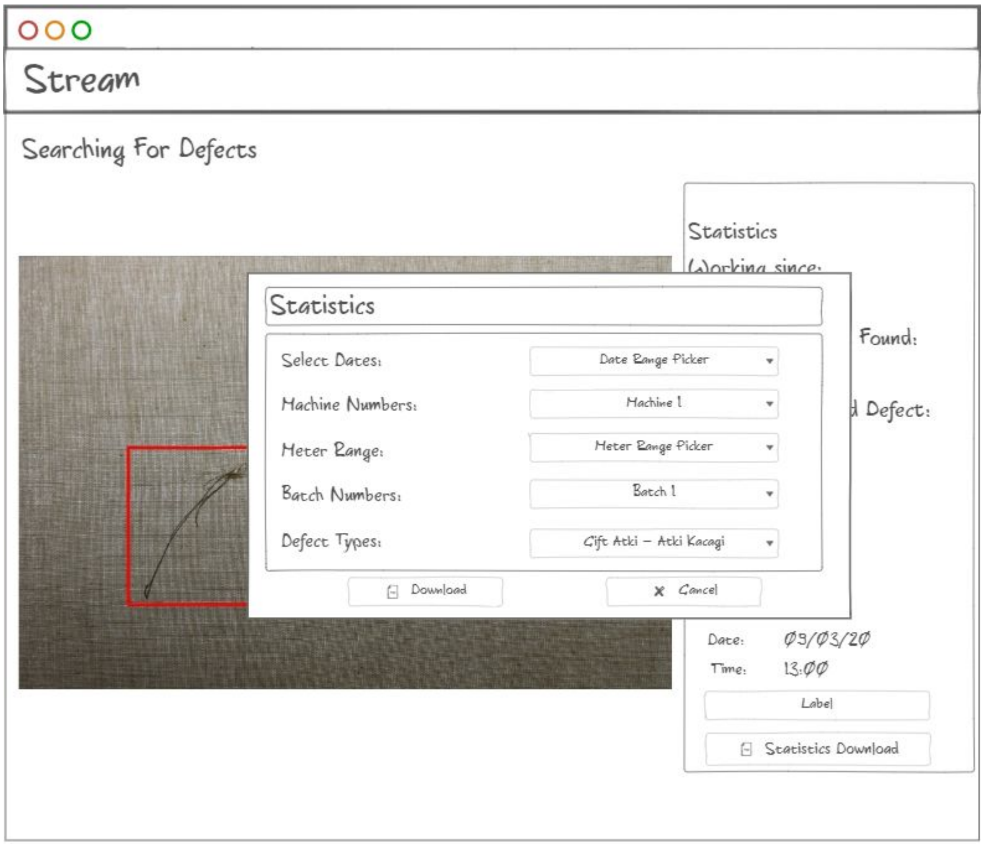


Figure 10: Statistics Download UI

The workers in the firm will be able to download the statistics with the chosen time period, machine numbers, meter range, batch numbers and defect types.

### 3.5.5.4 Defect Found Pop-up

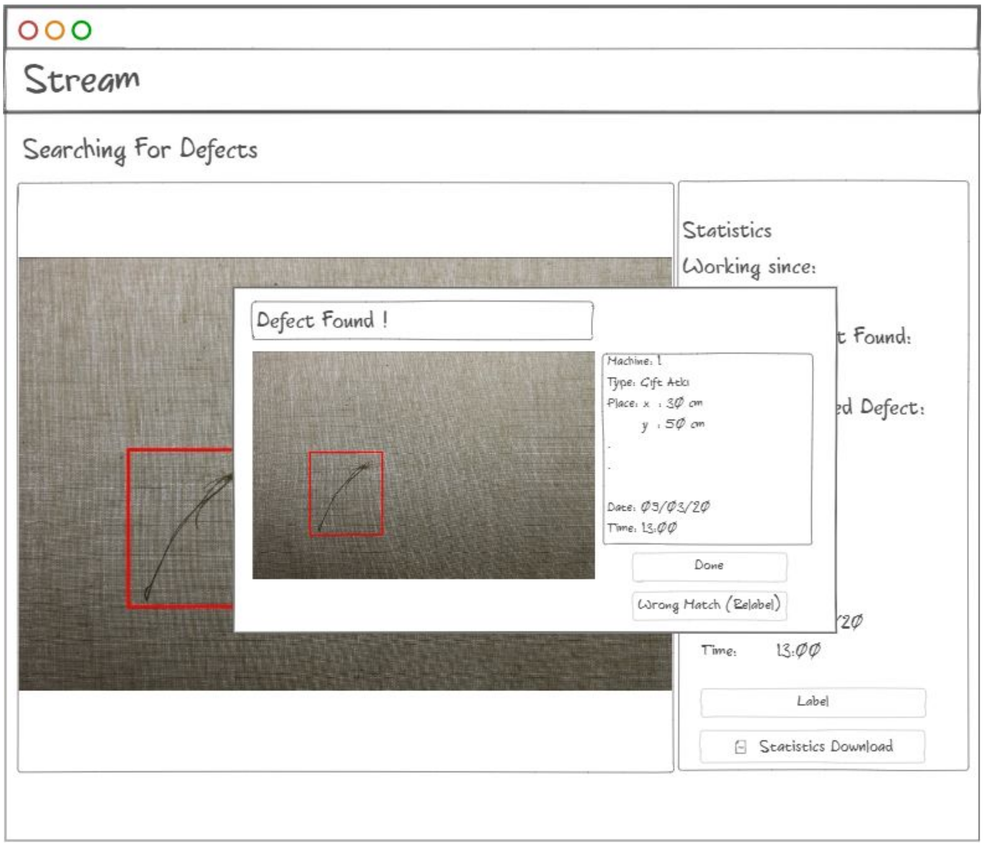


Figure 11: Defect Found UI

When the defect is found, the pop-up window will appear with some information. After the correction process, workers will be able to select the “Done” button, which shows it is a correct prediction. Also, workers will be able to select the ”Wrong Match” button, which indicates it is a wrong prediction, should be corrected. It directs the worker to the labeling page.

# 4. Other Analysis Elements

## 4.1. Consideration of Various Factors

### 4.1.1 Camera and Computer

The quality of the images and the specifications of the technology used to capture that image plays a significant role for the effective detection of anomalies in that image. In addition, when the classification of the anomalies is needed, the specifications are expected to be even higher. Therefore, the cameras to be used in the system should have higher resolution enabling the system to detect the faults with high success rate, to eliminate any part of the fabric which does not include any fault, and finally, determine the class/type of the fault with higher precision. These specifications are crucial while collecting training data as well since the quality of the image is important for CNN. In addition, both the cameras and the computer should have a long life time and therefore less frequent need for maintenance for the continuous working of the system.

### 4.1.2 Worker

The worker must follow the user interface of the system regularly and fix the faults when they are detected by the system as soon as possible so that system can continue to scan the fabrics. In addition, there must be a sufficient number of workers so that the faults can be fixed without any delay. Any deficiency in both cases results in the decrease in the fabric scanning stage of the production which are system is going to be responsible for.

### 4.1.3 Fabric Variation

The fabric type to be scanned in the factory is standard. Therefore, there is no need for including other types of current fabrics in the system. However, for further application of the system to the other companies which need the scanning of different type of fabrics, the system can be improved accordingly.

## 4.2. Risks and Alternatives

### 4.2.1 Risks

The main concern of the project is to hinder the reputation loss of the company due to sending of the fault-including fabrics to their customers, this is the reason they desire to change the current human-based fault detection system with an automatic one. Therefore, the system should perform effectively in detecting the errors with at least more than the success rate of the current system and even higher. We are aiming to reach more than 90% success rate in detecting the faults in the usage of our system and this ratio is found to be satisfactory by the company [6]. As long as we succeed to reach this success ratio there will be no risk issue in terms of the company’s reputation, which is the only possible risk issue related to the project at this stage.

### 4.2.2 Alternatives

Human-based fabric fault detections are shown to be ineffective as mentioned in the Introduction, meaning that software and hardware based systems should be applied for this purpose which are shown to have higher accuracy. There are various technologies which are based on the scanning of the images which are taken continuously in the same setting of the cloth inspection machine. CNN seems to be the leading technology for this purpose among the other ones such as auto-correlation function (AF), local binary pattern (LBP), Fourier transform (FT), wavelet transform (WT) and neural networks [7]. As a result, we have chosen CNN to solve the problem since it gives the best result although there are some other alternatives.

## 4.3. Project Plan

- Work Package 1 (March to April)
  Assigned to Alper (Leader) and Ziya.
    - Goal 1: Analyze camera requirements for data collection and prediction.
    - Goal 2: Developing Python application for data collection
    - Goal 3: Prepare data collection set-up in a factory and conduct tests.
- Work Package 2 (April to Jun)
  Assigned to all team Samet (Leader).
    - Goal 4: After data is received, apply preprocessing methods.
    - Goal 5: Start training R-CNN and measure performance.
    - Goal 5: Finalize high-level design report.
- Work Package 3 (Jun to Aug)
  Assigned all team Burak (Leader)
    - Goal 6: When the accuracy is above the limit (%70), test the system on live data at the factory.
    - Goal 7: Develop a self-improvement system for R-CNN so that the mistakes of the model can be fixed by the personnel.
- Work Package 4 (Aug to Jan)
  Assigned all team Orhan (Leader)
    - Goal 8: Prepare low-level design report and final report.
    - Goal 9: Completely integrate the project into the factory.

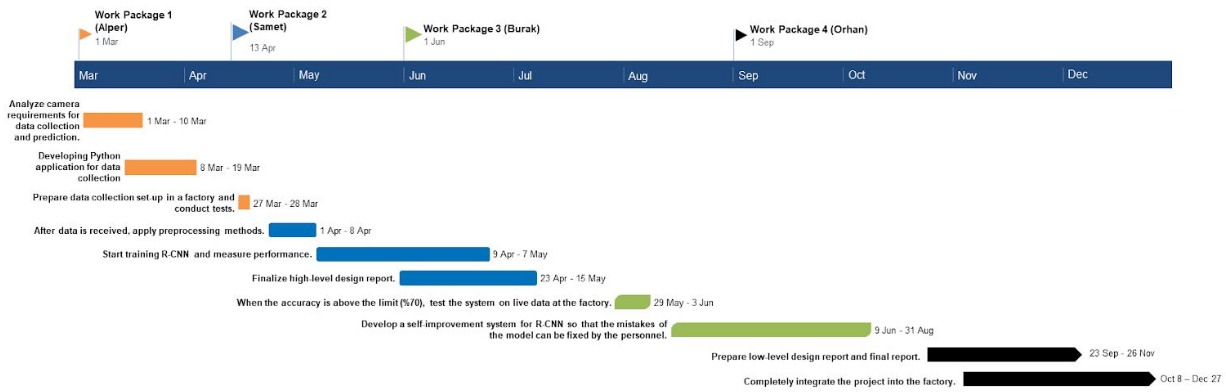


Figure 12: Gantt Chart

## 4.4. Ensuring Proper Teamwork

We are going to divide the workload based on the prior knowledge and experience of each of the team members. For instance, in this report we have assigned each UML diagram to the member who previously developed that diagram in previous courses and hence developed enough experience to create that diagram. Throughout the development of the project, we are going to apply the same rationale and divide the jobs in such a way that the most relevant member will do that job. Each of our group members has unique skills and experience and hence can contribute to the project in distinct ways. Some of us are good at developing algorithms, some of us better at user interface development. Further, we have also members skilled in Computer Vision and also in CNN. As a result, we will divide the work based on the merit of each person but this does not mean other members will not work on that. Others will work in that too but the skilled member will lead that part.

## 4.5. Ethics and Professional Responsibilities

### 4.5.1 Ethics

After the successful implementation of the project and installation of the system, the number of workers for fabric fault inspection and recovery is estimated to decrease since the checking part will be maintained by the system continuously. A warning and navigation system for the responsible workers in case of fault detection is planned to be implemented at further stages of the project. By this way, relatively less worker is estimated to be needed since the amount of total work is going to be decreased dramatically, that is, the continuous following of the fabric flowing for fault detection will not be the responsibility of the workers anymore. Only the number of workers sufficient to recover the fabric faults, which are to be determined by the system and sent to the navigation device, will be enough for the target stages of the workflow in the company: fault detection and fault recovery. In other words, the cloth inspector -currently responsible for fault detection & recovery- will be able to meet the recovery tasks of more than 2 -the current number- inspection machine on average. After the installation, not our team but the company will make a decision between dismissing any supernumerary and assigning them to other stages of workflow, and will be responsible for such a decision and its any legal and ethical consequences [ethics].

### 4.5.2.Professional Responsibilities

#### 4.5.1.1 Software Implementation

During the implementation and deployment of our project, we will follow the Code of Ethics proposed by the National Society of Professional Engineers [8].

#### 4.5.1.2 Data Privacy

Since our project involves the data which will be taken from the company, data will be secured in local storage or the database but it may be moved securely to a cloud server for training. We will not benefit from the data by selling it to third parties or organizations.

## 4.6. New Knowledge and Learning Strategies

The most important knowledge that we will need to learn is going to be R-CNN algorithms. They are quite powerful object detection algorithms and have long been used in several applications from autonomous cars to anomaly detection. Actually, thanks to off-the-shelf libraries using this algorithm is quite straightforward as we do not need to implement it from scratch. Yet, illiteracy using these algorithms without knowing how they work under the hood is easy but not much effective since we will not have much freedom in tweaking the parameters or layers of the model to increase the accuracy. Hence, we will also need in-depth knowledge of the algorithm. To this end, we aim to first get hands-on-experience to learn how the library works. That is, we are going to get our hands dirty. We will try out the online tutorials, tweak the parameters and observe how the algorithm's performance changes. In the meantime, we are going to follow online courses and the academic papers on R-CNN to get in-depth knowledge to be able to further improve our accuracy.

# 5. Glossary

**R-CNN:** Regional Convolutional Neural Network
**Worker:** Textile Expert or Cloth Inspection Expert or Quality Control Worker
**Manager:** Executives of the company
**Treatment:** Act of removing errors physically from the clothes

# 6. References

[1] M. S. Biradar, B. G. Sheeparmatti, P. M. Patil and S. Ganapati Naik, "Patterned Fabric Defect Detection Using Regular Band and Distance Matching Function," 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, 2017, pp. 1-6.

[2] K. Srinivasan, P. H. Dastor, P. Radhakrishnaihan, S. Jayaraman, "FDAS: A knowledge-based frame detection work for analysis of defects in woven textile structures", *J. Text. Inst.*, vol. 83, no. 3, pp. 431-44 7, 1992.

[3] CEO of Textile Company A, Interview, Nov 2019

[4] Ö. Kısaoğlu. (2006). Kumaş kalite kontrol sistemleri. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 12(2), 233-241.

[5] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[6] Export Manager of the Textile Company A, Interview, Feb 2020

[7] “Code of Ethics,” Code of Ethics | National Society of Professional Engineers. [Online]. Available: https://www.nspe.org/resources/ethics/code-ethics. [Accessed: 15-Feb-2020].

[8] J. F. Jing, H. Ma, and H. H. Zhang, “Automatic fabric defect detection using a deep convolutional neural network,” *Wiley Online Library*, 14-Mar-2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/cote.12394. [Accessed: 18-Mar-2020].