**JSS MAHAVIDYAPEETHA**

**JSS SCIENCE AND TECHNOLOGY UNIVERSITY**

JSS Technical Institutions Campus, Mysuru – 570006



Mini project report submitted in partial fulfilment of curriculum prescribed for the Digital Image Processing (CS662) course for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

**ASSET VISIONER AND AUDITING SYSTEM**

by

Chethana. S Naveen .J Nimish.L

01JST16CS024 01JST16CS057 01JST16CS063

Harshitha M.L Srihari Charith Arbaaz Khan

01JST16CS127 01JST16CS105 01JST16CS014

Under the guidance of

SHRUTHI.N.M

Assistant Professor

Dept. of CS & E, JSS STU Mysore

**ABSTRACT:**

One of the key investments in every organization are the assets. Systems for handling data about assets are valuable investment as well. Being able to have an overview of the assets, information about every detail regarding them and more over being able to analyse that data can be critical for the business. The subject of this project is to show the importance of the asset management strategy, point the key role that an asset management system has and list out the desired features. Another important aspect of the project is to show that Software as a Service (SaaS) business model is best suitable model, especially for small and medium companies. This project tries to define a methodology for evaluating asset management systems. Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance.

**ACKNOWLEDGMENT:**

We render our heartfelt gratitude to Assistant Professor Ms. Shruthi. N.M for continuously assisting us in the development of this project and giving valuable insight about the technological aspects. We are grateful for giving us this golden opportunity to do this wonderful project on this ever-growing topic which also helped us in doing a lot of research and we came to know about so many new things.

Contents

[INTRODUCTION: 1](#_Toc7039040)

[LITERATURE SURVEY : 2](#_Toc7039041)

[SYSTEM REQUIREMENTS: - 3](#_Toc7039042)

[EXISTING SYSTEM: 3](#_Toc7039043)

[PROPOSED SYSTEM: 4](#_Toc7039044)

[IMPLEMENTATION: 5](#_Toc7039045)

[**PYTHON CODE** 5](#_Toc7039046)

[LIBRARIES USED: 7](#_Toc7039047)

[**TensorFlow Object Detection** 7](#_Toc7039048)

[**OpenCV:** 7](#_Toc7039049)

[**Pillow:** 8](#_Toc7039050)

[**Keras :** 8](#_Toc7039051)

[**ImageAI :** 9](#_Toc7039052)

[**RetinaNet :** 9](#_Toc7039053)

[ALGORITHM USED : 10](#_Toc7039054)

[**1.Nearest-neighbor interpolation** 10](#_Toc7039055)

[**2.Whittaker–Shannon interpolation formula** 10](#_Toc7039056)

[**Equivalent formulation: Convolution / Lowpass filter** 11](#_Toc7039057)

[**3.Grayscale to RGB** 11](#_Toc7039058)

[RESULT 12](#_Toc7039059)

[**INTERFACE OF THE DOMAIN** 13](#_Toc7039060)

[CONCLUSION: 15](#_Toc7039061)

[FUTURE IN THIS DOMAIN : 16](#_Toc7039062)

[REFERENCES : 17](#_Toc7039063)

# INTRODUCTION:

In any organization, company asset management is a big and tedious task. Usually they do this manually. Recently, High-Definition video cameras have become more accessible and are being used for data capture in asset management systems. These types of camera provide more pixel data regarding assets appearing further away and it would be useful if we could take advantage of this extra data. In this project, we are interested in the asset detection stage of asset management systems. This asset detection can either be performed manually by users, or automatically, which increases the whole system’s efficiency. Object recognition methods are an obvious choice for the automation of asset detection. Since, these methods can be trained to perform detection of assets in the captured data. One application of being able to find assets at all distances (including far distances) is the determination of sighting distances. That is, organizations can determine how far away their assets can be seen. This is especially important. But, to perform this type of asset detection, methods are required which perform distance-invariant asset detection.

Asset management systems allow organizations to efficiently store data pertaining to the physical location of important assets. Asset detection is a key component of such systems, the automation of which greatly increases efficiency and for which object recognition techniques are an obvious choice. Hence, this project presents a literature survey covering the different methods applicable to distance-invariant object recognition.

Good news on this topic is the increasing number of various size organizations that became aware about the value of the comprehensive system implementation for asset management that will produce complete, precise and dynamic view over their assets but also it will produce indicators for most efficient further development.

With the benefits of contemporary technology, the organizations have opportunity to organize the assets they own, notice their unnecessary expenses and make most efficient strategy to utilize their current and future investment. Having in mind all of these benefits we put as a strategic goal the implementation of asset management system.

# LITERATURE SURVEY :

* ImageNet Classification with Deep ConvolutionalNeural Networks

<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

* Object Detection

<https://www.academia.edu/38702601/Object_Detection>

* Detection of median filtering in digital images

<https://www.engpaper.com/on-detection-of-median-filtering-in-digital-images.htm>

# SYSTEM REQUIREMENTS: -

* Hardware requirements:
* Machine with 4GB RAM
* Camera with minimum 5MP resolution
* GPU with minimum GTX 580 3GB
* Processor Intel i3 5th gen
* Software requirements:
* TensorFlow environment
* Keras and YOLO
* AWS cloud integrated with PostgreSQL
* Visual Studio 2010
* Anaconda Navigator

# EXISTING SYSTEM:

In any organization, the asset management is usually a tough task to manage. Usually assets are maintained manually, checking of the assets are also done manually. It involves trained people for performing the checking of assets. In general, a method’s performance decreased with increasing asset distance. There is no accuracy in this system. This system is manual and takes lot of time. Sometimes the asset might have been stolen, but it might not come into picture until asset is checked manually. Hence, there is a delay in noticing the missing/ stolen asset. There is no effective or accuracy in this system.

Object recognition methods are an obvious choice for the automation of asset detection. Since, these methods can be trained to perform detection of assets in the captured data. One application of being able to find assets at all distances (including far distances) is the determination of sighting distances

# PROPOSED SYSTEM:

Asset management systems provide the tools to perform the data collection, storage and presentation. The imaging sensor and positioning equipment data is synchronized and stored in a database. A post-processing asset detection step allows the detection of assets. This project has presented a survey of object recognition methods, applicable to the task of automatic asset detection in asset management systems, where assets may appear at any distance. This system is effective and accurate. It reduces manual stress, time and is cost effective too.

Features of proposed system: -

1. Scale-Invariant Object Recognition Objects which are different distances away appear as different sizes (or scales) in captured images.

2. Distance-Invariant Object Recognition We make the distinction between scale-invariant and

distance-invariance as; distance-invariant methods utilize specific information regarding the expected distance of an object, whereas, scale-invariant methods are only concerned with size.

3.Active Vision: The previous methods which have been described are all applicable to fixed camera systems. However, active vision methods employ non-fixed cameras systems to enable object recognition of both near and far objects.

# IMPLEMENTATION:

## **PYTHON CODE**

from imageai.Detection import ObjectDetection

import os

execution\_path = os.getcwd()

detector = ObjectDetection()

detector.setModelTypeAsRetinaNet()

detector.setModelPath( os.path.join(execution\_path , "resnet50\_coco\_best\_v2.0.1.h5"))

detector.loadModel()

detections = detector.detectObjectsFromImage(input\_image=os.path.join(execution\_path , "image.jpg"), output\_image\_path=os.path.join(execution\_path , "imagenew.jpg"))

for eachObject in detections:

print(eachObject["name"] , " : " , eachObject["percentage\_probability"] )

**Explaining how the code works**



In the above 3 lines, we imported the **ImageAI**object detection class in the first line, imported the python **os**class in the second line and defined a variable to hold the path to the folder where our python file, RetinaNet model file and images are in the third line.



In the 5 lines of code above, we defined our object detection class in the first line, set the model type to RetinaNet in the second line, set the model path to the path of our RetinaNet model in the third line, load the model into the object detection class in the fourth line, then we called the detection function and parsed in the input image path and the output image path in the fifth line.



In the above 2 lines of code, we iterate over all the results returned by the **detector.detectObjectsFromImage**function in the first line, then print out the name and percentage probability of the model on each object detected in the image in the second line.

# LIBRARIES USED:

## **TensorFlow Object Detection**

TensorFlow’s Object Detection API is a very powerful tool that can quickly enable anyone (especially those with no real machine learning background like myself) to build and deploy powerful image recognition software.

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). This architecture gives flexibility to the application developer: whereas in previous “parameter server” designs the management of shared state is built into the system, TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural networks. Several Google services use TensorFlow in production, we have released it as an open-source project, and it has become widely used for machine learning research. In this paper, we describe the TensorFlow dataflow model and demonstrate the compelling performance that Tensor- Flow achieves for several real-world applications.

## **OpenCV:**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding. The library is used extensively in companies, research groups and by governmental bodies.

## **Pillow:**

Pillow is a fork of the Python Imaging Library (PIL). PIL is a library that offers several standard procedures for manipulating images. It's a powerful library, but hasn't been updated since 2011 and doesn't support Python 3. Pillow builds on this, adding more features and support for Python 3. It supports a range of image file formats such as PNG, JPEG, PPM, GIF, TIFF and BMP. We'll see how to perform various operations on images such as cropping, resizing, adding text to images, rotating, grayscaling, etc. using this library.

## **Keras :**

**Keras** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [neural-network](https://en.wikipedia.org/wiki/Artificial_neural_network) library written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). Designed to enable fast experimentation with [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning), it focuses on being user-friendly, modular, and extensible.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, [objectives](https://en.wikipedia.org/wiki/Objective_function), [activation functions](https://en.wikipedia.org/wiki/Activation_function), [optimizers](https://en.wikipedia.org/wiki/Mathematical_optimization), and a host of tools to make working with image and text data easier. The code is hosted on [GitHub](https://en.wikipedia.org/wiki/GitHub), and community support forums include the GitHub issues page, and a [Slack](https://en.wikipedia.org/wiki/Slack_(software)) channel.

In addition to standard neural networks, Keras has support for [convolutional](https://en.wikipedia.org/wiki/Convolutional_neural_networks) and [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks). It supports other common utility layers like dropout, batch normalization, and pooling

## **ImageAI :**

**ImageAI** is a python library built to empower developers, reseachers and students to build applications and systems with self-contained Deep Learning and Computer Vision capabilities using simple and few lines of code.

ImageAI provided very powerful yet easy to use classes and functions to perform Image Object Detection and Extraction. ImageAI allows you to perform all of these with state-of-the-art deep learning algorithms like RetinaNet, YOLOv3 and TinyYOLOv3. With ImageAI you can run detection tasks and analyse images.

## **RetinaNet :**

Retina Net is a composite network composed of:

* a backbone network called Feature Pyramid Net, which is built on top of ResNet and is responsible for computing convolutional feature maps of an entire image.
* a subnetwork responsible for performing object classification using the backbone’s output.
* a subnetwork responsible for performing bounding box regression using the backbone’s output.

# ALGORITHM USED :

## **1.Nearest-neighbor interpolation**

**To implement resizing**

**Nearest-neighbor interpolation** (also known as **proximal interpolation** or, in some contexts, **point sampling**) is a simple method of multivariate interpolation in one or more dimensions.

Interpolation is the problem of approximating the value of a function for a non-given point in some space when given the value of that function in points around (neighboring) that point. The nearest neighbor algorithm selects the value of the nearest point and does not consider the values of neighboring points at all, yielding a piecewise-constant interpolant. The algorithm is very simple to implement and is commonly used (usually along with mipmapping) in real-time 3D rendering to select color values for a textured surface.

## **2.Whittaker–Shannon interpolation formula**

**To implement cropping**

The **Whittaker–Shannon interpolation formula** or **sin c interpolation** is a method to construct a continuous-time bandlimited function from a sequence of real numbers.

Given a sequence of real numbers, *x*[*n*], the continuous function

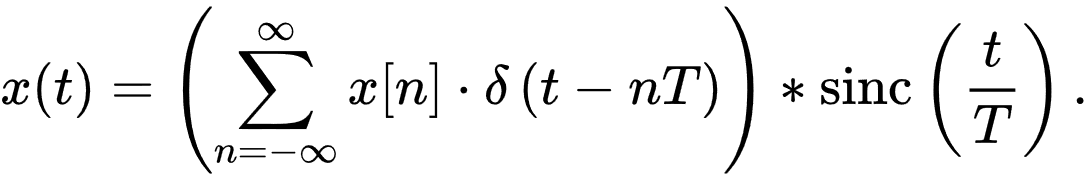
{\displaystyle x(t)=\sum \_{n=-\infty }^{\infty }x[n]\,{\rm {sinc}}\left({\frac {t-nT}{T}}\right)\,}

(where "sinc" denotes the [normalized sinc function](https://en.wikipedia.org/wiki/Normalized_sinc_function))

has a [Fourier transform](https://en.wikipedia.org/wiki/Fourier_transform), *X*(*f*), whose non-zero values are confined to the region |*f*| ≤ 1/(2*T*).  When parameter *T* has units of seconds, the bandlimit, 1/(2*T*), has units of cycles/sec ([hertz](https://en.wikipedia.org/wiki/Hertz)). When the *x*[*n*] sequence represents time samples, at interval *T*, of a continuous function, the quantity *fs* = 1/*T* is known as the [sample rate](https://en.wikipedia.org/wiki/Sample_rate), and *fs*/2 is the corresponding [Nyquist frequency](https://en.wikipedia.org/wiki/Nyquist_frequency). When the sampled function has a bandlimit, *B*, less than the Nyquist frequency, *x*(*t*) is a perfect reconstruction of the original function.

### **Equivalent formulation: Convolution / Lowpass filter**

The interpolation formula is derived in the Nyquist–Shannon sampling theorem article, which points out that it can also be expressed as the convolution of an infinite impulse train with a sinc function:

{\displaystyle x(t)=\left(\sum \_{n=-\infty }^{\infty }x[n]\cdot \delta \left(t-nT\right)\right)\*{\rm {sinc}}\left({\frac {t}{T}}\right).}

This is equivalent to filtering the impulse train with an ideal (*brick-wall*) low-pass filter.

## **3.Grayscale to RGB**

We can do this in three ways:

The **lightness** method averages the most prominent and least prominent colours: (max(R, G, B) + min(R, G, B)) / 2.

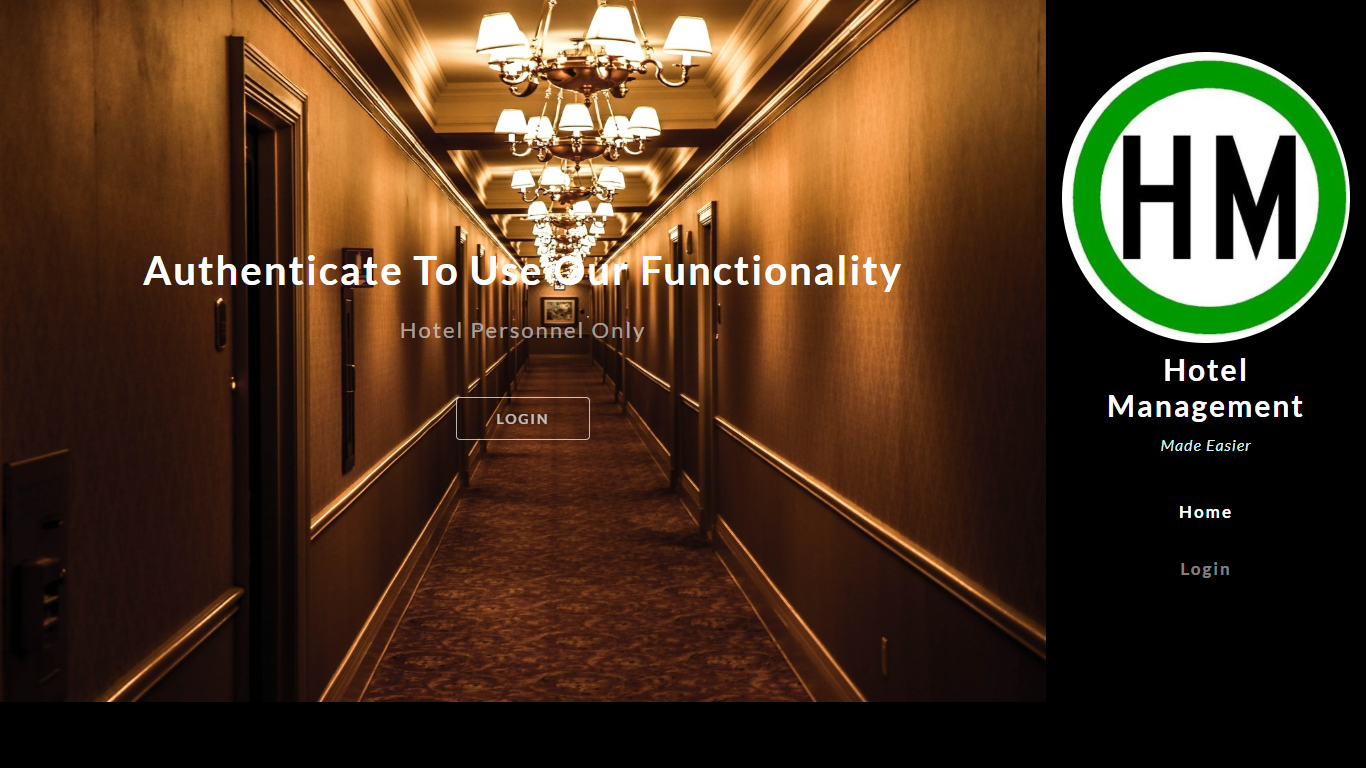
The **average** method simply averages the values: (R + G + B) / 3.

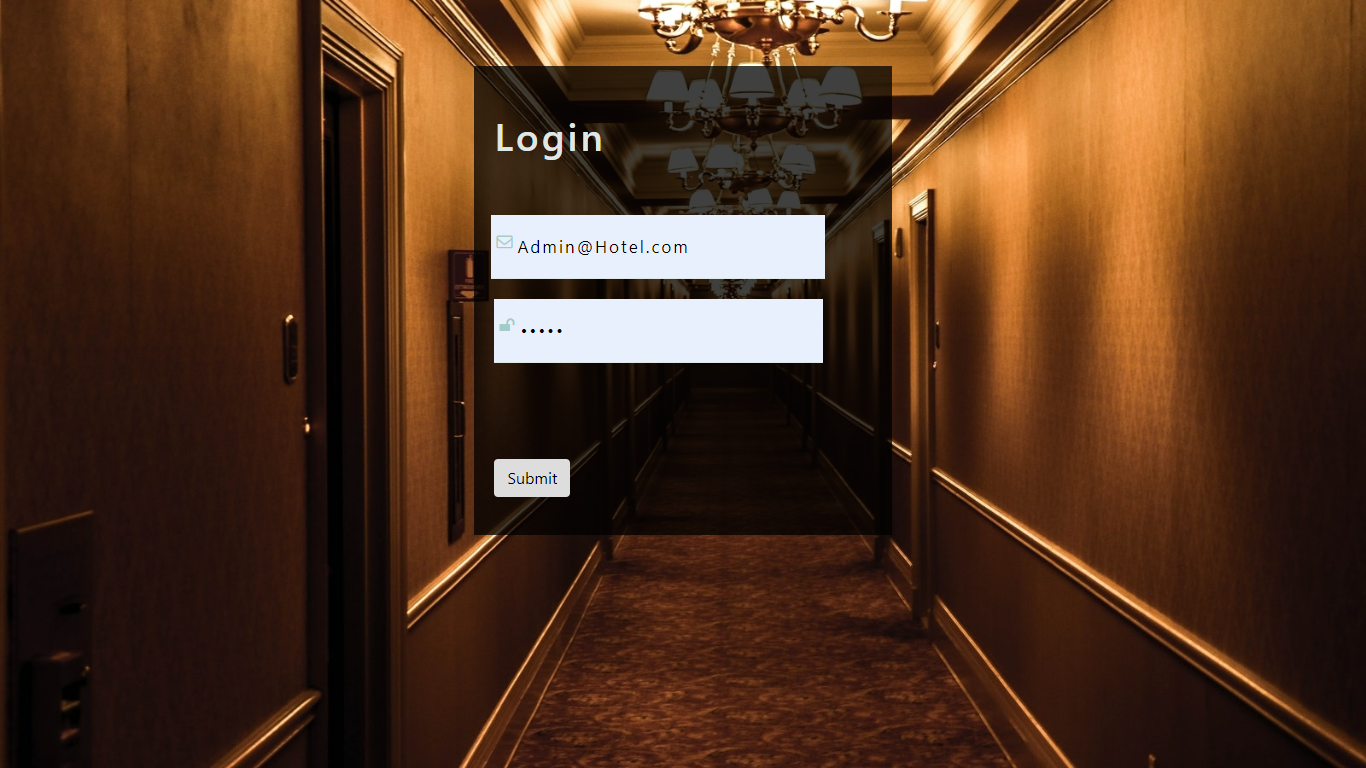
The **luminosity** method is a more sophisticated version of the average method. It also averages the values, but it forms a weighted average to account for human perception. We’re more sensitive to green than other colours, so green is weighted most heavily. The formula for luminosity is 0.21 R + 0.72 G + 0.07 B.

cv::cvtColor(gray, color, cv::COLOR\_GRAY2BGR)

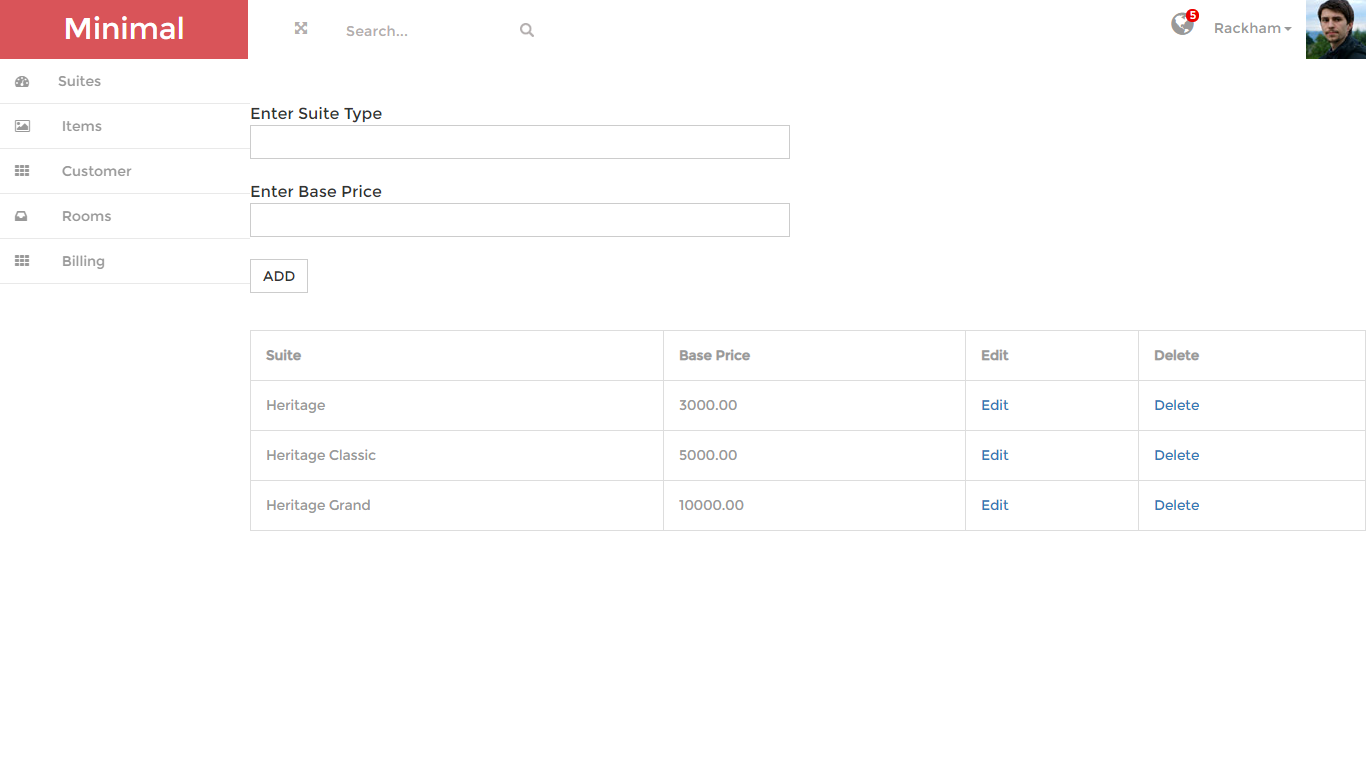
RESULT**:**

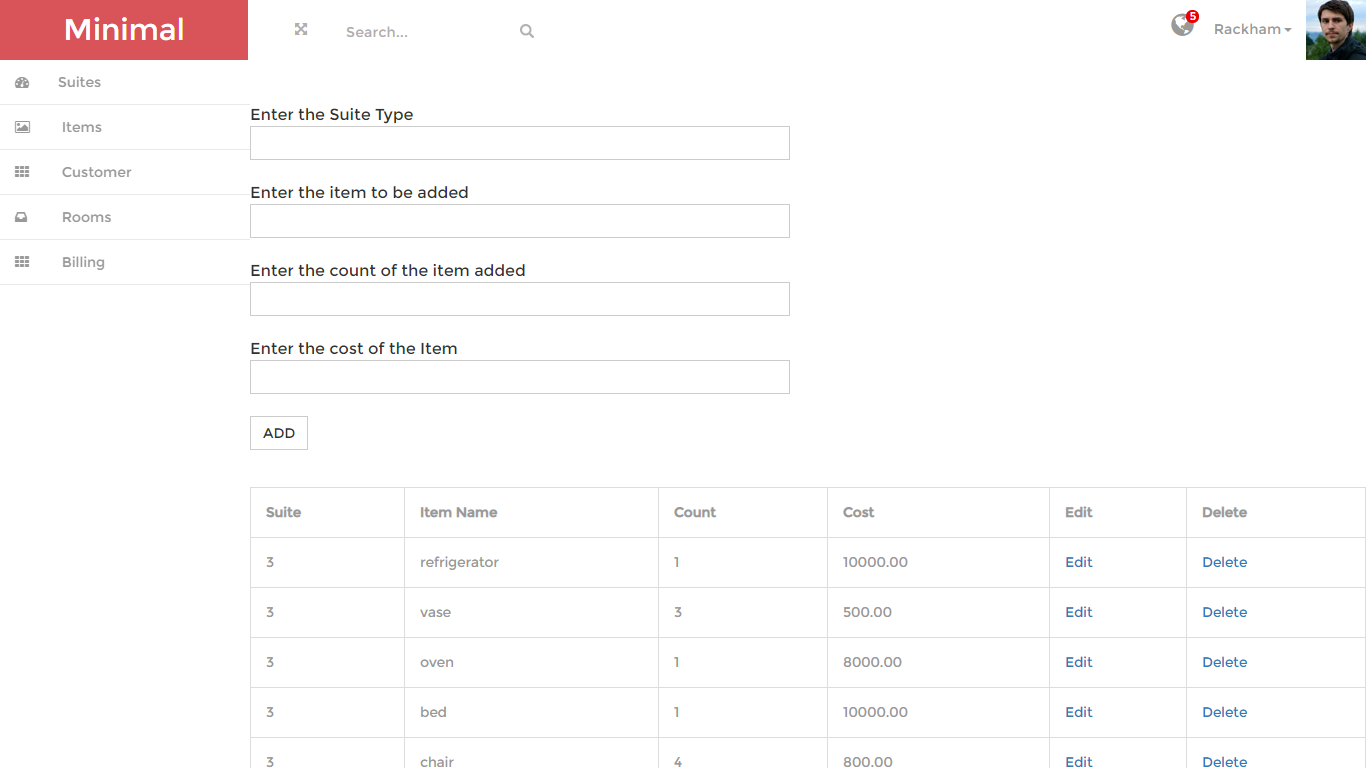
**LOGIN PAGE**

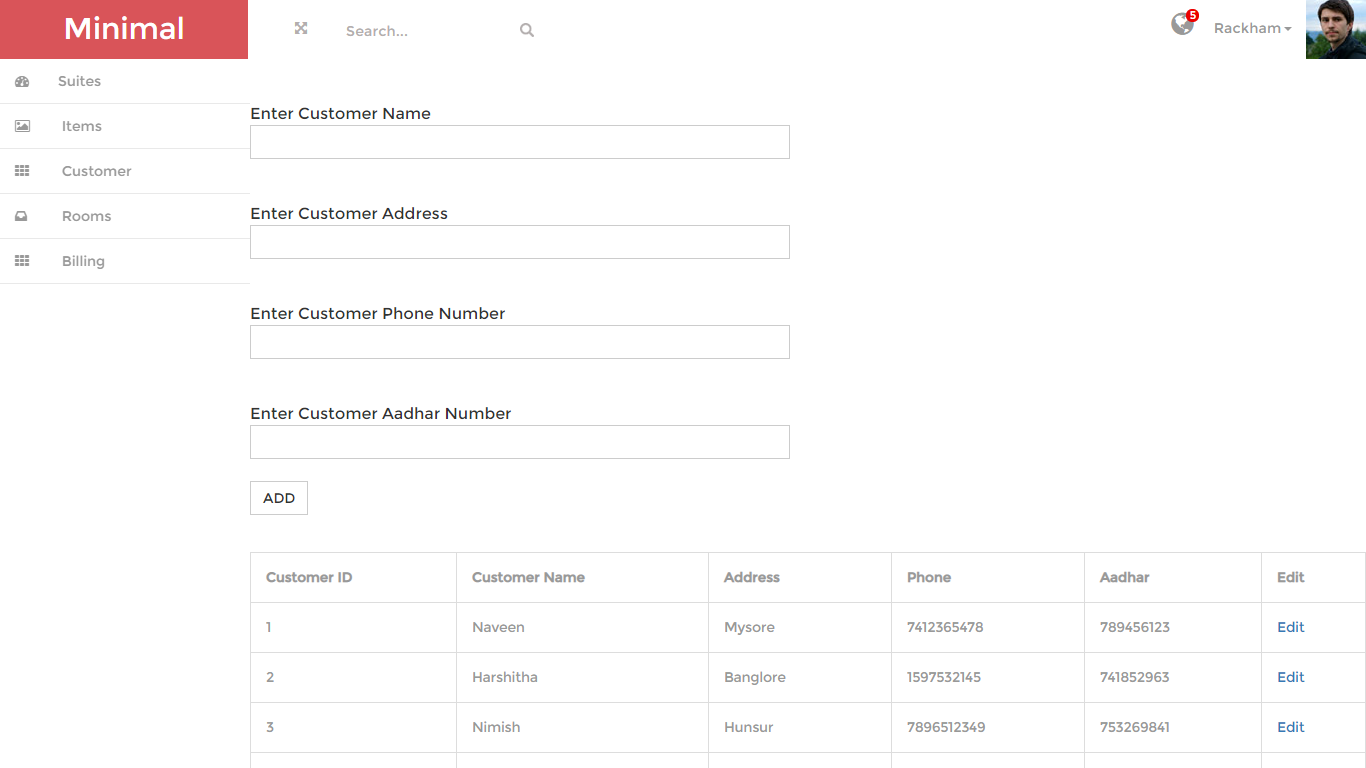


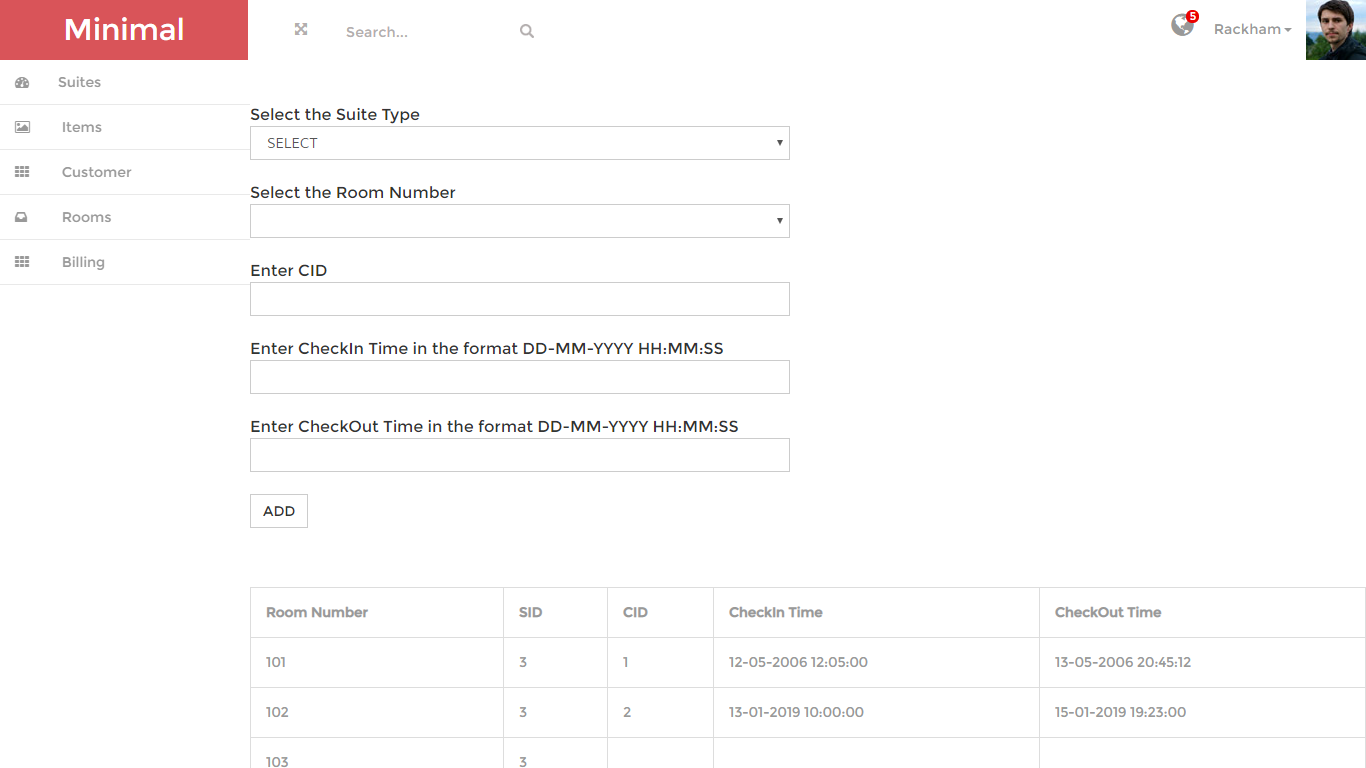


## **INTERFACE OF THE DOMAIN**









# CONCLUSION:

The image processing methodology used are RetinaNet and ImageAI which are very efficient image processing algorithms. Even though the input image is of lower resolution the algorithm is implemented in such a way that the given image is converted to a standard 256 X 256 resolution.

Object detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quad-copters, drones and soon service robots), the need of object detection systems is gaining more importance. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

# FUTURE IN THIS DOMAIN :

More or less, the **implementation of AI** is going to be the next big thing in the field of Image processing, and in fact the technology and tools are totally present to make use of Artificial Intelligence for the purpose of processing or analyzing images.

A hot topic is the use of Facial Recognition in Image Processing. A good display of this kind of technology is Facebook has algorithms that are used to detect faces and then ask you to tag the appropriate people in your pictures that you post.

In our project’s context the implementation of the Virtual concierge is the new big thing.

The virtual concierge is like a digital assistant for customers. Before they arrive at the property or during their stay, guests can access this digital service to see what the hotel offers, enjoy 360° video tours, find places to visit or events to attend nearby, learn about upgrade opportunities or extra packages to book. Unlike the website home page, this is the place for more targeted upsell promotions levelled with rich useful information that will be helpful for travellers and can improve the quality of their stay significantly. The best thing here is that the virtual concierge can be a section of the website connected to the hotel PMS software through the hotel app. This would allow for even more service personalisation and the creation of an unique digital gateway for two-way communication between the hotel and its guests.

# REFERENCES :

* Forrest N. et al, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size   
  <https://arxiv.org/abs/1602.07360>
* Kaiming H. et al, Deep Residual Learning for Image Recognition   
  <https://arxiv.org/abs/1512.03385>
* Bradski, Gary; Kaehler, Adrian (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc. p. 6.
* Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). ["Realtime Computer Vision with OpenCV"](http://dl.acm.org/citation.cfm?id=2206309). Queue. pp. 40:40–40:56. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1145/2181796.2206309](https://doi.org/10.1145/2181796.2206309).
* ["Details of package python-imaging in sid"](http://packages.debian.org/sid/python-imaging). packages.debian.org. [Software in the Public Interest](https://en.wikipedia.org/wiki/Software_in_the_Public_Interest). Retrieved December 8, 2013.
* Object Detection using machine learning [*https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606*](https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606)
* ["Keras backends"](https://keras.io/backend/). keras.io. Retrieved 2018-02-23.

https://keras.io/backend/

<https://www.superdatascience.com/blogs/the-ultimate-guide-to-artificial-neural-networks-ann>

·

·