Stock Counter

By:

Capt Muhammad Abu Bakar

Major Sarfraz Ahmad

Capt Qaiser Khan

Capt Sufyan Zafar

Supervised by:

Dr Naima Iltaf

Submitted to the faculty of Department of Software Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad,

in partial fulfillment for the requirements of B.E Degree in Software Engineering.

June 2023

ABSTRACT

We have developed an android based mobile application named “Stock Counter”. This Mobile Application is designed to accurately count and track objects in images taken with a mobile device. Using Tenser-flow algorithms, the application can detect and identify objects in images, then track them over time to provide accurate counts. The application is highly customizable, allowing users to set up different detection parameters, filter out unwanted objects, and adjust counting thresholds. The application is useful for a variety of applications, including retail inventory management, crowd control, and traffic analysis. Key features of the application include real-time object detection and counting, customizable detection parameters, and custom reporting. By providing businesses and individuals with an accurate and efficient way to count objects. This mobile application helps to improve productivity and reduce errors.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

Certified that the work contained in the thesis, "Stock Counter" carried out by Capt M Abu Bakar, Maj Sarfaraz Ahmad, Capt Qaiser Khan and Capt Sufyan Zafar under the supervision of Dr. Naima Iltaf for partial fulfillment of Degree in Bachelor of Software Engineering is correct and approved.

Approved by

____

Dr. Naima Iltaf

CSE-DEPT MCS

DATED: ____________________

DECLARATION

We hereby declare that this project report entitled “Stock Counter” submitted to the “Software Engineering**”**, is a record of an original work done by us under the guidance of Supervisor “Dr. Naima Iltaf” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science.

Team Members:	**Signature**
Capt M Abu Bakar	______
Maj Sarfaraz Ahmad	______
Capt Qaiser Khan	______
Capt Sufyan Zafar	______

Supervisor:	**Signature**
Dr. Naima Iltaf	______

Date:

____

Place:

____

DEDICATION

In the name of **ALLAH (SWT)**, the Most **Merciful**, the Most **Beneficent.**

To our parents, without whose unflinching support and unstinting cooperation,

A work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

To begin with, there is no greater guide than **ALLAH (SWT)** and we feel blessed that **Allah (SWT)** gave us enough strength to complete this project well in time. In addition to this, we all would deeply and genuinely like to thank Ma’am Dr. Naima Iltaf for her persistent guidance and continuous support. Ma’am you are an exceptional supervisor and without you we could not have come this far.

Thank you very much.

PLAGIARISM CERTIFICATE (TURNITIN REPORT)

This thesis has 7% similarity index. Turnitin report endorsed by Supervisor is attached.

Team Members:	**Signature**
Capt M Abu Bakar	______
Maj Sarfaraz Ahmad	______
Capt Qaiser Khan	______
Capt Sufyan Zafar	______

Supervisor:	**Signature**
Dr. Naima Iltaf	______

Table of Contents

List of Tables

Table of Figures

Chapter 1

Introduction

1.1 Introduction

The aim of this application is to count objects in an image accurately and efficiently using artificial intelligence and image processing algorithms. The Stock Counter will allow users to take a photo or upload an existing image from mobile phone gallery, and the application will then use Tensor-Flow algorithms to identify and count the objects in the image. The application will also have features such as the ability to select specific objects to count and exclude others, as well as the ability to view and export the count data. The Stock Counter will be useful for a variety of uses, including inventory management and scientific research. It will save time and reduce errors associated with manual counting, and provide a more accurate and reliable count of objects in an image.

1.2 Statement of Problem

Counting objects in an image manually can be a time-consuming and error-prone process, especially when dealing with a large number of objects or objects that are difficult to count. Manual counting can also be subjected to human error and inconsistencies, leading to inaccurate results. The scope of application depends on its specific features and capabilities. Some object counter applications may be designed to count only specific types of objects, such as people or vehicles, while others may the capacity to count a greater variety of items. This can have significant consequences in applications such as inventory management, quality control, and scientific research, where accurate object counting is crucial for decision-making. Furthermore, the current solutions available for object counting in images often require specialized equipment or expertise, making them inaccessible or costly for many users.

1.3 Statement of Goals & Scope of the Project

The Mobile Application project of object counting by images aims to provide an affordable and accessible solution manual counting of objects in a stock problem, using artificial intelligence and image processing algorithms to count objects in an image accurately and efficiently. Therefore, the main problem that the Stock Counter aims to solve is the need for a reliable and efficient tool for object counting in images, which is accessible and user-friendly for a broad variety of users. In conclusion, the Stock Counter application will be a useful tool for anyone who wants to rapidly and reliably count things in an image, and will offer a range of features and functionalities to meet the needs of different users.

1.4 Challenges

1.4.1 Scale and Perspective

The projection of a 3D scene onto a camera's 2D sensor is referred to as perspective in imaging. The geometry imbalance of scene items and their proximity to the camera determine this connection. Perspective determines how objects are distributed within a scene and can cause a significant difference in object scale, with objects closer to the camera appearing larger and having better resolution while objects farther from the camera appear smaller and having poorer resolution. The need for a model to learn to extract characteristics across scales and to learn how to separate individual objects from dense groups when those objects are tiny and poorly resolved presents a substantial challenge for object counting approaches. Researchers have come up with a variety of solutions to the issue of size and perspective in photos, which will be covered in more detail in the following sections. But the issue is still open and difficult to solve. Additionally, even though this is a problem for many object counting issues in natural pictures, perspective concerns won't always become an issue for object counting issues.

1.4.2 Inter-Object Variance

In contrast to inter-object variance, which arises as a result of a unique attribute of an object class, scale variance originates as a natural property of perspective within complex situations. Consider, for instance, how diverse individuals may be in terms of a variety of features, such as height, age, gender, and dress preferences. Or, take into account the fact that cars might be found in a vast range of hues, dimensions, and forms. For object counting approaches, which would need to account for this variation among object types, this poses an extra issue. Multiple variables, such as an imbalance in the distribution of object attributes within specific datasets, may aggravate this. The intricacy of object counting issues is evident when compared to the issue of perspective-based variance. It takes sophisticated algorithms to learn functions that describe the possibly high variance within an object class in order to count objects.

Chapter 2

Literature Review

2.1 Introduction

Object and stock counting is the important working need in retail stores, warehouses, medical stores, and farm houses of cattle, car parking and many other fields of life. It is hectic and time consuming job. Engineers have been working to automate the counting methods for long. The literature suggests that Stock Counting is a rapidly evolving field, with a range of application approaches being employed. Several studies have been conducted to develop methods for counting objects in images, with various application approaches being employed.

2.2 Existing solutions and their limitations

2.2.1 CNN approach for Object Detection and Counting

Convolutional neural networks (CNNs) are essential to many deep learning-based computer vision applications. CNNs are a subset of neural networks that Yann LeCun, a forerunner in deep learning, developed in the 1980s and are effective in detecting patterns in multidimensional landscapes. CNNs are used to handle various forms of data, but they are particularly effective at processing pictures.

A convolutional neural network is a piece of software that takes meaningful values out of the input picture using a number of convolutional layers. A convolutional layer has multiple filters, which are square matrices that move over the picture and register the weighted sum of pixel values at various points. Each filter performs distinct duties, has unique values, and draws out unique aspects from the input image. A collection of "feature maps" is what a convolution layer produces.

To detect a pyramid of visual patterns, convolutional layers are stacked on top of each other. For instance, the bottom layers will create feature maps for corners, edges that are both vertical and horizontal, and other intricate patterns. The following layers are capable of identifying more intricate and varied patterns, such circles and grids. As you go further into the network, the layers will be able to recognize more complex things, including people, animals, homes, trees, buses, trucks, and vehicles.

Pooling layers are a common technique used by convolutional neural networks to gradually reduce the size of their feature maps while keeping the most prominent elements. The primary pooling layer now employed in CNNs, max-pooling, maintains the determined value in a patch of pixels. As an example, if we use a pooling layer with a size of 2, it will pick patches of 2 by 2 pixels from the feature maps created by the layer before it and maintain the one with the highest value. The most important elements are kept while the map sizes are split evenly. CNNs can oversimplify their capabilities and become less sensitive to the transposition of objects across pictures by pooling layers.

The final product of the convolution layers is compressed into a one-dimensional matrix, which serves as an arithmetic representation of the characteristics included in the image. Then, using a sequence of "fully connected" layers of artificial neurons, the matrix is used to map the characteristics to the desired output of the network.

Picture classification is the most fundamental job for convolutional neural networks. In this task, a picture is sent as input to the network, which then outputs a list of values that indicate the likelihood that the image belongs to one of several classes. Consider training a neural network to recognize each of the 1,000 kinds of chemicals contained in the popular open-source dataset ImageNet. Your output layer will then have 1,000 mathematical outputs, each of which understands the likelihood that the image belongs to one of those classes.

The ideas of local receptive fields, sparse weights, and parameter sharing are all present in convolutional neural networks. Convolutional neural networks are better suitable for learning from picture input because of these three ideas, which provide them a convincing transformation and scale invariance compared to other neural networks.

A study by Zhou et al. (2018) proposed a technique for applying deep learning to the task of counting items in photos. The method involved training a CNN to envisage the number of objects in an image, given the image as input. The method was shown to achieve high accuracy on a variety of object counting tasks.

Main disadvantage are;

- Requires very large amount of data in order to perform better than other techniques.
- The categorization of images in various positions.
- Negative instances.
- The Coordinate Frame.

A study by Kang et al. (2019) proposed a method for counting objects in images using an attention-based application approach. The method involved training a CNN to predict object locations and counts. The method was shown to achieve state-of-the-art performance on several object counting benchmarks.

The main disadvantage of the attention mechanism is that it adds more weight parameters to the model, which can increase training time especially if the input data.

One study by Lempitsky et al. (2010) proposed a method for counting objects in images using a regression-based approach. The method involved training a regressor to estimate the number of objects in an image, given a set of image features. The method was shown to achieve high accuracy on a variety of object counting tasks.

Disadvantage is false detections.

2.2.2 YOLO model for Object Detection and Counting

YOLO (You Only Look Once) is a popular object detection model known for its speed and accuracy. It was first introduced by Joseph Redmon et al. in 2016 and has since undergone several iterations, the latest being YOLO v7. The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improve performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is. YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

YOLO v7, the latest version of YOLO, has several improvements over the previous versions. One of the main improvements is the use of anchor boxes.

Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes. It uses nine anchor boxes, which allows it to detect a wider range of object shapes and sizes compared to previous versions, thus helping to reduce the number of false positives.

A key improvement in YOLO v7 is the use of a new loss function called "focal loss." Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects. Focal loss battles this issue by down-weighting the loss for well-classified examples and focusing on the hard examples—the objects that are hard to detect.

Additionally, compared to earlier versions, it has a greater resolution. YOLO v3 utilizes this to process photographs at a resolution of 608 x 608 pixels, greater than the 416 x 416 resolution. Therefore, this model can more precisely recognize small objects.

Limitations of YOLO v7

- It is a capable and effective object identification system with several drawbacks.
- Like many other object detection algorithms, this method put extreme effort to distinguish and identify little things. In congested environments or when items are far away from the camera, it might not be able to accurately detect the objects.
- It is not flawless in detecting items at various sizes, which can make it difficult to notice objects that are either extremely huge or very little in comparison to the other objects at that particular scene.
- It is light intensity sensitive as well as other environmental conditions have impact on it, therefore, in real-world applications where lighting conditions might change, it could be challenging to employ.
- As, it is computationally exhaustive, this can become a drawback in many cases, such as, on limited resource devices like smartphones or other edge devices, it is challenging to operate it in real-time.
- Speed is one of the major disadvantages of YOLO v7. It is capable of producing 155 frames of images per second, which is a significantly faster rate of production. The peak processing speed for even the simplest YOLO model was 45 frames per second. Because of this, it may be used in delicate real-time applications like surveillance and autonomous driving, where higher processing speeds are crucial.

2.2.3 SSD model for Object Detection and Counting

Real-time object detection is a feature of SD. Faster R-CNN employs boundary boxes that are generated by a region proposal network to categorize things. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second, far below what real-time processing needs. SSD speeds up the process by eliminating the need for the region proposal network. To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN's accuracy using lower resolution images, which further pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN. (Accuracy is measured as the mean average precision mAP: the precision of the predictions.)

The SSD object detection composes of 2 parts:

- Extract feature maps, and
- Apply convolution filters to detect objects.

Feature maps are excerpted by SSD using VGG16. After that, it uses the Conv4_3 layer to find objects. To illustrate, we draw the Conv4_3 with a spatial resolution of 8 x 8 (it should be 38 x 38). It generates four object predictions for each cell (also known as location). Each prediction consists of a boundary box, 21 scores for each class (plus one additional class for no object), and we choose the class with the greatest score as the bounded item's class. No of the depth of the feature maps, Conv4_3 makes a total of 38 x 38 x 4 predictions: four predictions per cell. Many forecasts, as predicted, do not include any objects. SSD reserves the class "0" to denote that it does not have any objects.

As contrast to using the delegated region proposal network, SSD resolves in a fairly simple manner. Tiny convolution filters are used to determine the location and class scores. After mining the feature maps, SSD smears three-by-three convolution filters for each cell to represent forecasts. Each filter produces 25 channels.

Disadvantage is that, Insufficient high-level properties may not be produced by a neural network's shallow layers to do prediction for small objects. Therefore, SSD performs worse for little items than for bigger stuff. A significant quantity of data is needed to train the sophisticated data augmentation need.

Chapter 3

Requirement Specification

3.1 Introduction

The Stock Counter is incorporating most of the option existing on mobile application that can count stock items in an image accurately and efficiently with some added features to best facilitate the user regarding low down counting errors.

3.2 Purpose

The Stock Counter could be beneficial for the store managers, warehouse managers and shopkeepers because boxes and objects are being sold and purchased frequently. It is challenging for managers to count stocked objects one by one and doing the same job many times in a day is time taking and difficult. Physically counting the objects can also lead to counting errors. So, this application can be very well-versed to save time, provide an accurate count and maintaining record for as long as manager want with minimum effort.

3.3 Main Scope

- User can register / sign up or login if already registered.
- User can capture Image or Import from gallery.
- Application can process the image.
- Application will show result.
- User can save that result.
- User Information is secured.

3.4 Classes

System User

This software is primarily to be used by shopkeepers, suppliers and warehouse managers who will be getting benefits of the application facility.

Operating Environment

To run this application user requires the most common product these days, a smart phone with and android operating system.

- Hardware: Smart phone.
- Software: Android Studio, python.
- Operating System: Android Operating System.
- Operating System Version: Android 9 and above.

Design and Implementation Constraints

The system requires a Wi-Fi connection, which is critical to its operation. Since it is a stock counting application, majority of the work, namely by clicking pictures or import from files will require a connection with the server.

3.5 User Interfaces

Hardware Interfaces

The application runs on an android based smart phone.

Software Interfaces

Android Studio (Using Kotlin Programming Language)

Firebase (As a Server)

Tensor Flow (Python Programming Language)

3.6 Application Interfaces

Figure 1: Startup Page

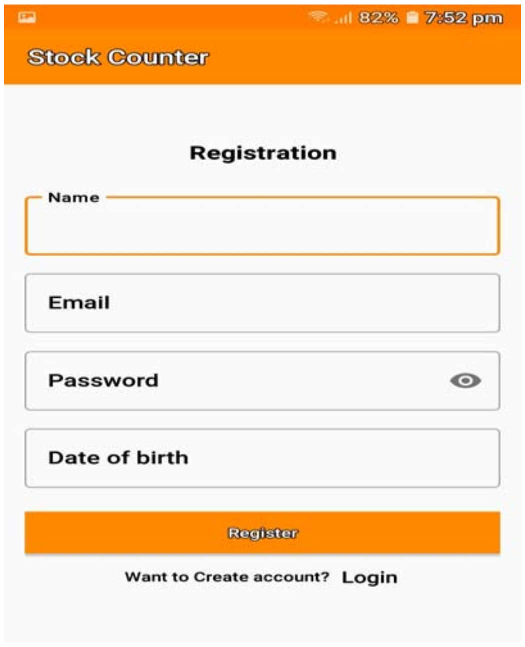

Figure 2: Signup Page

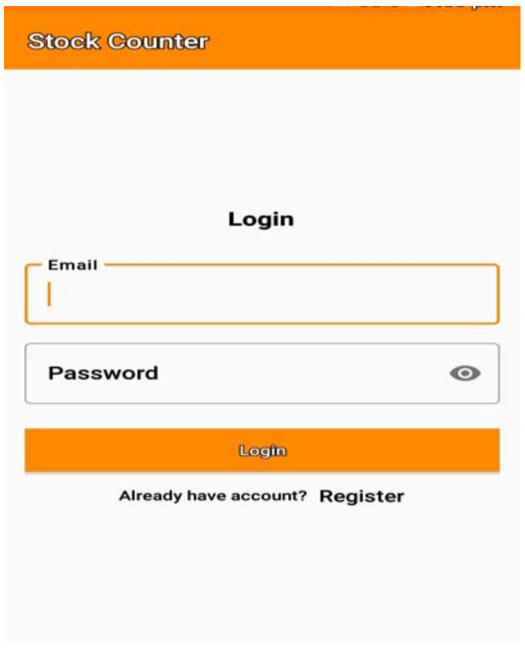

Figure 3: Login Page

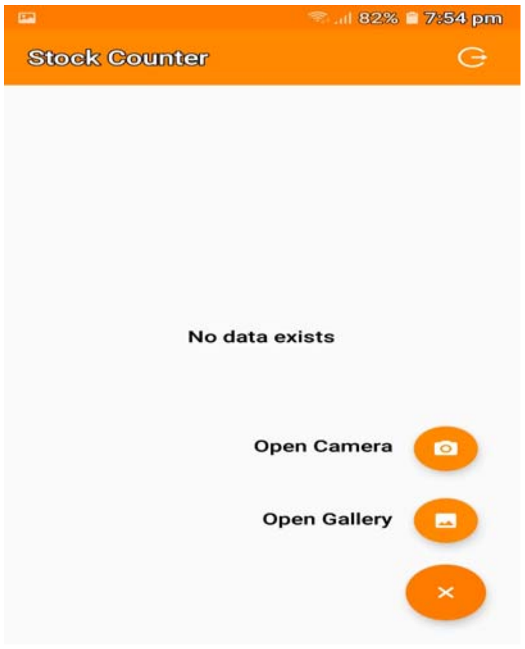

Figure 4: Home Page

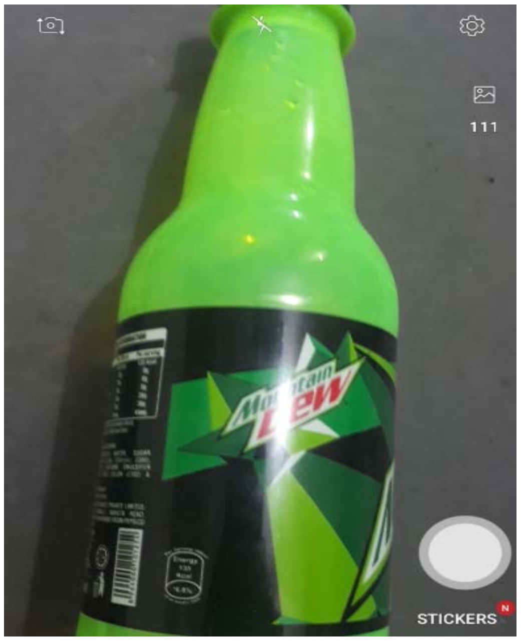

Figure 5: Camera Access View

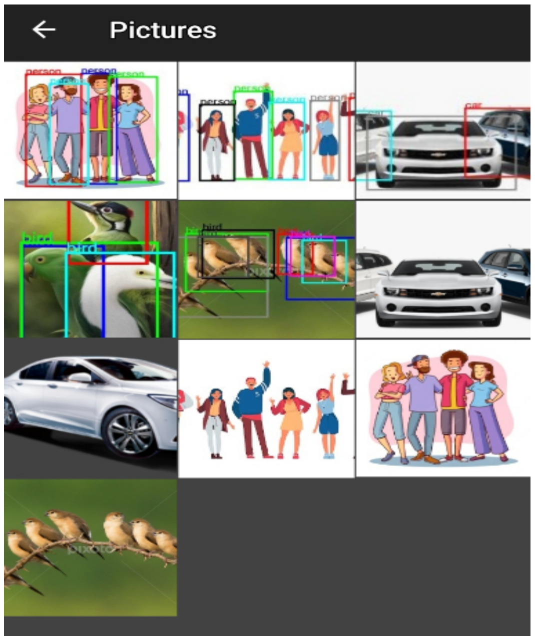

Figure 6: Gallery Access View

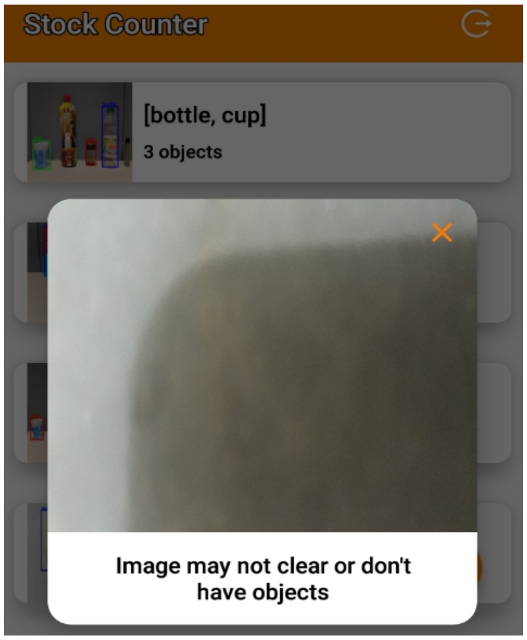

Image may not clear or don't have objects

Figure 7: Unclear Image Notification

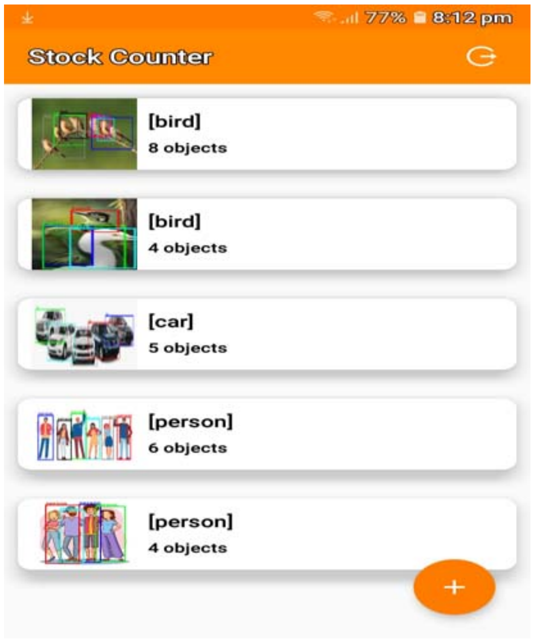

Figure 8: Log

3.7 Functional Requirement

Table 1: Sign up

Feature	**Sign up**
Description	This feature allows the user to sign up using his / her own email and a new password.
Priority	3 (Lowest)
Normal Flow	User enters his name, email, password and date of birth. The application stores the credentials provided by users and generates an account.
Alternate Flow	If any of the asked credentials by the application are left empty by the user, application will not generate the account and inform user to fill the requirement by showing the notification as “Empty Strings”.
Functional Requirement	The system is able to provide users with the required credentials’ platform. The system is able to read and process the credentials provided by the users. The system is able to create a user account.

Table 2: Login and Logout

Feature	**Login and Logout**
Description	This feature allows the user to login his / her accounts using his own email and password at a screen provided by the stock counter.
Priority	3 (Medium)

Normal Flow	User enters his email and passwords. The application verifies and displays the main screen. User clicks on logout button and the application logs out.
Alternate Flow	If id or password mismatch occurs the system will shut down until user provides correct credentials.
Functional Requirement	The system is able verify whether the id and password is correct. User's accounts opens only when the id and password has been verified. The system is able to log out when user clicks the logout icon.

Table 3: Capture Image and Scanning

Feature	**Capture Image and Scanning**
Description	This feature of the application provides the users with the facility of camera access to capture the image using "Open Camera" option available on the application homepage.
Priority	6 (Highest)
Normal Flow	User enters his email and passwords. The application verifies and displays the main screen. User clicks on "Open Camera" option. User captures the stock image as normal image capturing. By click on the tick on the top right corner at the screen at this stage submits the image for the objects identification and stock count.
Alternate Flow	If a blank or unclear image is captured, application notifies the user to

	recapture the image by giving a notification as “Image may not be clear or don’t have objects”.
Functional Requirement	The system is able to provide user with “Open Camera” option. The system is able to open mobile phone’s camera when user select associated option. System is able to take input image of stock provided by the user. System is able to scan the uploaded image for stock count.

Table 4: Upload Image from Gallery and Scanning

Feature	**Upload Image from Gallery and Scanning**
Description	This feature of the application provides the users with the facility of uploading the image from mobile phone’s gallery using “Open Gallery” option available on the application homepage.
Priority	6 (Highest)
Normal Flow	User enters his email and passwords. The application verifies and displays the main screen. User clicks on “Open Gallery” option. User uploads the stock image from mobile phone’s gallery. By click on the tick on the top right corner at the screen at this stage submits the image for the objects identification and stock count.
Alternate Flow	If a blank or unclear image is captured, application notifies the user to recapture the image by giving a notification as “Image may not be clear or don’t have objects”.

Functional Requirement	The system is able to provide user with “Open Gallery” option. The system is able to open mobile phone’s gallery when user select associated option. System is able to take input image of stock provided by the user. System is able to scan the uploaded image for stock count.

Table 5: Count Results

Feature	**Count-Results showing to Users**
Description	This feature of the application provides the users with the facility of seeing count results with the input image.
Priority	6 (highest)
Normal Flow	After taking input from the user as an uploaded image or captured image, system provides the output on the screen as types of objects and number of objects in that specific image.
Alternate Flow	If a blank or unclear image is captured or uploaded, application notifies the user to recapture the image by giving a notification as “Image may not be clear or don’t have objects”.
Functional Requirement	The system is able to recognize and count objects in the image. System is able to produce results on the screen for the users.

Table 6: Storage and Retrieval

Feature	Storage and Retrieval
Description	This feature of the application provides the users with the facility of seeing count results of all scans performed and taking screenshots of the log.
Priority	3 (Medium)
Normal Flow	All the scans are stacked on the home screen back view, providing the small size of image along with the types of objects and number of objects in the image. Users can see and take screen shots of scan details. Users can also take screenshots of scanned image at the time of seeing scan results of single image.
Functional Requirement	System is able to produce list of scans on the homepage for the users. System allows the user to take screenshots of images and results as well as save in the mobile phone's gallery.

3.8 Non-Functional Requirement

Availability: The application is available for the users all the time, provided mobile is in working sate and the internet connection is available.

Accuracy: The application is accurate enough in counting the objects even in challenging lighting when the objects are partially obscured.

Backup: Backup is enabled for the application. If a user upgrades to a new Android-powered device, application restores the user's previous data when the application is logged in.

Performance: The application is able to count objects quickly and a good accuracy, without any lags or delays.

Reliability: The application is reliable and able to count objects consistently, without crashes.

Security: The user's data is safeguarded by the program, which is secure.

Scalability: The program can handle a huge number of images and objects and is scalable.

Compatibility: The application is compatible with different types of smartphones and operating systems.

Usability: The application is easy to use, with a simple and intuitive interface that requires minimal training.

User-Friendly Interface: The application has a simple and intuitive interface that allows the users to easily capture and upload images as well as submitting images for counting objects.

Accessibility: By adhering to accessibility guidelines, the program is usable by all users, including those with impairments.

Performance under different conditions: The scans are performed accurately and quickly under various conditions, including different lighting conditions and object sizes.

Battery Friendly: The application does not drain the battery of the user's phone excessively, ensuring that it can be used for longer periods without the need for frequent charging.

Chapter 4

Software Design Specification

4.1 Introduction

Scope of this document focuses on various aspects of Stock Counter project which includes architectural, procedural design and design assumptions, dependencies and constraints. A wide range of design considerations is explained, from design restrictions to design goals and guidelines construct the scope of this document. Diagrams are used to furnish the documents to make it unambiguous on reader mind.

4.2 Architecture Diagram

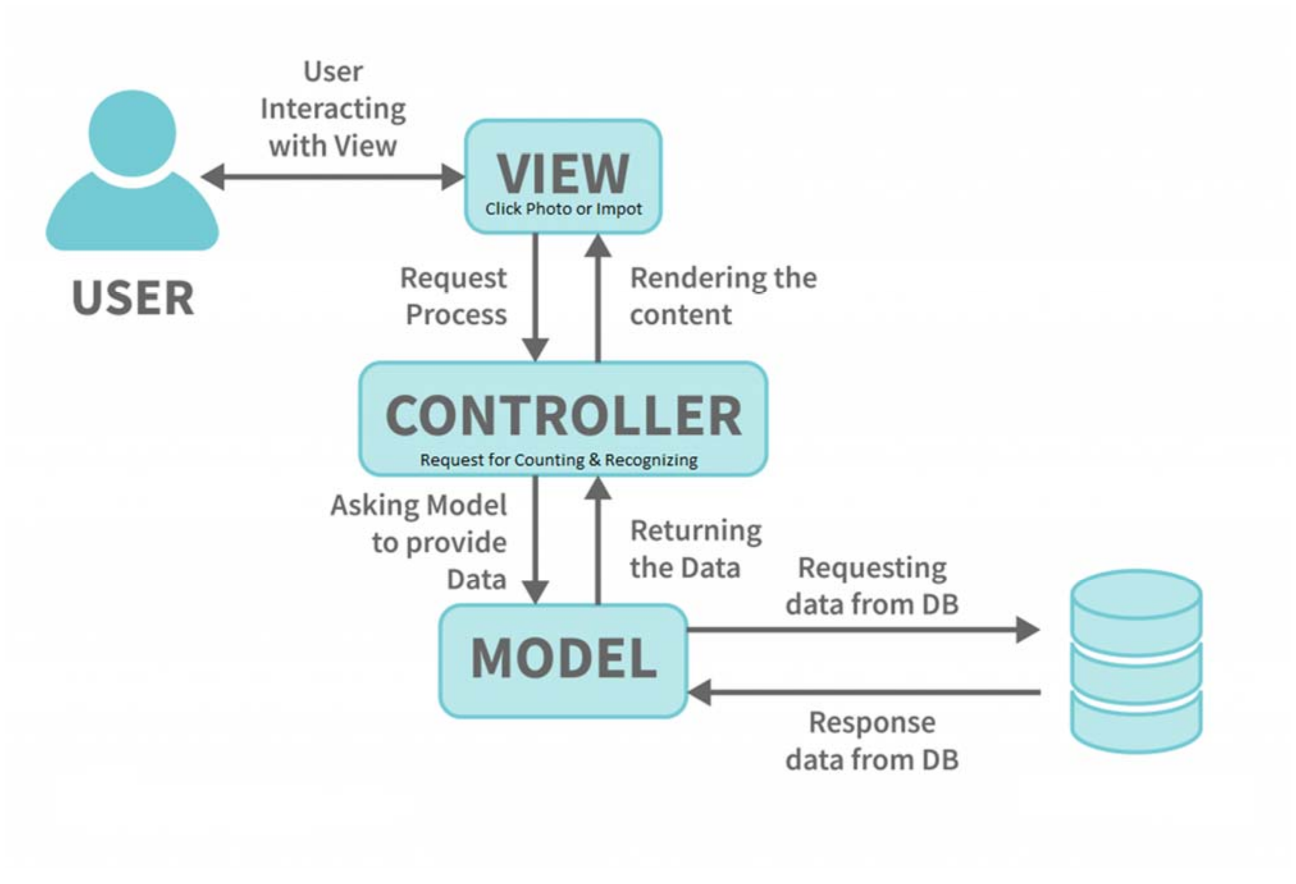

Figure 9: Architecture Diagram

4.3 Use-Case Diagrams

Sign up

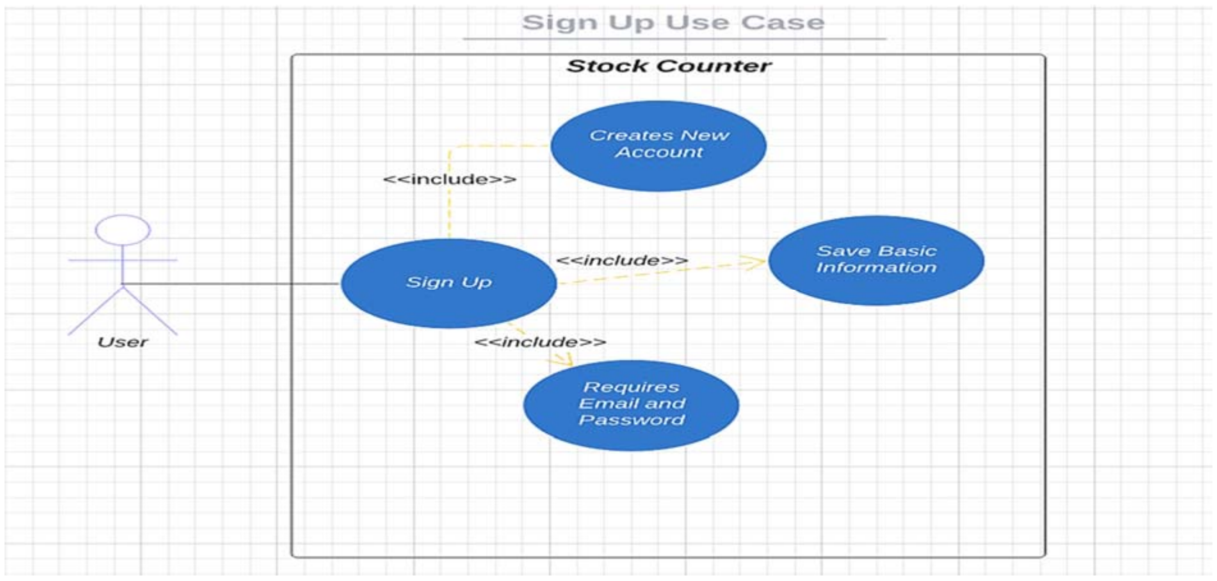

Figure 10: Sign up - Use Case Diagram

Login and Logout

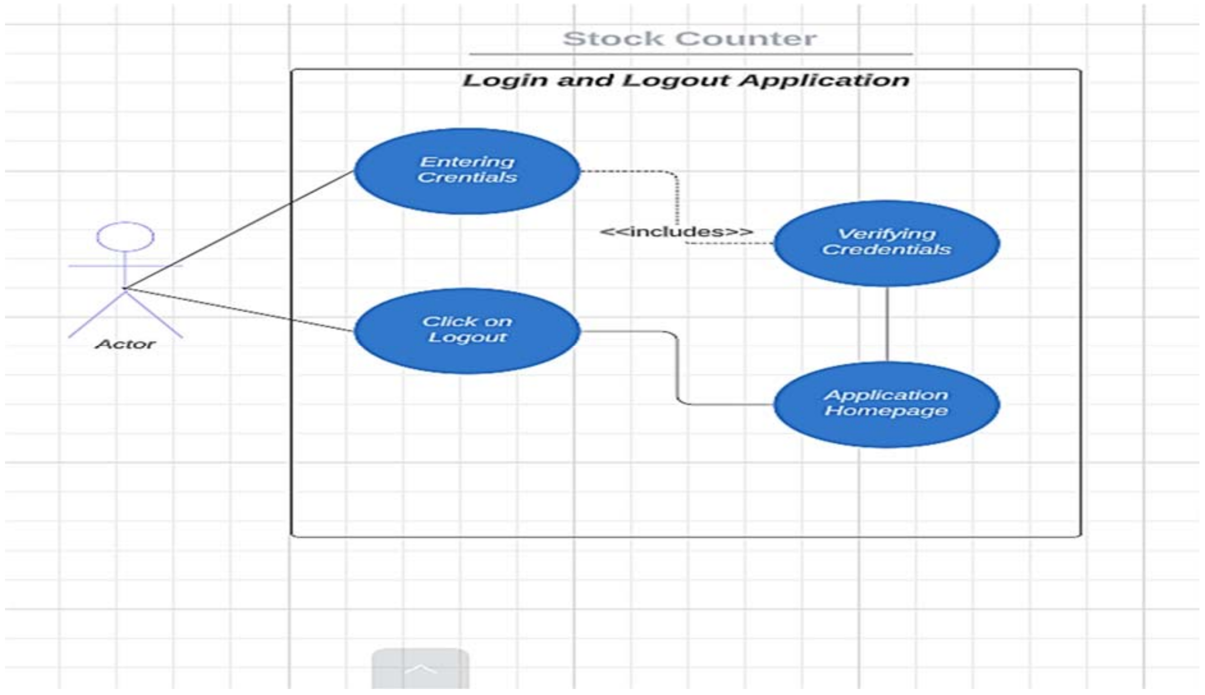

Figure 11: Login and Logout - Use Case Diagram

Capture Image and Scan

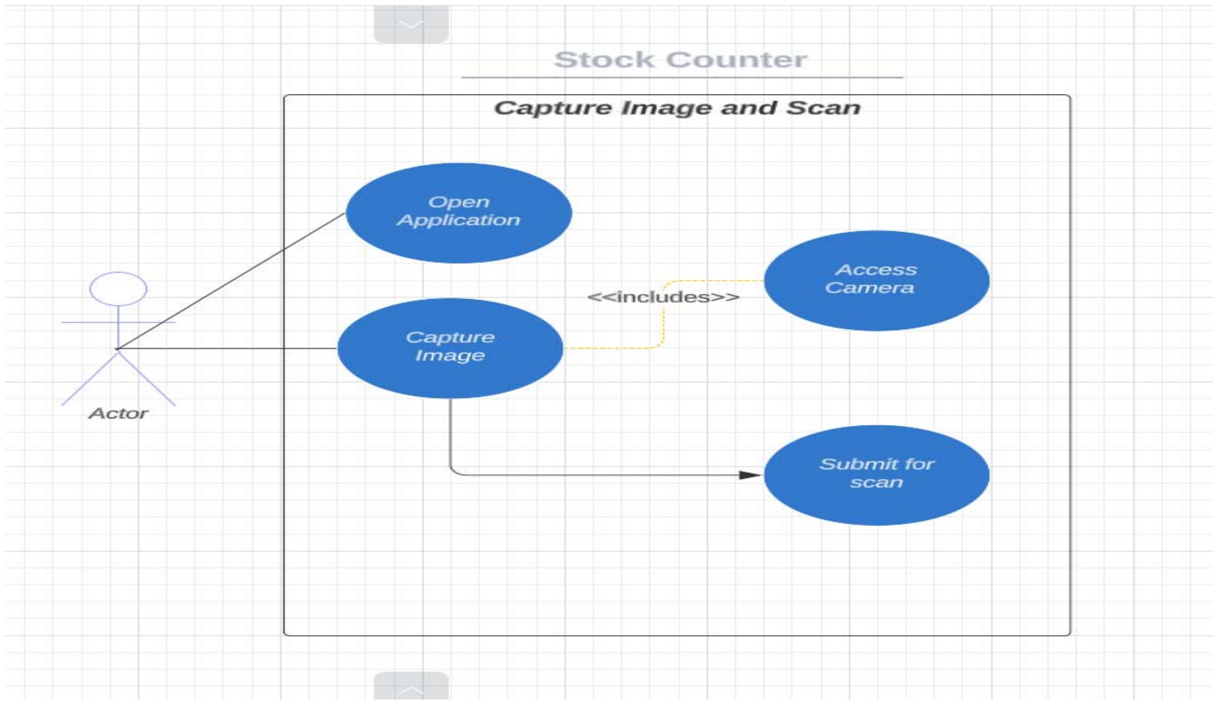

Figure 12: Capture Image and Scan - Use Case Diagram

Upload Image and Scan

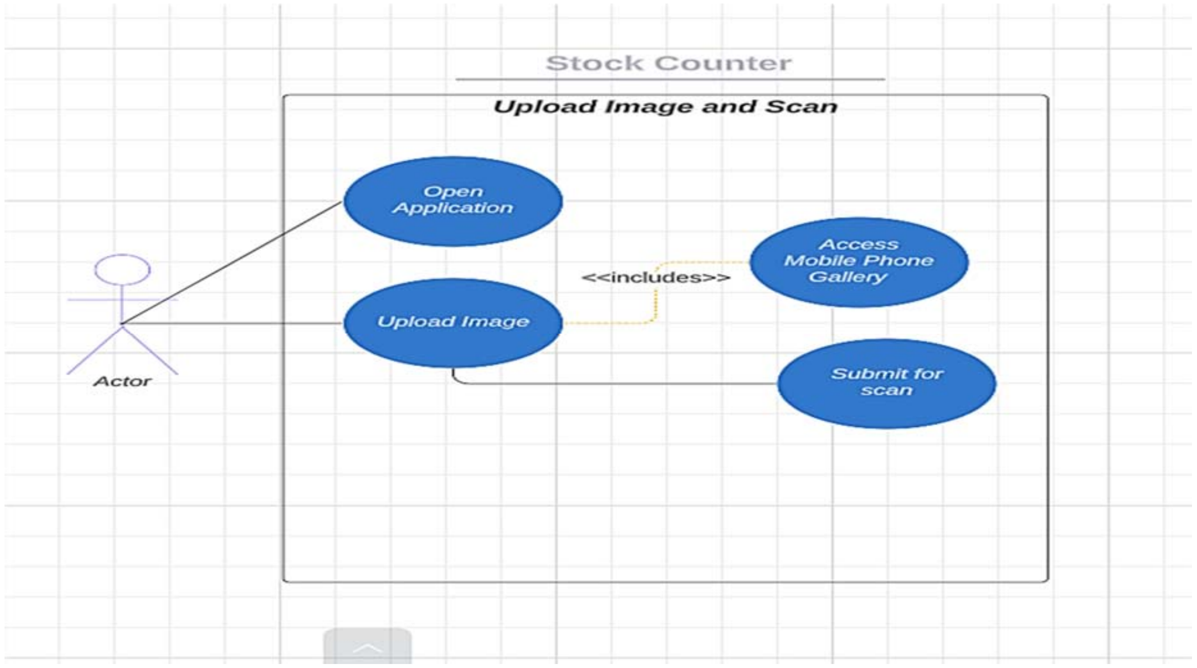

Figure 13: Upload Image and Scan - Use Case Diagram

Count-Results to Users

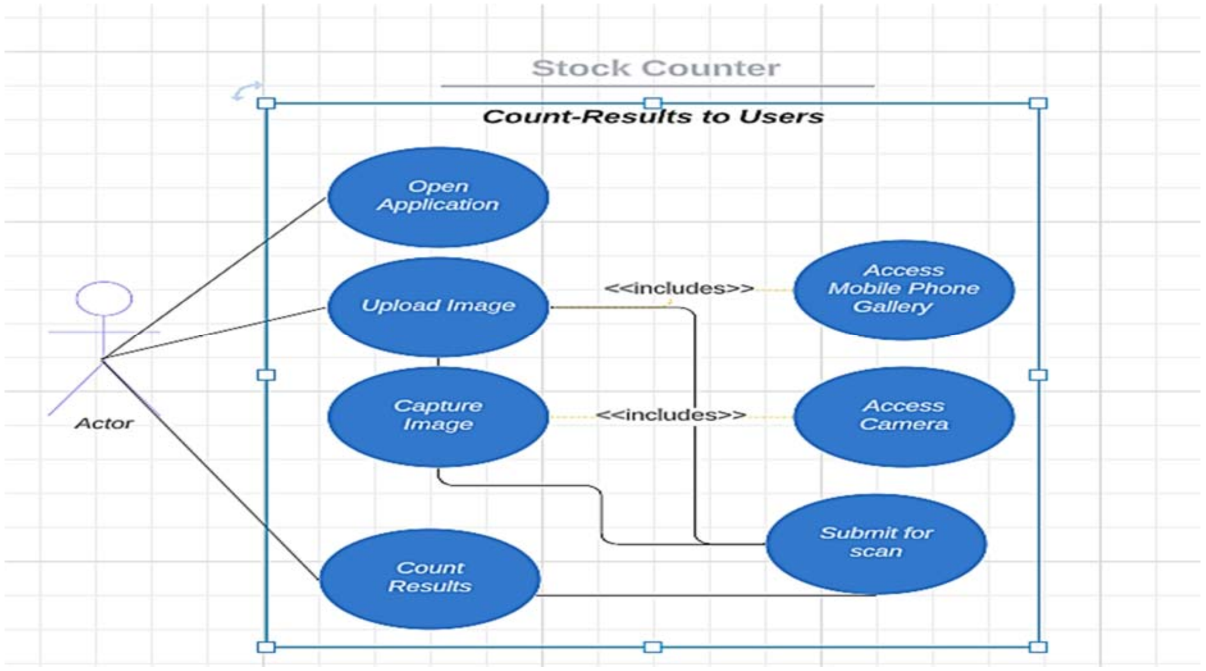

Figure 14: Count-Results to Users - Use Case Diagram

Storage and Retrieval

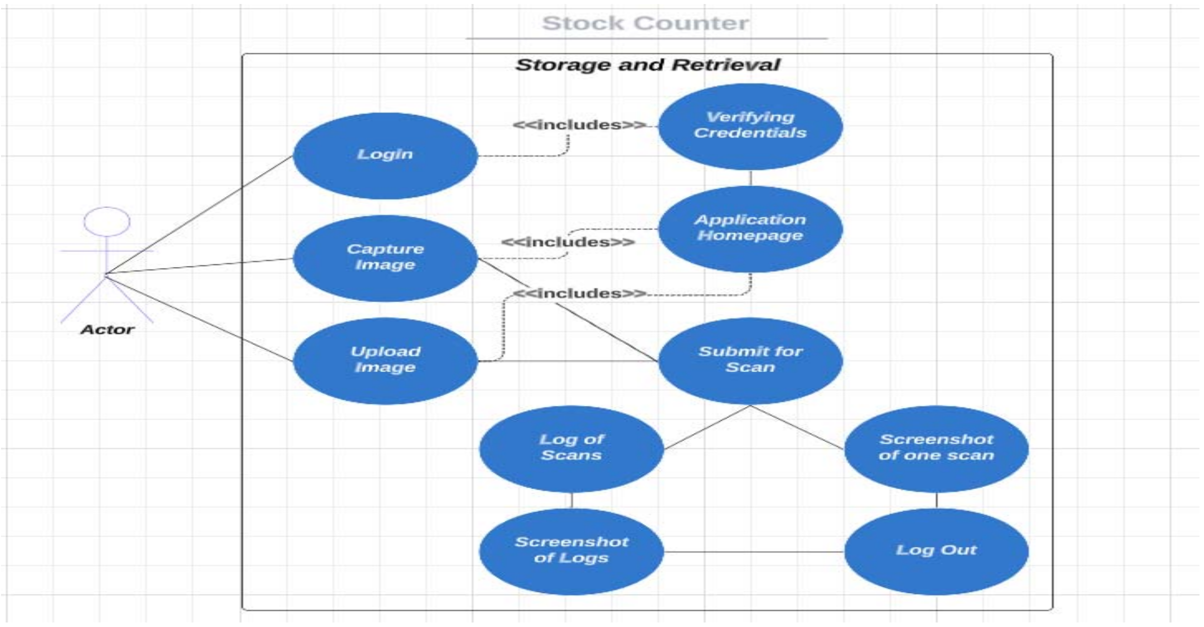

Figure 15: Storage and Retrieval - Use Case Diagram

4.4 Activity Diagrams

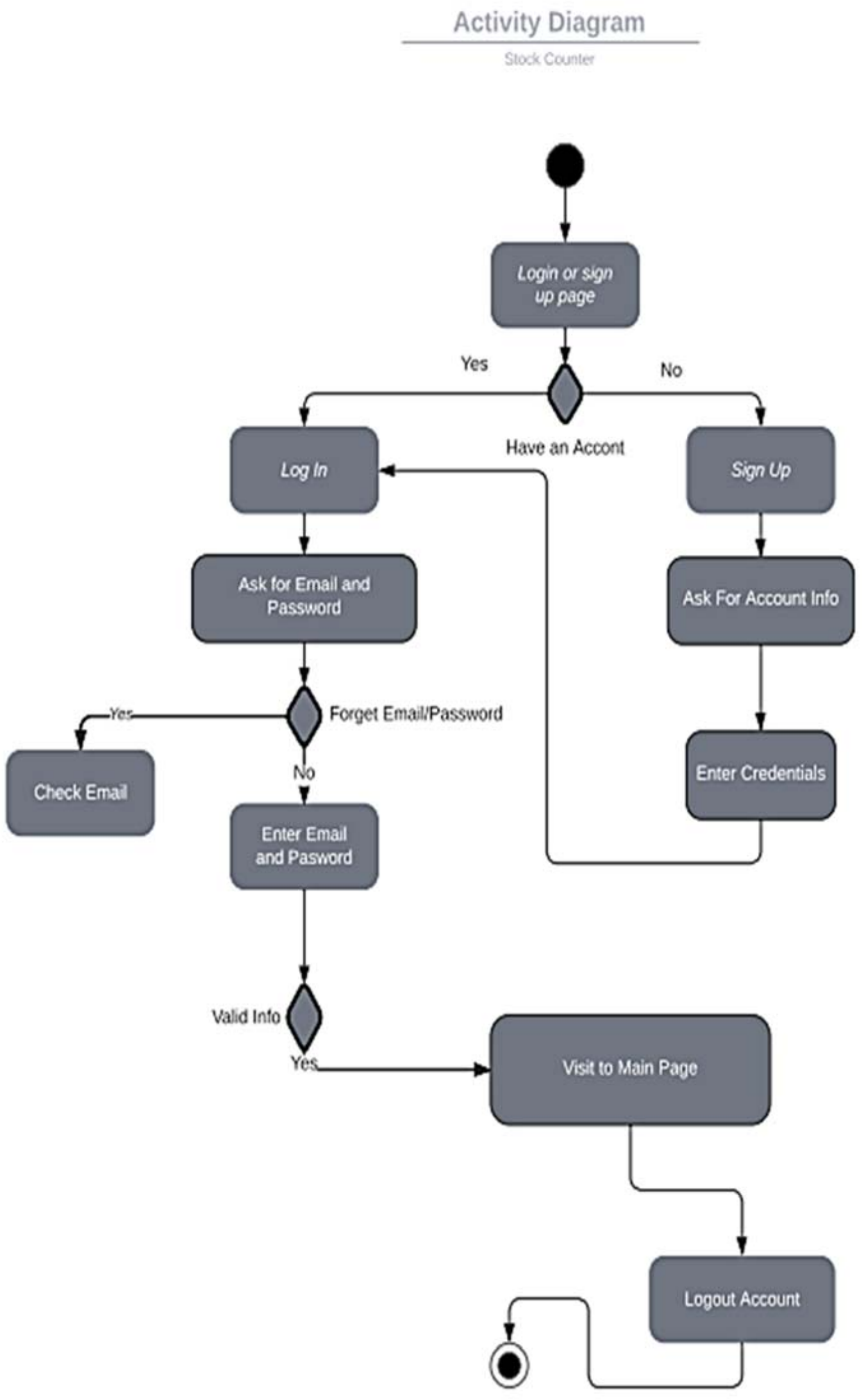

Figure 16: Activity Diagram (a)

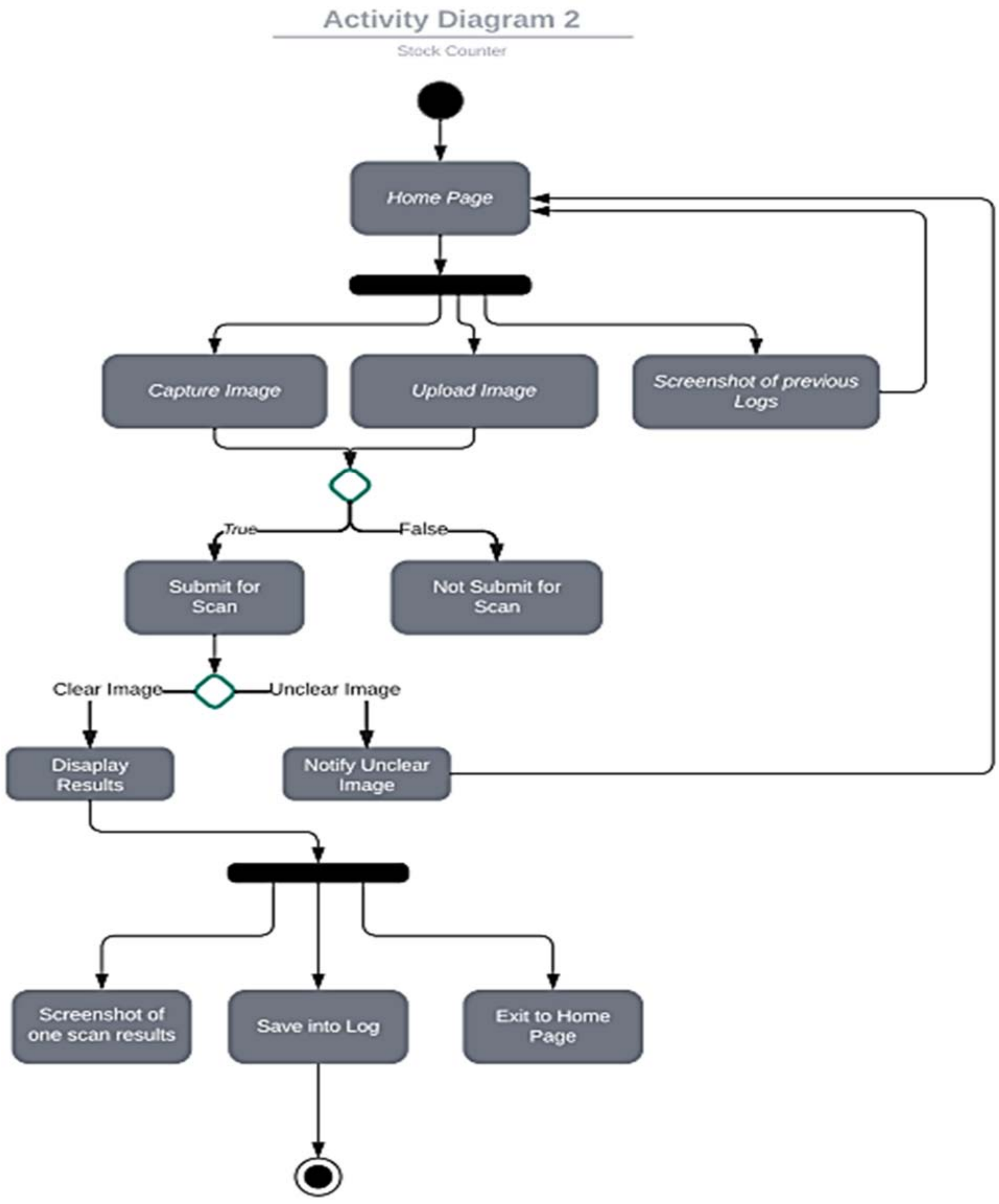

Figure 17: Activity Diagram (b)

4.5 Sequence Diagrams

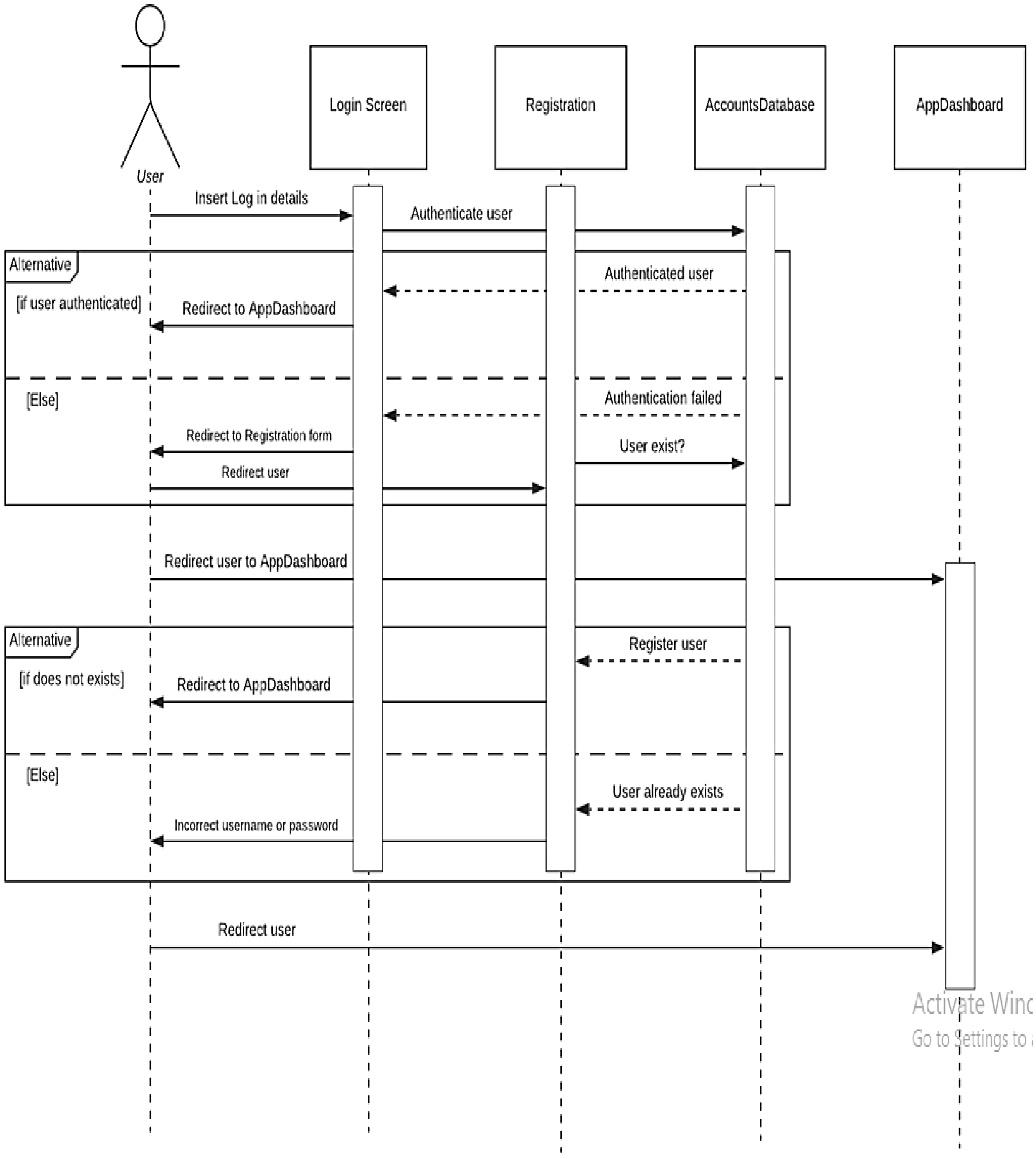

Figure 19: Sequence Diagram (a)

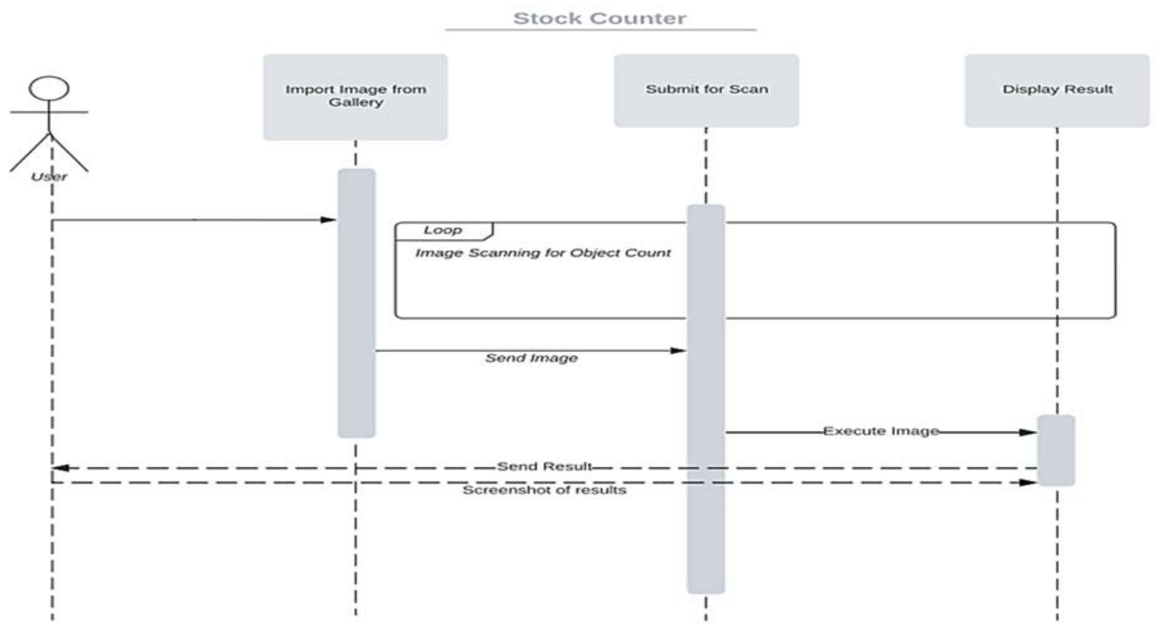

Figure 20: Sequence Diagram (b)

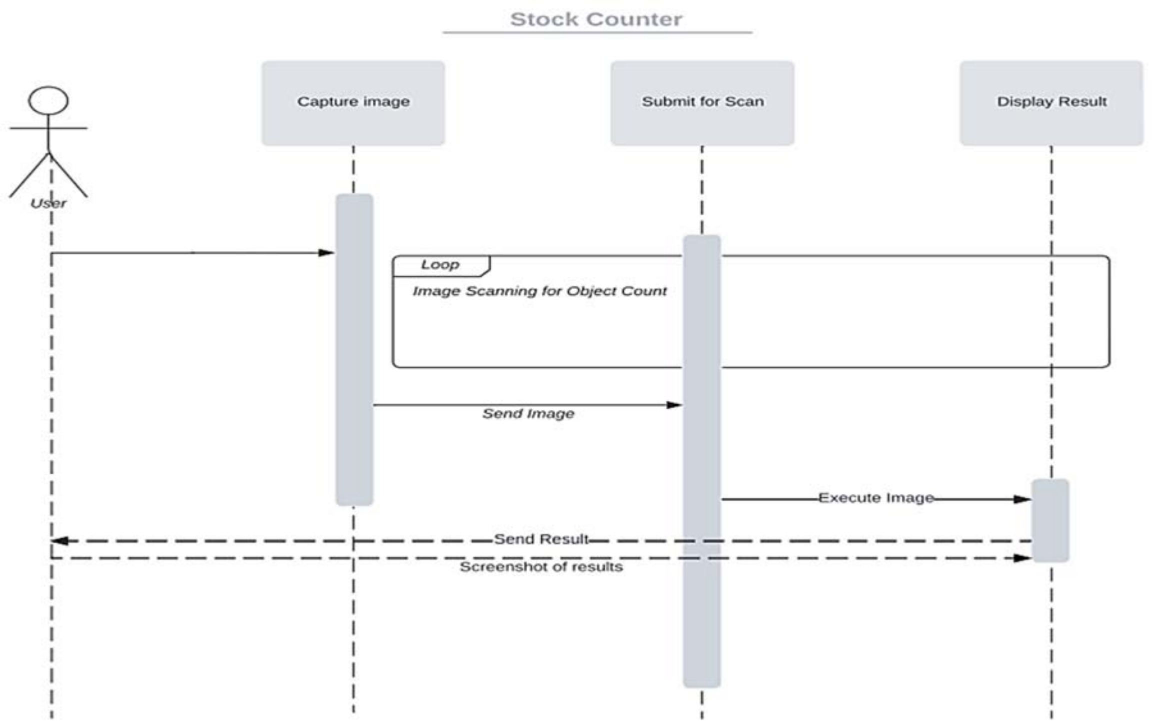

Figure 21: Sequence Diagram (c)

4.6 Class Diagram

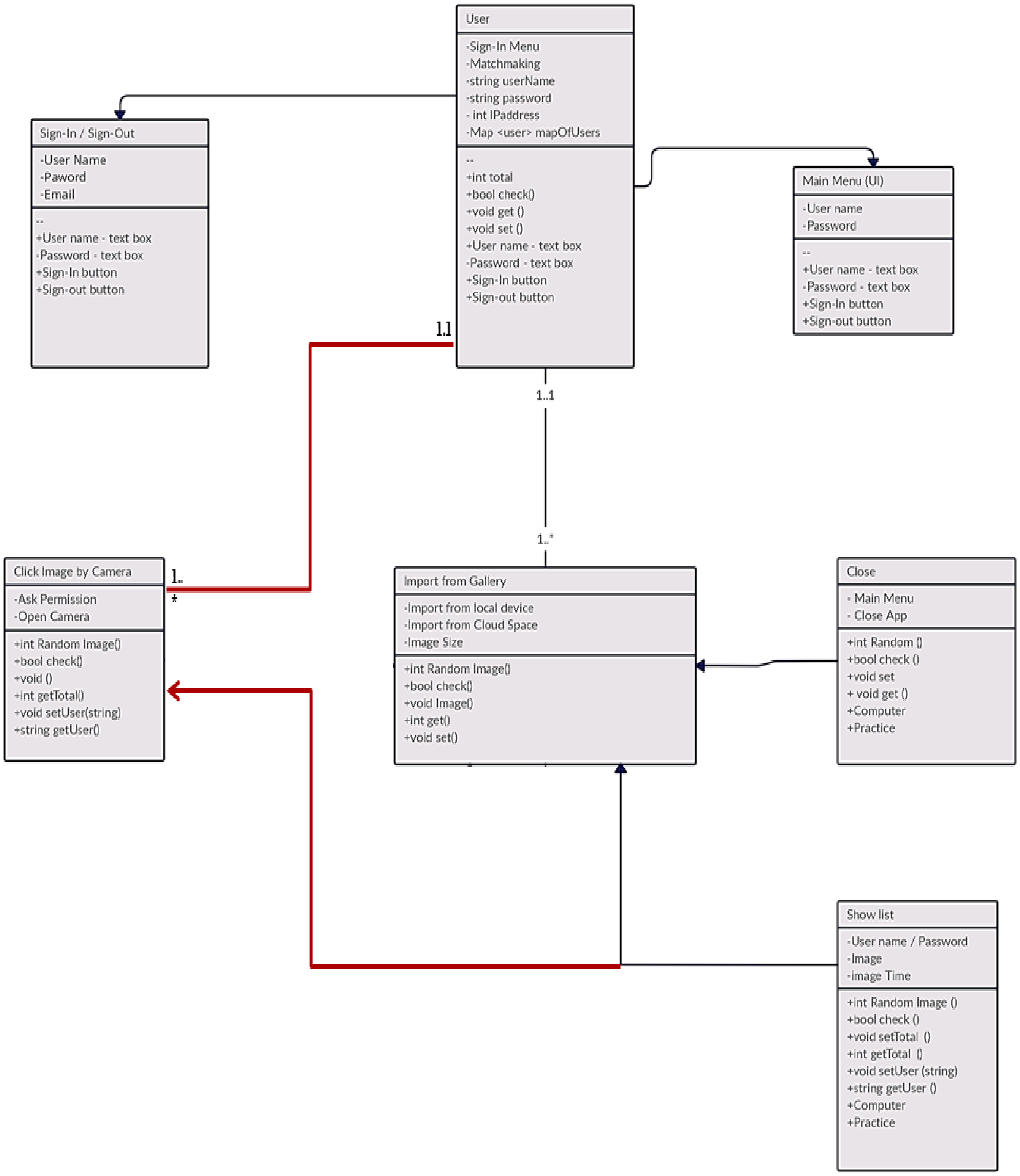

Figure 22: Class Diagram

Chapter 5

Implementation

5.1 Introduction

This Chapter describes about different phases of implementation in details.

Programming Language Used

Kotlin is used as the programming language for the development of application. Tensor Flow model is used to train the model and Python Programming Language is used for development of the model.

Development Tools

Android Studio (Flamingo 2021-2022) is used to develop the application.

Database

The Firebase is used to manage the data which is stored in application.

Operating System

This application is designed for the android platform. So, the application will run on all android and tablet on which android 9 is supported.

5.2 Implementation Phases

Stock Counter application has been implemented in the follow phases:

- User Interface
- Backend Functionalities
- Database Configuration
- Integration of Firebase with Android Application

5.3 User Interface

First the appearance of application at user side is set according to the function and services that application offers for the user. The screens are linked in the order of their appearance.

Icons and buttons are categorically placed presenting individual function of the application.

5.4 Backend Functionalities

The functionalities and their logic are implemented in android development environment in code that is not visible to the user. The flow of logic makes a function work properly through a number of input, queries, check, interacts with firebase and outputs the desired results.

5.5 Database Configuration

The data that interacts with the application is placed in database tables of Firebase from where it is fetched as needed. Firebase provides a suite of tools and services to support the creation of top-notch applications by developers. The process of configuring Firebase involves setting up and connecting your application to Firebase's services, such as authentication, real-time database, cloud storage, and more.

5.6 Integration of Database with Android Application

A cloud tool called Firebase Remote Configuration enables you to modify the functionality and look of your application without having to wait for users to download an update. The functionality and look of your application are controlled by the in-application default values you define when utilizing Remote Configuration. You may later alter in-application default values for all application users or for certain user segments using the Firebase interface or the Remote Configuration backend APIs. The frequency with which your application checks for updates and applies them might have little to no performance impact. Your application controls when updates are applied.

5.7 Modules

Login Module

Module is linked with the database and forms the data access layer of the application. Business Layer consists of all the functions that are then accessed from the view. Usernames and Passwords stored in the database are compared with username and password entered by user.

Homepage Module

This is the main module, which greets the user following successful login. This is used to access all the different functionality of the application and can be considered as a main menu. It is consisted of two tabs. Using first tab the user can see the option that is on right bottom corner and the second tab is used to display the list of result.

Image Module

In this module user can click image by camera or import from phonebook. A class with the same name is provided by the Image module and is used to represent a PIL image. The module also offers a variety of factory operations, including tools for generating new pictures and loading images from files.

Result Module

After execution the image this module can display the result of counted stock by model. In this module a list of scan results is shown on main page.

Chapter 6

Testing and Analysis

6.1 Introduction

Software testing is the process of inspecting a software program or system to find flaws, faults, or problems in its operation, effectiveness, security, and user interface. In order to make sure the software satisfies the end-user's expectations and business requirements, it must be verified and validated against the requirements and standards for which it was designed. Software testing is a crucial component of the software development lifecycle since it helps to assure the quality and dependability of the program, reduce the risk of failure, and have a positive influence on the business or organization.

6.2 Verification and Validation

Verification refers to software evaluation procedure to determine whether it meets its design and development specifications and requirements. This process involves reviewing and testing the software's code, design documents, and other artifacts to ensure they meet the specified criteria. Validation, on the other hand, refers to the process of evaluating a software product to determine whether it meets the user's needs and expectations and operates as intended in the real-world environment. This process involves testing the software's functionality, performance, security, and usability against the user's requirements and expectations. Together, verification and validation help ensure the quality and reliability of the software and reduce the risk of failure and negative impact on the business or organization.

6.3 Functional Testing and Testing Plans

The purpose of functional testing is to ensure that the software performs its intended functions precisely and successfully. This testing process comprises creating test cases based on the requirements and specifications of the product and running them to validate the behavior of the software. The test cases may include inputs, expected outcomes, and the steps to be taken when executing the test. With the appropriate tools and techniques, functional testing may be carried either manually or automatically. It is a critical phase in the software development lifecycle that guarantees the program's reliability and quality, lowers the risk of failure, and benefits the business or organization.

Functional Testing plans

The plans of functional testing are as following;

Requirements

Review the software's requirements and specifications to identify the functional areas that need to be tested.

Test Cases

Create test cases based on the identified functional areas. Each test case should outline the required inputs, anticipated results, and test-running procedures.

Test Environment

Determine the test environment needed for the functional testing, including the hardware and software requirements, test data, and test tools.

Test Execution

Conduct the functional tests by executing the test cases in the test environment.

Defect Reporting

Document and report any defects or issues encountered during the functional testing process.

Test Results

Analyze the test results to find out, if the software fulfills its intended requirements and specifications.

Test Coverage

Ensure that the functional testing covers all the functional areas of the software and that all test cases have been executed.

Test Automation

Consider automating the functional testing process using tools and techniques to increase efficiency and accuracy.

Table 7: Registration Testing

Test Case Name	Registration Testing
Test Case Number	1
Description	This test is aimed at checking the functionality of registration module of our application. The user should be able to choose a name, email, password and date of birth to successfully register into the application.
Testing Technique used	White Box Testing
Preconditions	The user should have application installed on his/her android based mobile phone. While using the application, mobile phone must be connected to the internet.
Input	User provides his basic credentials as; name, email, password and date of birth.
Steps	User enters the required credentials and presses “Register”.
Expected Output	After getting the required credentials in a right form, application generates the account and directs user to the homepage.
Alternative Path	Empty Fields – In case of any empty field of credentials, “Empty Strings” message will be shown on the screen. Invalid Email – In case of an invalid email, “Not a valid Email” message will be shown on the screen. Duplicate Email – In case of an already existing email address, “It is a Duplicate Email” error message will be shown on the screen.
Actual Output	Confirmed as expected.

Table 8: Login Testing

Test Case Name	Login Testing
Test Case Number	2
Description	This test is aimed at checking the functionality of login module of our application. The user should be able to enter his email and password to successfully login into the application.
Testing Technique used	White Box Testing
Preconditions	The user should already have registered (created an account). While using the application, mobile phone must be connected to the internet.
Input	User provides his required credentials as; email and password.
Steps	User enters the required credentials and presses "Login".
Expected Output	After getting the required credentials in a right form, application enters the related account and directs user to the homepage and "Logged in successfully" message is shown on the screen.
Alternative Path	Invalid Email – In case of an invalid email, application is shut down. Invalid Password – In case of an invalid password, application is shut down.
Actual Output	Confirmed as expected.

Table 9: Capture Image and Submit for Count Testing

Test Case Name	Capture Image and Submit for Count Testing
Test Case Number	3
Description	This test is aimed at checking the functionality of Capture Image and Submit for Count module of our application. The user should be able to open options available on the home-screen of the application, select the option of "Open Camera", capture image and submit for count results.
Testing Technique used	White Box Testing
Preconditions	The user should be logged into the account. While using the application, mobile phone must be connected to the internet.
Input	User captures the image and submits the image for object count.
Steps	User selects the option of "Open Camera". Gets camera access. Captures the image and submits for count. Application produces the count results on the screen.
Expected Output	After getting the image, application runs the algorithm for scan and displays the results to the user.
Alternative Path	Unclear / Blurred / Blank Image – In case of blank / blurred / unclear image, application notifies the user with the message "Image may not be clear or don't have objects".
Actual Output	Confirmed as expected.

Table 10: Upload Image and Submit for Count Testing

Test Case Name	Upload Image and Submit for Count Testing
Test Case Number	4
Description	This test is aimed at checking the functionality of Upload Image and Submit for Count module of our application. The user should be able to open options available on the home-screen of the application, select the option of "Open Gallery", upload image and submit for count results.
Testing Technique used	White Box Testing
Preconditions	The user should be logged into the account. While using the application, mobile phone must be connected to the internet.
Input	User uploads the image and submits the image for object count.
Steps	User selects the option of "Open Gallery". Gets gallery access. Uploads the image and submits for count. Application produces the count results on the screen.
Expected Output	After getting the image, application runs the algorithm for scan and displays the results to the user.
Alternative Path	Unclear / Blurred / Blank Image – In case of blank / blurred / unclear image, application notifies the user with the message "Image may not be clear or don't have objects".
Actual Output	Confirmed as expected.

Table 11: Results of Scans Testing

Test Case Name	Results of Scans Testing
Test Case Number	5
Description	This test is aimed at checking the functionality of Log of Scans module and individual scan result of application. The user should be able to save scan results and log of all previous scans.
Testing Technique used	White Box Testing
Preconditions	The user should be logged into the account. While using the application, mobile phone must be connected to the internet.
Input	User submits the image for object count.
Steps	User submits captured or uploaded images for count. Application produces the count results on the screen. User downloads the image with tags of counted results and names of objects into the mobile phone's gallery by clicking on the option of "Save Object". Current count results are added to the logs in the application on the homepage. User views the entire previous log along with the current scan results.
Expected Output	User can save and view current scan results as well as see previous scan details.
Alternative Path	N/A
Actual Output	Confirmed as expected.

Table 12: Logout Testing

Test Case Name	Logout Testing
Test Case Number	6
Description	This test is aimed at checking the functionality of Log Out module of our application. The user should be able to log out of the application by clicking log out icon on the top right corner of the homepage.
Testing Technique used	White Box Testing
Preconditions	The user should be logged into the account. While using the application, mobile phone must be connected to the internet.
Input	User clicks on the log out icon on the top right corner of the homepage.
Steps	User clicks on the log out option. Application generates confirmation message for the user as "Are you sure, you want to logout!", along with the options of "YES" and "CANCEL". If user selects "CANCEL", user stays on the homepage. If user selects "YES", user is logged out.
Expected Output	User logs out of the application and application shows login page again.
Alternative Path	N/A
Actual Output	Confirmed as expected.

6.4 Unit testing

Individual units or components of the software program in this project are tested separately from the rest of the system to confirm their operation and behavior. Unit testing checks that each piece of software operates as intended and adhere to its specified requirements and standards. Developers can do manual unit testing or use automated testing tools and approaches. It is a crucial step in the software development lifecycle that guarantees the software's quality and dependability and reduces the possibility of mistakes and flaws.

6.5 Integration testing

The goal of integration testing is to confirm that the components comply to their set criteria and standards and work correctly together. Test cases must be created and run in order to confirm the behavior of the units in order to test their integration. It is possible to carry out this testing in a top-down, bottom-up, or combination manner. It is an integral aspect of the software development lifecycle that guarantees the program's quality and reliability and lowers the likelihood of errors and defects resulting from the interaction of various sections or units.

6.6 System testing

In this testing procedure, the software's usability, performance, security, and compliance with planned requirements and specifications are all examined. Both human and automated tools and methods are available for system testing. It is a crucial step in the software development lifecycle that helps to guarantee the quality and dependability of the product and reduce the risk of failure and adverse effects on the company or organization. System testing ensures that a program accomplishes duties as intended. Unlike white box testing, it is a black box testing that aims at the functionality of the system rather than the inner workings.

It may guarantee that all forms of user input produce the intended results across the application. In the software development process, system testing is the third level of testing. It is often performed before to acceptance testing and following integration testing.

6.7 Summary

The system is tested thoroughly. All the modules have been tested for validation and results were found 95% accurate. Some test cases with failure status were reviewed again and the missing functionality was added. The accuracy of the application depends on several factors, such as the quality of the camera, lighting conditions, and the complexity of the scene being monitored. In general, Stock counter applications may be less accurate in crowded or cluttered environments, or in low-light conditions.

Chapter 7

Future work

7.1 Introduction

There is a scope for a number of advanced features that can still be added in Stock Counter in order to make it more efficient and interactive for use. The use of image recognition technology is becoming increasingly popular in various industries, including retail, healthcare, and security. One area where image recognition can be particularly useful is in counting objects in images. As such, the development of a mobile application that can accurately count objects in images has the potential to be a game-changer in several fields. However, to ensure that the application is effective and user-friendly, several key considerations need to be taken into account during its development, such as accuracy, speed, and ease of use. Therefore, further research and development are necessary to create a robust and reliable mobile application that can count objects in images with high accuracy and efficiency.

Small Objects Recognition

Improve the object recognition capabilities of your application to detect a wider range of objects. This can be achieved by training the application on larger and more diverse datasets.

Real-time Object Counting

Implement real-time object counting functionality so that users can track the number of objects as they move in front of the camera.

User Feedback and Data Collection

Collect user feedback to improve the application's user interface and functionality. Additionally, consider collecting data on the types of objects that users are counting to improve the application's recognition capabilities.

Machine Learning and Artificial Intelligence

Incorporation of ML and AI algorithms to improve the application accuracy and performance over time, this can help the application to learn from user interactions and improve its object recognition and counting capabilities.

Object Tracking

Implement object tracking functionality to enable users to track the movement of objects and count them accurately. This can be useful in scenarios where objects are moving rapidly or in a crowded environment.

Multiple Objects Counting

Allow users to count multiple objects of the same type simultaneously. This can be useful in scenarios where there are multiple objects of interest in the frame.

Offline Capabilities

Add offline capabilities to your application so that users can continue to use it even when they don't have an internet connection. This can be achieved by storing object recognition models and other relevant data locally on the user's device.

Integration with Other Applications

Consider integrating your application with other applications and services to provide users with a more comprehensive solution. For example, you could integrate your application with a navigation application to help users locate the objects they are counting more easily.

Appendix A

Assumption: A proposition that is accepted as true or taken for granted without sufficient evidence or conclusive proof.

Availability: Present and ready for use; at hand; accessible.

API: Specifies how some software components should interact with each other, in addition to accessing database or computer hardware, such as hard disk drives or video cards.

Class: A category or group of objects that share common characteristics and behaviors, often representing real-world entities (such as people, places, or things) within a particular business or problem domain.

Dependency: Dependency is the reliance of a module on some other module, event, factor or a group outside of its control.

Feature: A feature is a quality attribute of an application which satisfies a specific requirement. It provides a specific capability to a user and enables the fulfillment of a business objective.

Flexibility: Ability to adapt to possible or future changes as per any requirements by the users or stake holder.

Functional Requirement: A functional requirement is a specific function or behavior that a system must be able to perform or exhibit under certain conditions or situations.

Hardware: Any physical equipment or a component of a computing device and the associated physical equipment or a computing device as a whole directly involved in the performing data-processing.

Hardware Interface: The physical characteristics of each interface between the hardware components and the software product of the system.

Implementation: It is the execution of an idea, model, design, plan, standard, specification, algorithm or policy.

Interface: A point where two systems, subjects, organizations, etc., meet and interact.

Modifiability: In the context of software development, refers to the ease or difficulty with which changes can be made to the software.

Non-functional requirement: Non-functional requirements refer to the constraints, properties, or characteristics that a software system must exhibit or comply with, beyond its observable behavior.

Operating Environment: The circumstances, surrounding and potentially affecting something that is operating.

Operating System: It is defined as a software system that manages hardware resources of a computer and provides various essential services and interfaces to enable the execution of computer programs.

Perspective: The way in which objects appear to the eye.

Process: A process is a defined set of activities or tasks that are performed in a specific sequence to achieve a specific goal. A process description is a documented definition of these activities, including their inputs, outputs, and associated roles and responsibilities.

Portability: Portability refers to the ease with which software can be moved or transferred from one platform or environment to another, with minimal modifications or changes required.

References: It is a comprehensive list of all the documents or websites' links referred to in this SRS. These sources include system requirements specifications, code, use cases, or a vision and scope document.

Response: A reaction, as that of an organism or a mechanism, to a specific stimulus.

Requirement: A requirement describes customer's needs or objectives, or feature that a product must possess in order to meet an objective. It is a characteristic or property that the product must possess to deliver value to its stakeholders.

Reliability: Dependable, how often the software fails.

Scope: The scope draws the boundary between what lies in and what lies out for the project. It is the description of facility or task that current project will do and which requirement and needs it will address.

Software requirements specification: It is the sum and detailed description of functional and non-functional requirements that a software or application will fulfill.

Specification: The process of documenting a system's requirements in a structured, manageable and shareable form.

System requirement: It is the top-level requirement for a product. System requirement contains several subsystems, which could be all-software or software and hardware.

Usability: Fit for use; convenient to use.

Use case: A use case is a detailed description of an interaction or scenario between a user ("actor") and a system that leads to a valuable outcome for the user. It typically includes a sequence of steps, including the various actions taken by the user and the system's responses to those actions. A use case may also describe any alternative paths or exceptions that could occur during the interaction. The ultimate goal of a use case is to provide an understanding of how the system should act and respond to make sure the system fulfills the user's requirements and expectations from the user's point of view.

Use case diagram: It is the graphical representation of the system's use cases and actors involved in them. It provides a high-level view of the system's functionality and the various actors that interact with it to achieve their goals. Use case diagrams typically depict the relationships between the actors and the use cases they perform, helping to visualize how the system supports the actors in achieving their goals. In essence, a use case diagram provides a concise and comprehensive summary of the system's use cases and actors, serving as a useful tool for communication and collaboration between stakeholders involved in the system's design and development.

User: A user refers to an individual or entity that interacts with a system, software, or service. This interaction can be direct or indirect and may include the use of system outputs, but not necessarily the generation of those outputs. The term user encompasses a wide range of individuals, including customers, clients, employees, administrators, and other stakeholders who interact with the system in some way.

User class: A classification of users in a system based on their shared characteristics and requirements for the system.

User Interface: The user interface refers to the visual and interactive elements of a software product or system through which users interact with it. It includes all the screens, buttons, menus, forms, and other components that users interact with to achieve their tasks. The user interface design encompasses the layout, organization, and presentation of these elements to provide users with a seamless and intuitive experience. It also involves the design of the interactions between the user and the system, including input validation, error handling, and feedback mechanisms. In essence, the user interface is the means by which users can interact with the software product, and its design plays a crucial role in determining the overall user experience.

User requirement: A user requirement is a statement that describes a specific need, goal, or expectation that a user has for a system or software product. User requirements are typically gathered through user interviews, surveys, or other forms of user feedback and are essential in guiding the design and development of a system that meets the needs of its users. User requirements are often documented in a user requirements document (URD) and serve as a basis for creating the system's functional and non-functional requirements.

Usability: It defines level of difficulty to learn and operate a system.

Validation: The process of evaluating a deliverable to see if it adheres to the customer's requirements and expectations.

Verification: The process of reviewing a work product to confirm that it adheres to the predefined requirements and criteria established during the development phase when it was produced.

Wi-Fi: It is a widely used wireless networking technology that utilizes radio waves to enable high-speed wireless internet and network connections. It allows devices to connect to the internet or to each other without the need for physical cables, providing greater flexibility and convenience.

Sequence Diagram: It is a type of interaction diagram that shows how objects communicate with each other in a particular scenario of a system or software application. It illustrates the sequence of actions that occur between objects, including the sequence in which messages are sent and received, the lifelines of objects, and the time at which the messages are exchanged.

Glossary

APP: Application

ML: Machine Learning

OS: Operating System

IDE: Integrated Development Environment

API: Application Programming Interface

UML: Unified Modeling Language

PC: Personal Computer

YOLO: You Only Look Once

PIL: You Only Look Once

CNN: Python Imaging Library

R-CNN: Region-Based Convolutional Neural Network

VGG: Visual Geometry Group

SSD: Single Shot Detector

URD: User Requirements Document

Bibliography

https://blog.roboflow.com/how-to-count-objects-in-an-image-using-python/

https://dev.to/stokry/how-to-count-objects-on-an-image-with-python-142h

https://github.com/DoganK01/YOLOV7-OBJECT-COUNTER-V1.2

https://blog.roboflow.com/retail-store-item-detection-using-yolov5/

https://github.com/googlesamples/mlkit/tree/master/android/vision-quickstart

https://developers.google.com/ml-kit/vision/object-detection/android

https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06

https://www.v7labs.com/blog/yolo-object-detection