Automated Inspection Report

1. Objective:

To automate room cleanliness inspection reports using computer vision to classify room images as clean or messy and generate detailed, human-readable reports, enhancing efficiency and accuracy in facility management.

2. Technologies Used:

1) Machine Learning & Deep Learning

- **TensorFlow & Keras:** Utilized for loading the pre-trained deep learning model (model1.h5) and making predictions.
- **VGG16 Preprocessing:** Implements the preprocess_input function to prepare images for model prediction, optimized for **VGG16** architecture.

2) Data Processing & Utilities

- **NumPy**: Supports numerical computations, particularly for handling image arrays and data manipulation.
- **Image Processing**: OpenCV, Matplotlib, Seaborn – Used for image manipulation, visualization, and analysis.
- **Data Augmentation**: ImageDataGenerator – Facilitates data augmentation techniques to improve model generalization.

3) Report Generation

- **ReportLab:** A Python library used to generate professional PDF reports summarizing inspection results.
- **ImageReader (from ReportLab):** Embeds the uploaded room image into the PDF report for visual reference.

4) Frontend Technologies

- **HTML:** Used for structuring the web pages (index.html, result.html).

5) Backend Technologies

- **Flask (Python Framework):** Handles routing, form submissions, file uploads, and rendering web pages (app.py).
- **Jinja2 Templating Engine:** Dynamically renders data within HTML templates (e.g., {{ prediction }} in result.html).

6) **Deployment & Configuration**

- **Flask Development Server:** Runs the application locally with debugging enabled for development purposes (app.run(debug=True)).
- **Directory Structure:** Maintains organized folders for efficient file management and application scalability.

3. Flow Diagram:

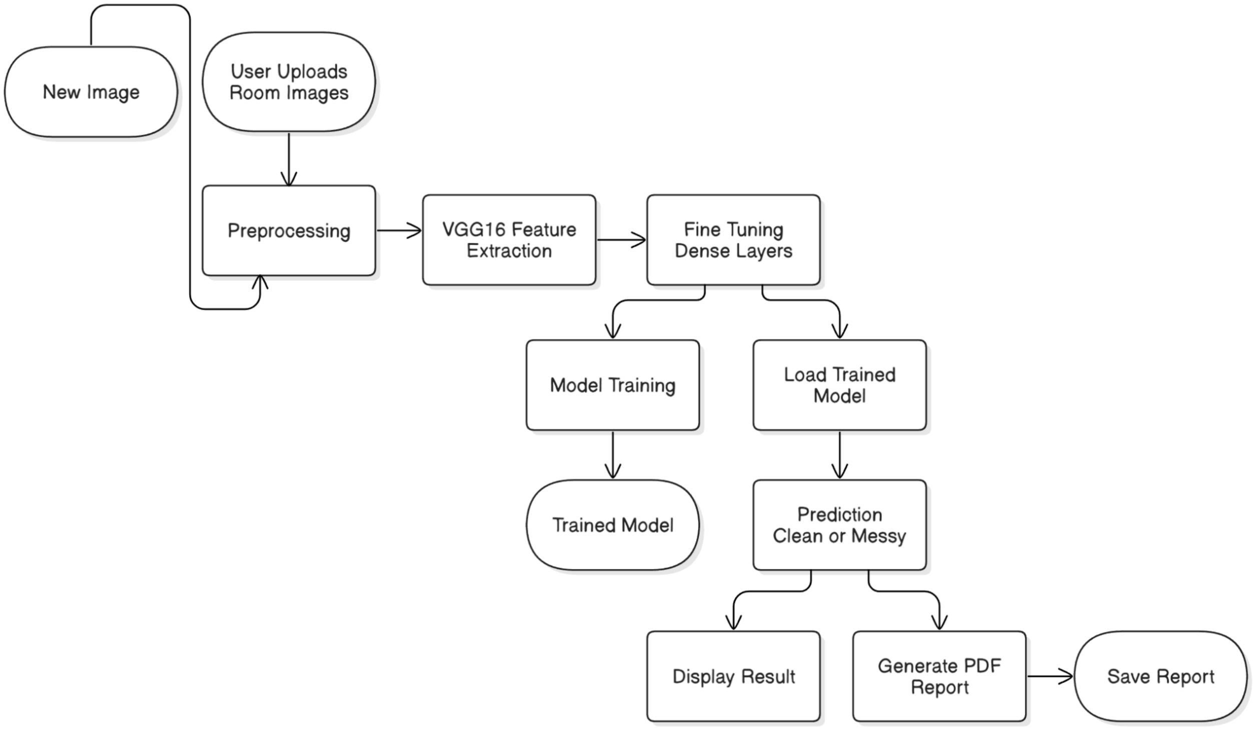
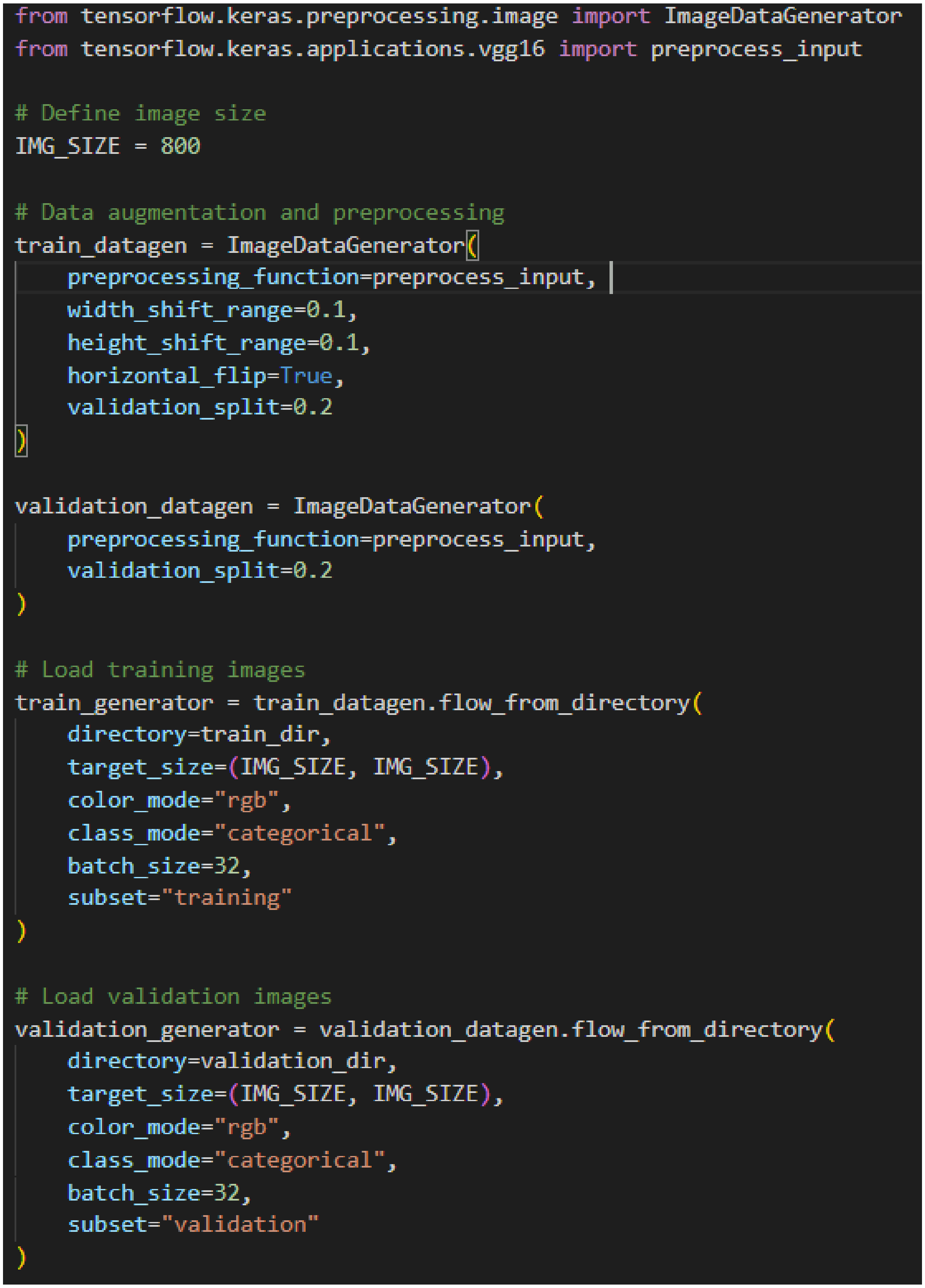


Fig.1 Model Architecture

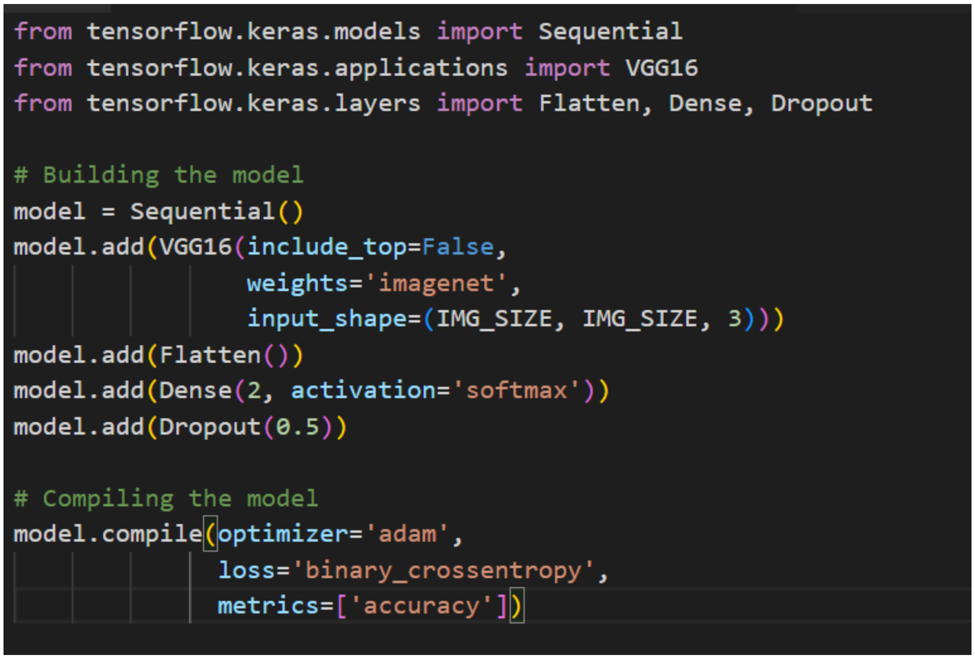
4. Code Implemented:

1) Data Preprocessing



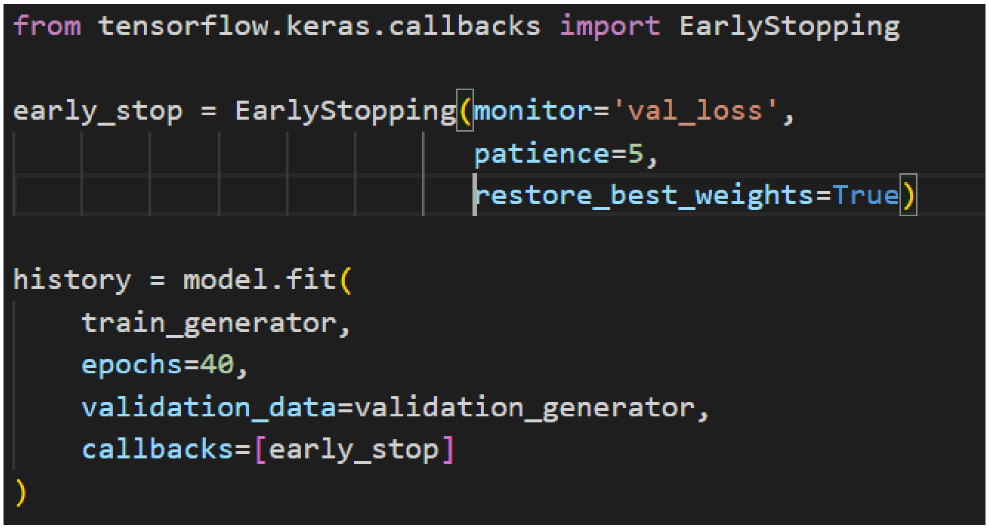
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import preprocess_input

# Define image size
IMG_SIZE = 800

# Data augmentation and preprocessing
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    validation_split=0.2
)

validation_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    validation_split=0.2
)

# Load training images
train_generator = train_datagen.flow_from_directory(
    directory=train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    color_mode="rgb",
    class_mode="categorical",
    batch_size=32,
    subset="training"
)

# Load validation images
validation_generator = validation_datagen.flow_from_directory(
    directory=validation_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    color_mode="rgb",
    class_mode="categorical",
    batch_size=32,
    subset="validation"
)
```

2) Model Architecture



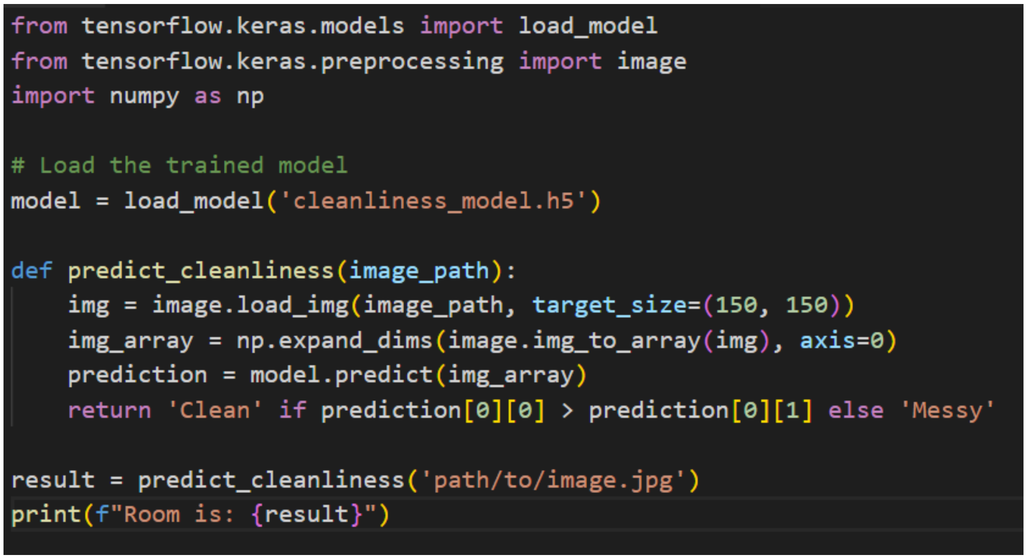
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Flatten, Dense, Dropout

# Building the model
model = Sequential()
model.add(VGG16(include_top=False,
                weights='imagenet',
                input_shape=(IMG_SIZE, IMG_SIZE, 3)))
model.add(Flatten())
model.add(Dense(2, activation='softmax'))
model.add(Dropout(0.5))

# Compiling the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

3) Model Training



```python
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss',
                           patience=5,
                           restore_best_weights=True)

history = model.fit(
    train_generator,
    epochs=40,
    validation_data=validation_generator,
    callbacks=[early_stop]
)
```

4) Prediction & Report Generation

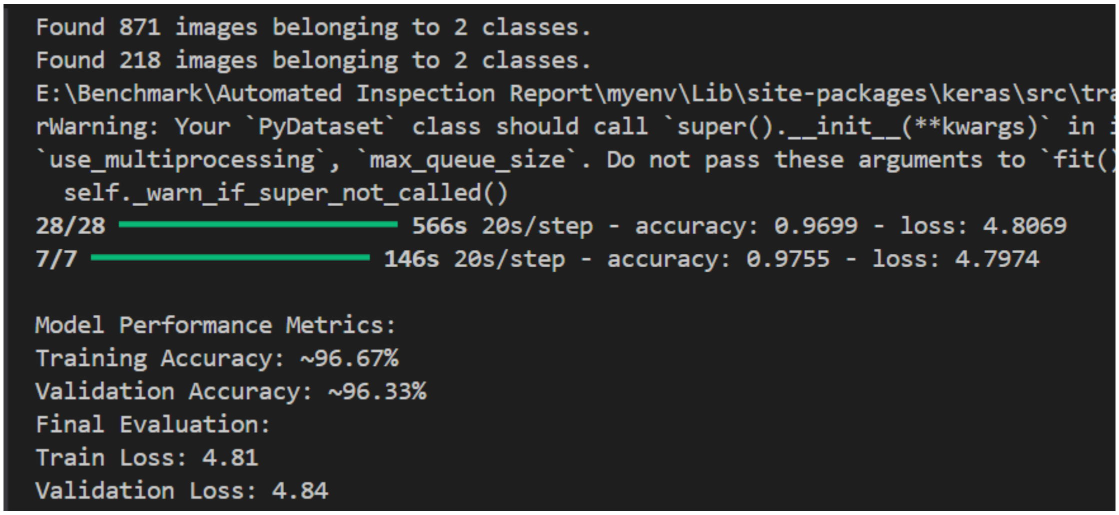


```python
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

# Load the trained model
model = load_model('cleanliness_model.h5')

def predict_cleanliness(image_path):
    img = image.load_img(image_path, target_size=(150, 150))
    img_array = np.expand_dims(image.img_to_array(img), axis=0)
    prediction = model.predict(img_array)
    return 'Clean' if prediction[0][0] > prediction[0][1] else 'Messy'

result = predict_cleanliness('path/to/image.jpg')
print(f"Room is: {result}")
```

5. Result:

1) Model Performance Metrics:

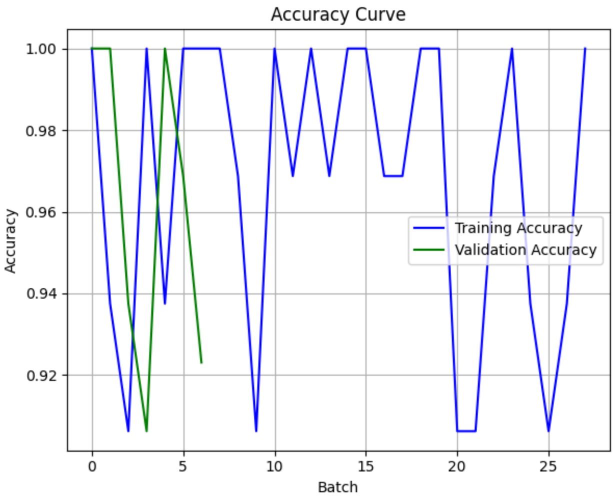
- Training Accuracy: ~96.67%
- Validation Accuracy: ~96.33%
- Final Evaluation:
    - Train Loss: 4.81
    - Validation Loss: 4.84



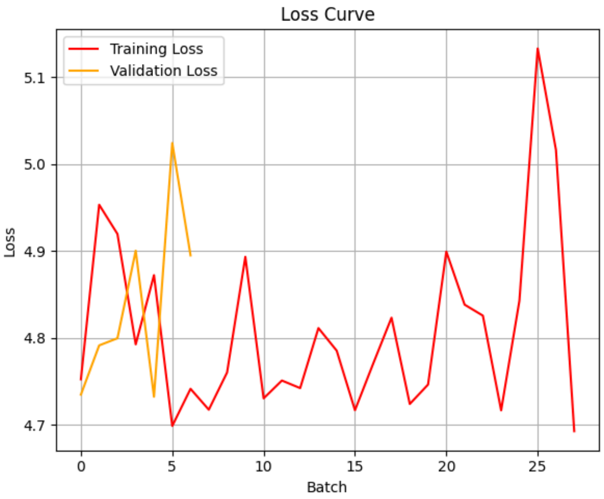
```
Found 871 images belonging to 2 classes.
Found 218 images belonging to 2 classes.
E:\Benchmark\Automated Inspection Report\myenv\Lib\site-packages\keras\src\tra
rWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in i
`use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()
  self._warn_if_super_not_called()
28/28 ━━━━━━━━━━━━━━━━━━━━ 566s 20s/step - accuracy: 0.9699 - loss: 4.8069
7/7 ━━━━━━━━━━━━━━━━━━━━ 146s 20s/step - accuracy: 0.9755 - loss: 4.7974

Model Performance Metrics:
Training Accuracy: ~96.67%
Validation Accuracy: ~96.33%
Final Evaluation:
Train Loss: 4.81
Validation Loss: 4.84
```

2) Learning Curves:

Accuracy Curve:



Loss Curve:



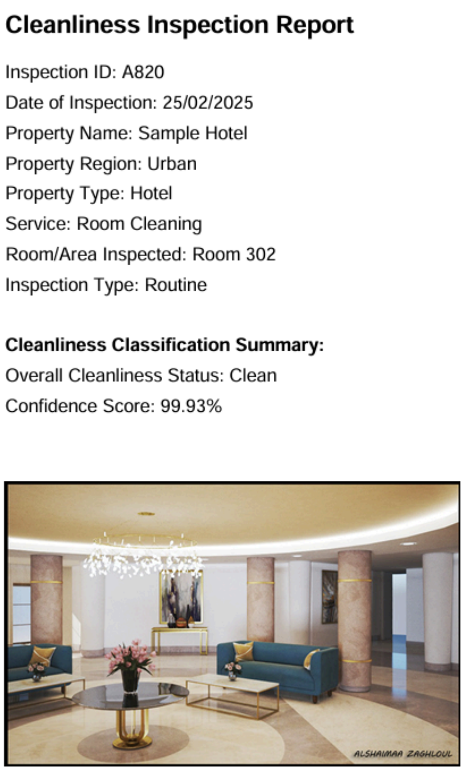
3) Classification Results:

- **Classes:**
  - Clean (Label: 0)
  - Messy (Label: 1)
- **Sample Image Prediction:**
  - Predicted Class: Clean/Messy
  - Confidence Score: ~99.00%

4) PDF Report Generation:

A PDF report titled **"Cleanliness Inspection Report"** was successfully generated, including:

- Inspection ID
- Date of Inspection
- Hotel and Room Details
- Cleanliness Status & Confidence Score
- Attached Image for Reference

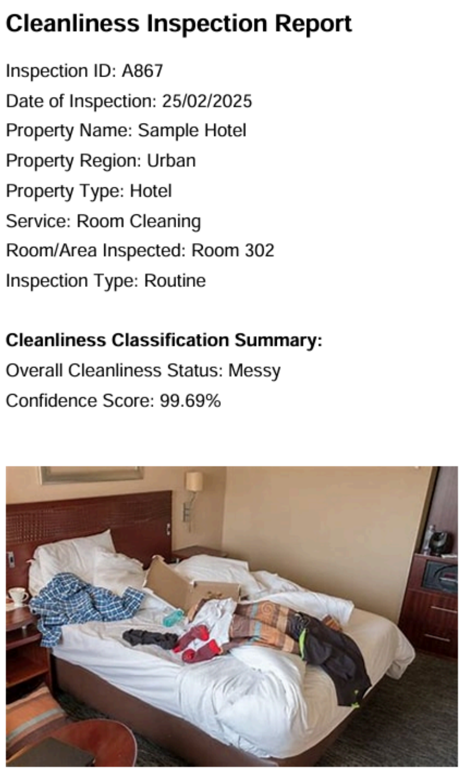


Cleanliness Inspection Report

Inspection ID: A820
Date of Inspection: 25/02/2025
Property Name: Sample Hotel
Property Region: Urban
Property Type: Hotel
Service: Room Cleaning
Room/Area Inspected: Room 302
Inspection Type: Routine

Cleanliness Classification Summary:
Overall Cleanliness Status: Clean
Confidence Score: 99.93%

ALSHAIMAA ZAGHLOUL

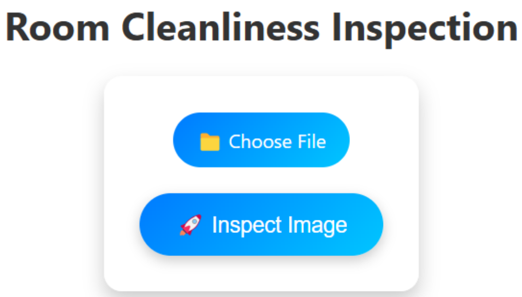


Cleanliness Inspection Report

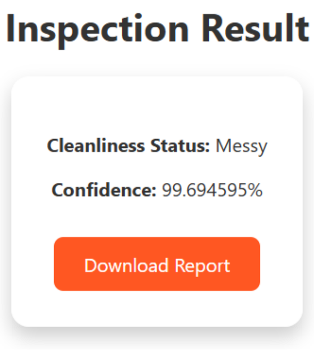
Inspection ID: A867
Date of Inspection: 25/02/2025
Property Name: Sample Hotel
Property Region: Urban
Property Type: Hotel
Service: Room Cleaning
Room/Area Inspected: Room 302
Inspection Type: Routine

Cleanliness Classification Summary:
Overall Cleanliness Status: Messy
Confidence Score: 99.69%

5) User Interface:



Room Cleanliness Inspection



Inspection Result

Cleanliness Status: Messy

Confidence: 99.694595%

6) Model Optimization Summary

- **Architecture:** Transfer Learning using **VGG16** with additional Dense and Dropout layers.
- **Optimizer:** Adam with a learning rate of `1e-5`
- **Loss Function:** Binary Crossentropy with label smoothing
- **Regularization:** L2 Regularization and Dropout (0.5) to reduce overfitting.

6. References:

1) jchen9619/Convolutional-Neural-Network-for-Messy-Clean-Room-Detection
2) https://www.kaggle.com/code/aayushmishra1512/messy-clean-vgg19
3) Top 4 Pre-Trained Models for Image Classification- Analytics Vidhya
4) https://youtu.be/taC5pMCm70U?feature=shared
5) https://youtu.be/zBOavqh3kWU?feature=shared
6) https://www.reportlab.com/opensource/
7) https://flask.palletsprojects.com/