# Lost and Found!

A web application that keeps track of missing belongings in a hotel. You lost something? Look here.

Nikhil Kumar Parashar
(19BCE0076)
Anmol Bansal
(19BCE0630)

# Problem Statement

Current Hotel Management ignores the Lost and Found management. Even if they have a lost and found section it is not managed well enough. So when you lose your belonging it gets tough to get them back.

This is where our web application comes into play. Every hotel can manage the lost item easily. All they have to do is upload the image of the retrieved lost item to their database. When the person returns looking for their belonging they just have to provide an image of the item and they can easily find the item.

# File Description

01

## templates

Stores the HTML files

1. index.html - the home page of our app where we upload the image of lost item and retrieve the top 5 images that match the description
2. upload.html - the upload page where we upload a new lost item found in the hotel by management

02

## static

**Stores all the data that remains static irrespective of the instance you are building**

1. feature - Stores the deep features of each image in a form of NumPy array
2. image - image database of all the lost belongings
3. uploaded - stores the query image once uploaded
4. bg.png - background of the website
5. style.css - styling of the website

# File Description

03

DeepFeatures.py

Where we call and instantiate our VGG16 model that helps in Deep Feature synthesis

04

SaveFeatures.py

Where we iterate through our database and save all the deep features extracted

05

Server.py

Where we retrieve top 5 lost objects and display on the web.

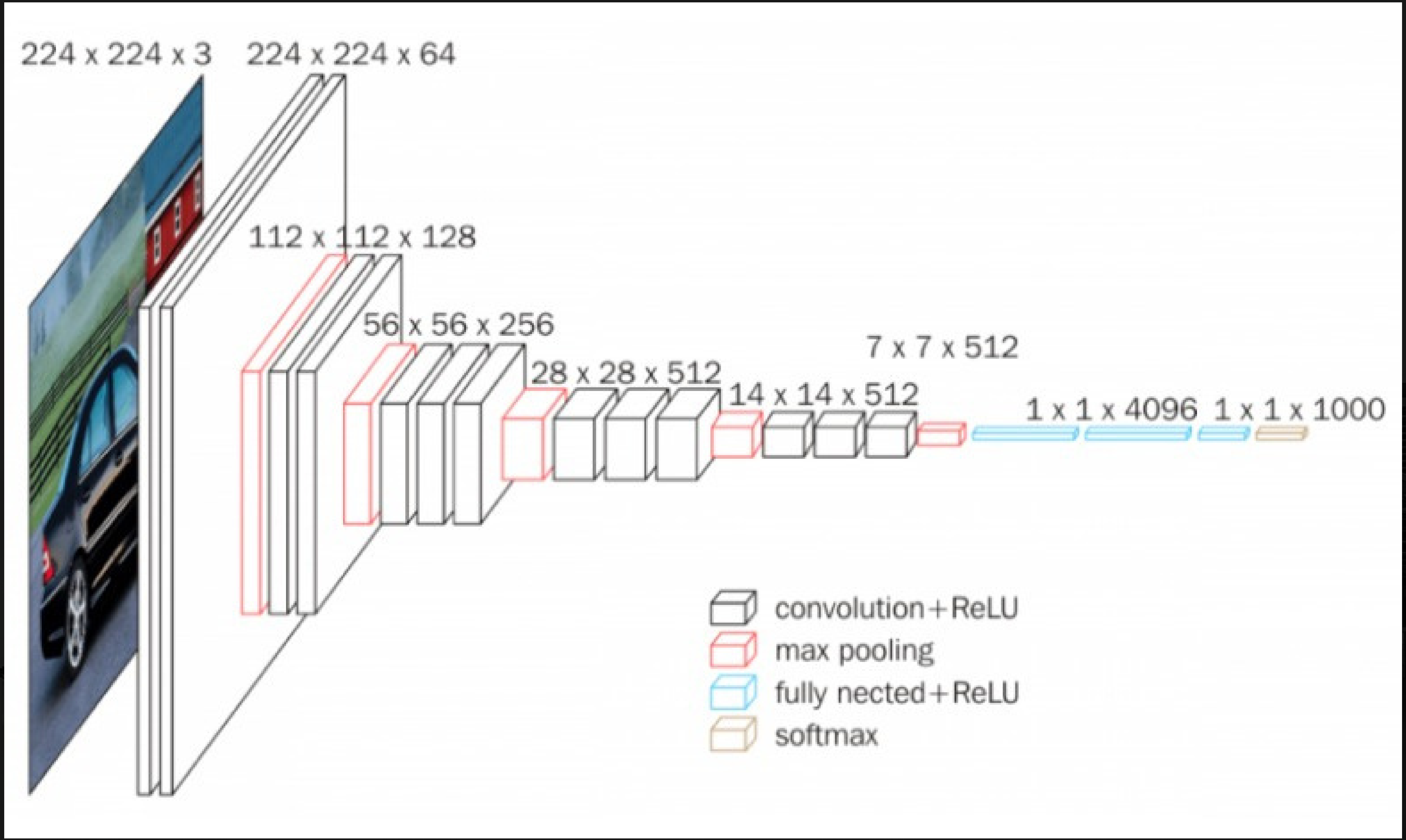
# Model Used

VGG-16

We made use of VGG16 model provided by keras. VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it.

VGG-16 is a convolutional neural network that 16 layers deep. The model loads a set of weights pre-trained on ImageNet. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.
The default input size for VGG16 model is 224 x 224 pixels with 3 channels for RGB image. It has convolution layers of 3x3 filter with a stride 1 and maxpool layer of 2x2 filter of stride 2.

# VGG16 Architecture



# Deep Feature Synthesis

Since images are Unstructured Data we cannot directly train our model over the images. For that we need to convert them to structure data and extract the features. Hence we convert images into numpy arrays and perform Deep Feature Synthesis and save them into another numpy array, which is later used for retrieval tasks.

Deep feature synthesis stacks multiple transformation and aggregation operations to create features from data spread across many tables. Like most ideas in machine learning, it’s a complex method built on a foundation of simple concepts.

# Modules Used

### Flask

an API of Python that allows us to build up web-applications

### Pillow

a Python Imaging Library (PIL)

### Keras and Tensorflow

most used libraries used in development of Deep Learning models

### Matplotlib

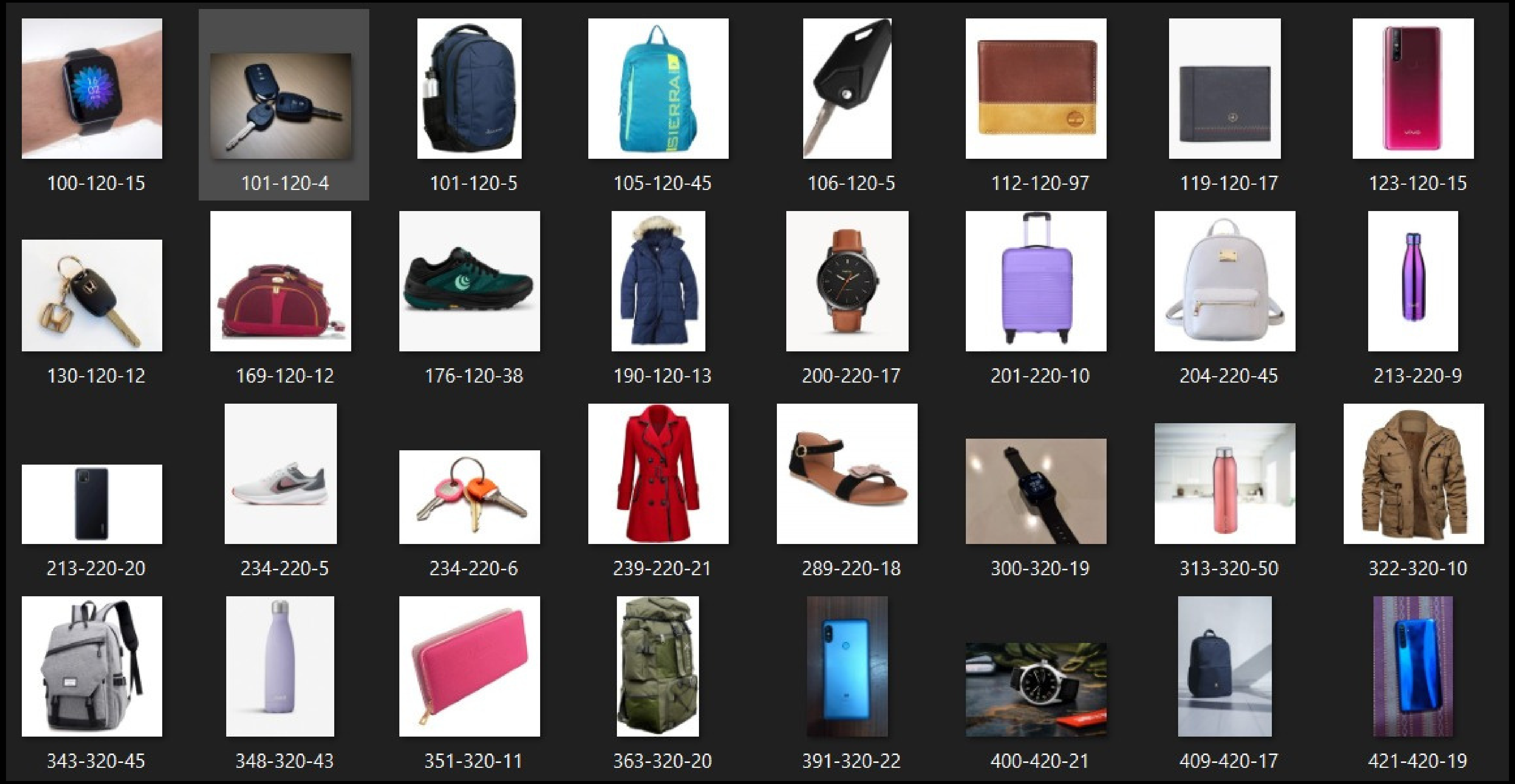
cross-platform, data visualization and graphical plotting library for Python

### Numpy

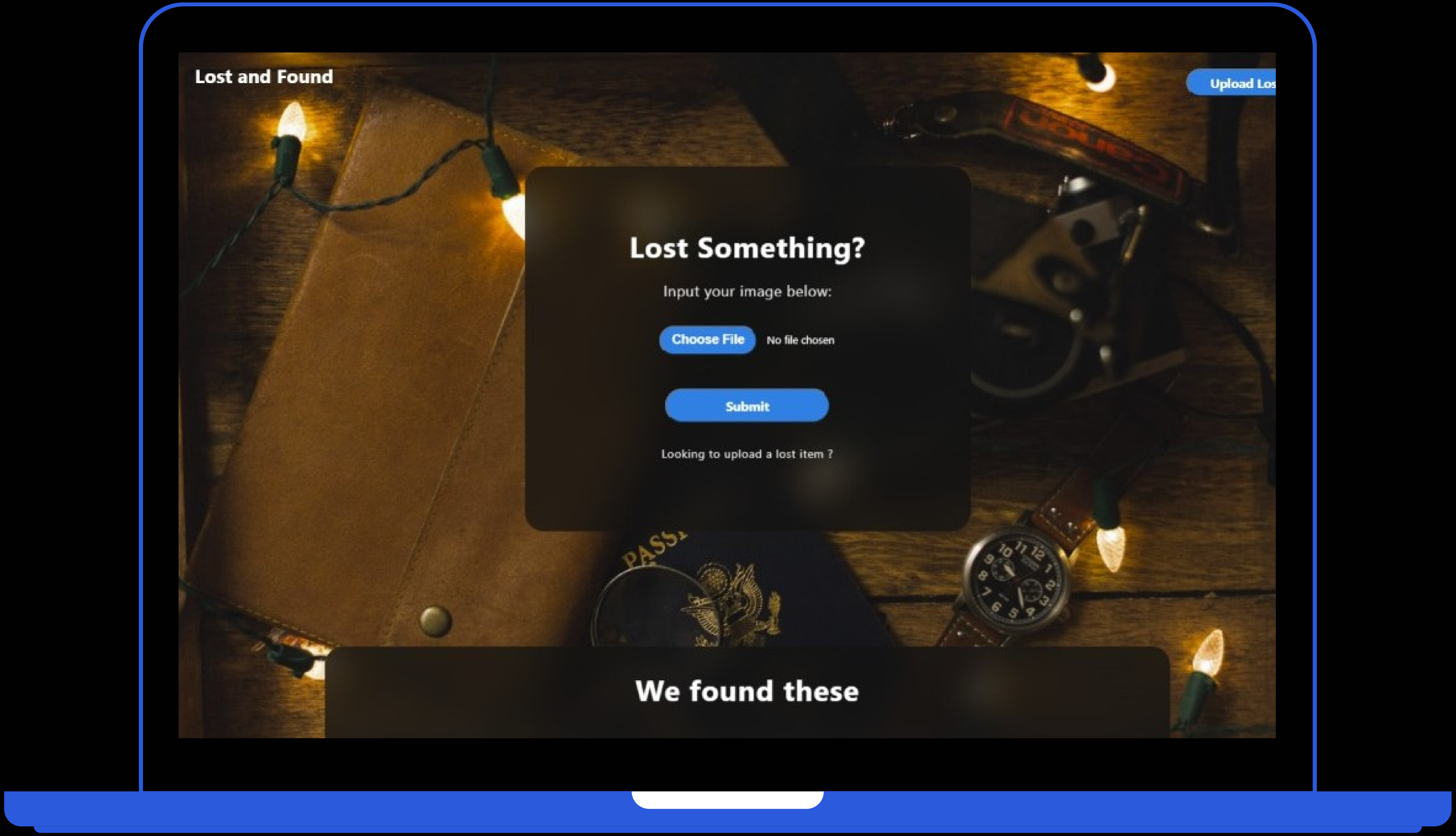
Python library that provides a simple yet powerful data structure: the n-dimensional array

# Our Database

The image are stored under a name, which is in the format roomno-storeroomno-rackno which helps us in finding the corresponding item easily.



# Demo



Lost and Found

## Lost Something?

Input your image below:

Looking to upload a lost item ?

## We found these

# Verification of the Retriever

To verify if the item is returned to the right user, the hotel will request the ID that the person used to check in to the room. Then the hotel, from their database can retrieve their room number that can later be used to cross check if the item claimed was found from the corresponding room, proving if the item belongs to them or not.

# What's Happening Next

1.) The database of the hotel and web app could be linked to make the verification easier.

2.) Image extension for now is limited to only jpg files, but can be extended to make this more dynamic.

3.) Database can be made less redundant by keeping track of time since the lost object has been kept stored, so as to remove if no one comes looking for it within a given period of time. This will also help in freeing up spaces in store room.

4.) Move this application from website to a desktop app, to make the access easier.