# **Project Report**

## 1.INTRODUCTION:

### **1.1 Project Overview**

According to the next 25 years, the less developed countries' waste accumulation will increase drastically. With the increase in the number of industries in the urban area, the disposal of the solid waste is really becoming a big problem, and the solid waste includes paper, wood, plastic, metal, glass etc. The common way of managing waste is burning waste and this method can cause air pollution and some hazardous materials from the waste spread into the air which can cause cancer. Hence it is necessary to recycle the waste to protect the environment and human beings' health, and we need to separate the waste into different components which can be recycled using different ways.

### **1.2 Purpose**

The present way of separating waste/garbage is the hand-picking method, whereby someone is employed to separate out the different objects/materials. The person who separates waste, is prone to diseases due to the harmful substances in the garbage. With this in mind, it motivated us to develop an automated system which is able to sort the waste. and this system can take a short time to sort the waste, and it will be more accurate in sorting than the manual way. With the system in place, the beneficial separated waste can still be recycled and converted to energy and fuel for the growth of the economy. The system that is developed for the separation of the accumulated waste is based on the combination of Convolutional Neural Network

# 2.IDEATION & PROPOSED SOLUTION

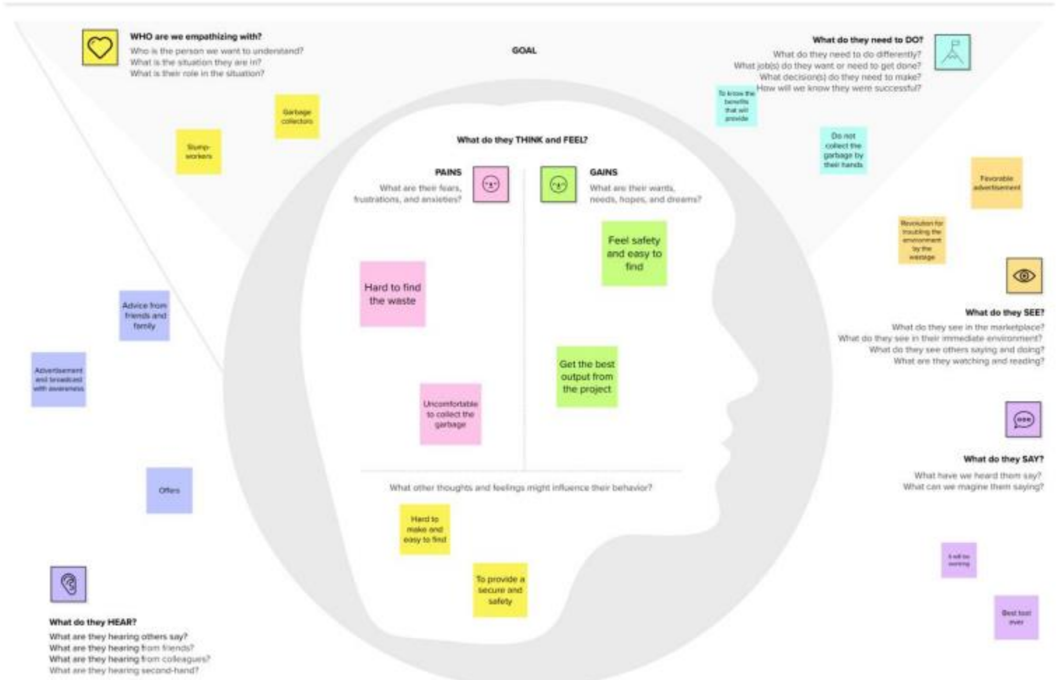
## 2.1 Problem Statement Definition

| Problem Statement (PS) | I am (Customer) | I'm trying to | But | Because | Which makes me feel |
|---|---|---|---|---|---|
| PS-1 | Garbage collector | Classify the Garbage by Separation | we are checking one by one is very hard so it's takes a long time | all the wastages are combining together and it's hard to find | uncomfortable to search the waste |
| PS-2 | Garbage separator | Separate the waste which can be recycled | It's leads to cause disease due to the harmful substances in the garbage | to collect the waste by hand | In problem |

### 2.2 Empathy Map Canvas

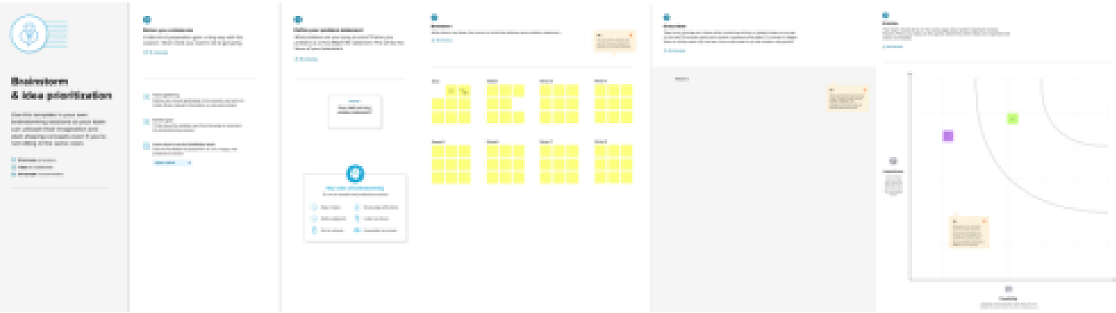
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.

Intelligent Garbage Classification using Deep learning

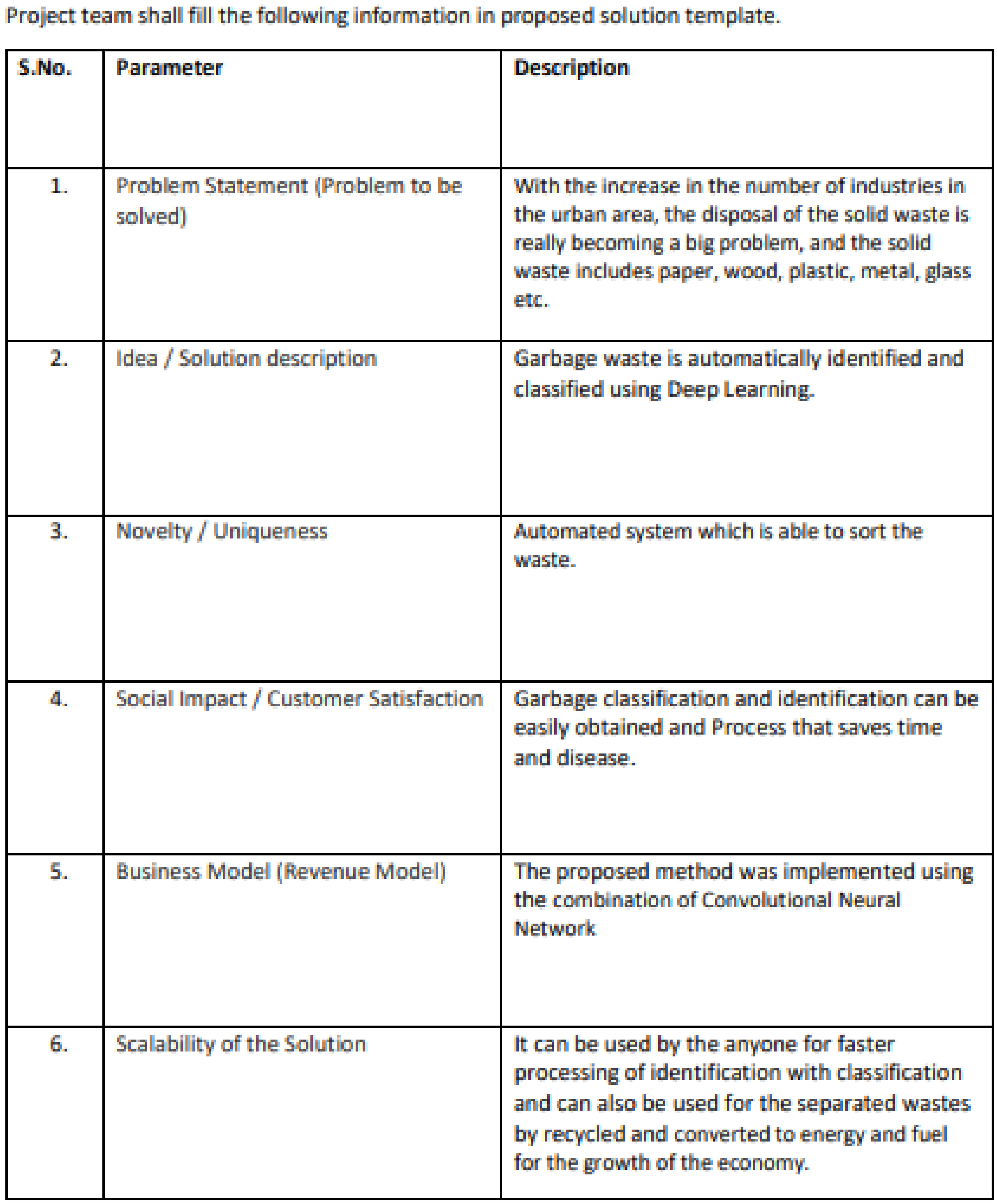

WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
GOAL
Garbage collectors
Slump-workers
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
To know the benefits that will provide
Do not collect the garbage by their hands
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
Hard to find the waste
Uncomfortable to collect the garbage
Feel safety and easy to find
Get the best output from the project
What other thoughts and feelings might influence their behavior?
Hard to make and easy to find
To provide a secure and safety
Favorable advertisement
Revolution for troubling the environment by the wastage
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they SAY?
What have we heard them say?
What can we magine them saying?
Best tool ever
Advice from friends and family
Advertisement and broadcast with awareness
Offers
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?

### 2.3 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.



## 2.4 Proposed Solution



Project team shall fill the following information in proposed solution template.

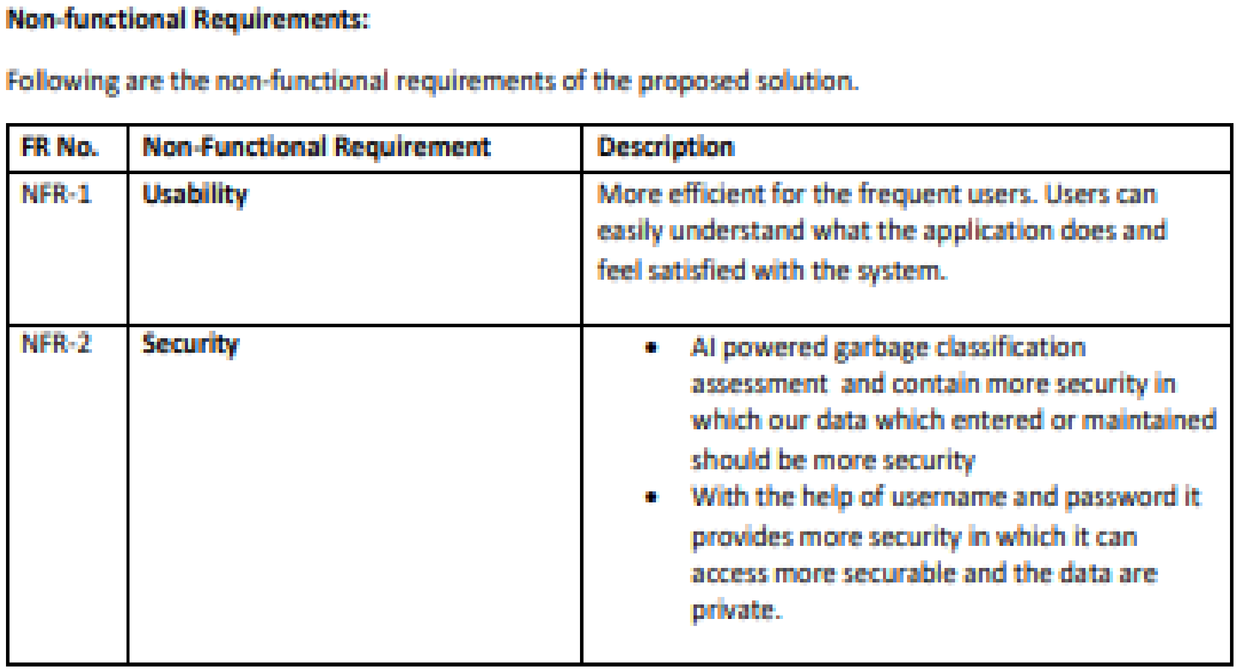
| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | With the increase in the number of industries in the urban area, the disposal of the solid waste is really becoming a big problem, and the solid waste includes paper, wood, plastic, metal, glass etc. |
| 2. | Idea / Solution description | Garbage waste is automatically identified and classified using Deep Learning. |
| 3. | Novelty / Uniqueness | Automated system which is able to sort the waste. |
| 4. | Social Impact / Customer Satisfaction | Garbage classification and identification can be easily obtained and Process that saves time and disease. |
| 5. | Business Model (Revenue Model) | The proposed method was implemented using the combination of Convolutional Neural Network |
| 6. | Scalability of the Solution | It can be used by the anyone for faster processing of identification with classification and can also be used for the separated wastes by recycled and converted to energy and fuel for the growth of the economy. |

# 3. **REQUIREMENT ANALYSIS**

### Functional Requirements:

Following are the functional requirements of the proposed solution.

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | Garbage Classification | • Classified by User<br>• Classified by Deep learning<br>• Classified by Sensor |
| FR-2 | Garbage Confirmation | • Confirmation through Sensor<br>• Confirmation through CNN |
| FR-3 | Garbage Details | Users are required to register the garbage details like name of the waste, type, shape etc. |
| FR-4 | Requirements | The user which is able to sort the waste with the system can take a short time to sort the waste, and it will be more accurate in sorting than the manual way. With the system in place, the beneficial separated waste can still be recycled and converted to energy and fuel for the growth of the economy. The system that is developed for the separation of the accumulated waste is based on the combination of Convolutional Neural Network. |



### Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

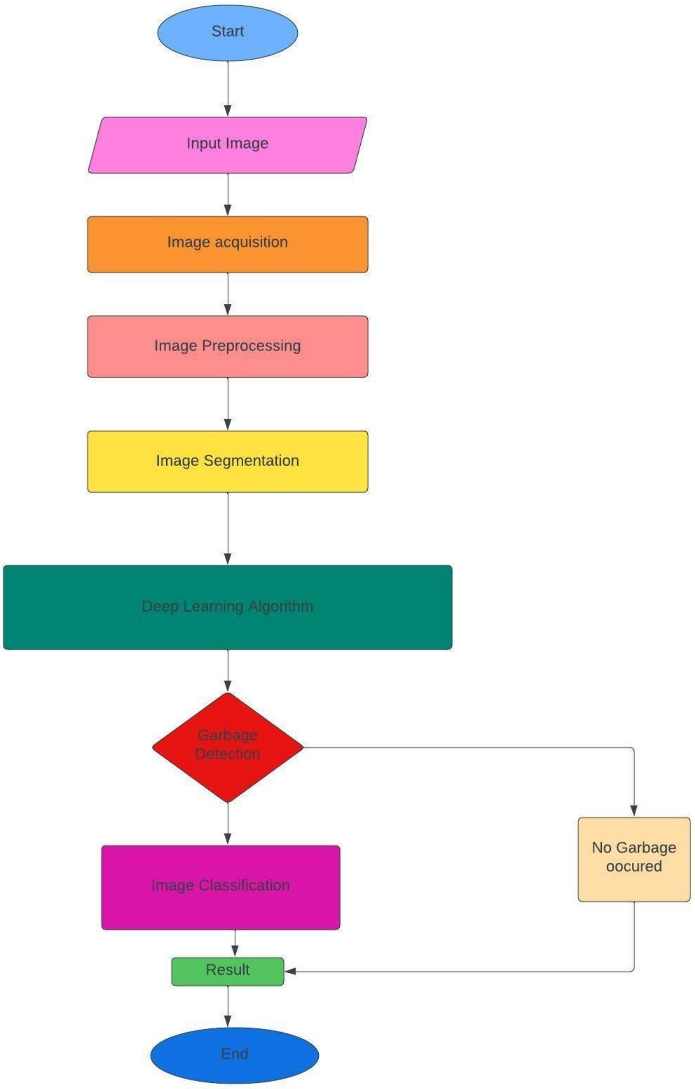
| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | Usability | More efficient for the frequent users. Users can easily understand what the application does and feel satisfied with the system. |
| NFR-2 | Security | • AI powered garbage classification assessment and contain more security in which our data which entered or maintained should be more security<br>• With the help of username and password it provides more security in which it can access more securable and the data are private. |

|  |  |  |
|---|---|---|
| NFR-3 | **Reliability** | This application must perform without failure in 90 percent of use cases during a month.it is more reliable. |
| NFR-4 | **Performance** | This application supporting 1,050 users per hour must provide 5 seconds or less response time in a desktop browser, including the rendering of text and images, over an LTE connection. The performance of application is effective and efficient. |
| NFR-5 | **Availability** | The web dashboard must be available to user's 99.9 percent of the time every month during business hours EST. Users can access anytime and anywhere. |
| NFR-6 | **Scalability** | The application must be scalable enough to support 10,000 visits at the same time while maintain optimal performance and efficient to retrieve image in large scale thus improving scalability. |

# 4. PROJECT DESIGN

## 4.1 Data Flow Diagrams

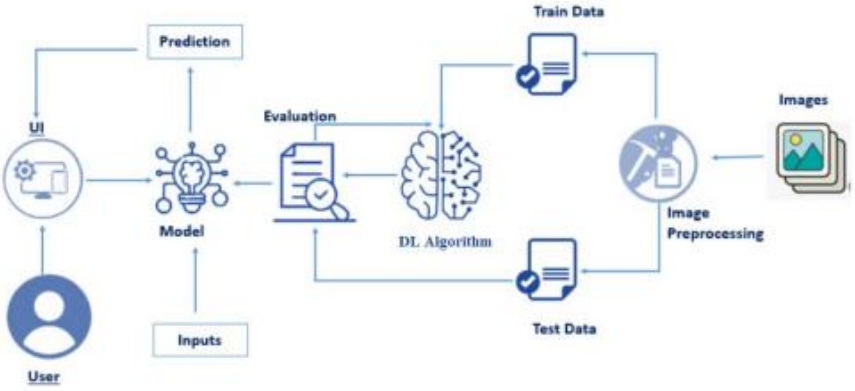
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the rightamount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



## 4.2 Solution & Technical Architecture

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | How user interacts with application e.g. Web UI, Mobile App, Chatbot etc. | HTML, CSS, JavaScript / Angular Js / React Js etc. |
| 2. | Application Logic-1 | Logic for a process in the application | Java / Python |
| 3. | Application Logic-2 | Logic for a process in the application | IBM Watson STT service |
| 4. | Application Logic-3 | Logic for a process in the application | IBM Watson Assistant |
| 5. | Database | Data Type, Configurations etc. | MySQL, NoSQL, etc. |
| 6. | Cloud Database | Database Service on Cloud | IBM DB2, IBM Cloudant etc. |
| 7. | File Storage | File storage requirements | IBM Block Storage or Other Storage Service or Local Filesystem |
| 8. | External API-1 | Purpose of External API used in the application | IBM Weather API, etc. |
| 9. | External API-2 | Purpose of External API used in the application | Aadhar API, etc. |
| 10. | Machine Learning Model | Purpose of Machine Learning Model | Object Recognition Model, etc. |
| 11. | Infrastructure (Server / Cloud) | Application Deployment on Local System / Cloud<br>Local Server Configuration:<br>Cloud Server Configuration : | Local, Cloud Foundry, Kubernetes, etc. |

**Table-2: Application Characteristics:**

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | List the open-source frameworks used | Technology of Opensource framework |
| 2. | Security Implementations | List all the security / access controls implemented, use of firewalls etc. | e.g. SHA-256, Encryptions, IAM Controls, OWASP etc. |
| 3. | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | Technology used |
| 4. | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | Technology used |
| 5. | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc. | Technology used |

**User Stories**

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Team Member |
|---|---|---|---|---|---|---|
| Customer (Mobile user) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can access my account / dashboard | High | Thiru |
| | | USN-2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Surya |
| | | USN-3 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | Low | Rithishree |
| | | USN-4 | As a user, I can register for the application through Gmail | | Medium | Thiru |
| | Login | USN-5 | As a user, I can log into the application by entering email & password | | High | Surya |
| | Dashboard | | | | | |
| Customer (Web user) | | | | | | |
| Customer Care Executive | | | | | | |
| Administrator | | | | | | |
| | | | | | | |

## 5. CODING & SOLUTIONING

### 5.1 Feature 1

# IMAGE PREPROCESSING

# 1.Import The ImageDataGenerator Library

In [25]:

```
from keras.preprocessing.image import ImageDataGenerator
```

# 2.Configure ImageDataGenerator

In [26]:

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.1,
                                   zoom_range = 0.1,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

# 3. Apply ImageDataGenerator Functionality To Trainset And Testset

In [28]:

```
train_transform =
train_datagen.flow_from_directory('/content/drive/MyDrive/Dataset/Garba
ge classification/Garbage classification',
                                                 target_size=(128,128),
                                                 batch_size=64,

class_mode='categorical')
test_transform =
test_datagen.flow_from_directory('/content/drive/MyDrive/Dataset/Garbag
e classification/Garbage classification',
                                            target_size=(128,128),
                                            batch_size=64,
                                            class_mode='categorical')
```

```
Found 2527 images belonging to 6 classes.
Found 2527 images belonging to 6 classes.
```

# MODEL BUILDING

### 1. Importing The Model Building Libraries

In [30]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.optimizers import Adam
```

### 2.Initialize The Model

In [31]:

```
model=Sequential()
```

### 3.Adding CNN Layers

In [40]:

```
model=Sequential()
```

In [41]:

```
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='re
lu'))
model.add(MaxPooling2D(2,2))
```

In [42]:

```
model.add(Convolution2D(64,(3,3),padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))
```

In [43]:

```
model.add(Convolution2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
```

In [44]:

```
model.add(Convolution2D(32,(3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))
```

In [45]:

```
model.add(Flatten())
```

### 4.Adding Dense Layers

In [46]:

```
model.add(Dense(kernel_initializer='uniform',activation='relu',units=15
0))
model.add(Dense(kernel_initializer='uniform',activation='relu',units=68
))
model.add(Dense(kernel_initializer='uniform',activation='relu',units=6)
)
```

### 5.Creating A Model Object

In [47]:

```
model.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_7 (Conv2D)           (None, 126, 126, 32)      896       
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 63, 63, 32)       0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 63, 63, 64)        18496     
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 31, 31, 64)       0         
 2D)                                                             
                                                                 
 conv2d_9 (Conv2D)           (None, 29, 29, 32)        18464     
                                                                 
 max_pooling2d_8 (MaxPooling  (None, 14, 14, 32)       0         
 2D)                                                             
                                                                 
 conv2d_10 (Conv2D)          (None, 14, 14, 32)        9248      
                                                                 
 max_pooling2d_9 (MaxPooling  (None, 7, 7, 32)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 1568)              0         
                                                                 
 dense (Dense)               (None, 150)               235350    
                                                                 
 dense_1 (Dense)             (None, 68)                10268     
                                                                 
 dense_2 (Dense)             (None, 6)                 414       
                                                                 
=================================================================
Total params: 293,136
Trainable params: 293,136
Non-trainable params: 0
_________________________________________________________________
```

### 6. Configure the Learning Process

In [48]:

```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['acc']
)
```

## 7. Train the Model

In [49]:

```
res = model.fit_generator(
  train_transform,
  steps_per_epoch=2527//64,
  validation_steps=782//64,
  epochs=30,
  validation_data=test_transform
)
```

```
<ipython-input-49-cf483d53cb30>:1: UserWarning: `Model.fit_generator` is de
precated and will be removed in a future version. Please use `Model.fit`, w
hich supports generators.
  res = model.fit_generator(
Epoch 1/30
39/39 [==============================] - 969s 25s/step - loss: 6.7970 - acc
: 0.2274 - val_loss: 6.4493 - val_acc: 0.2630
Epoch 2/30
39/39 [==============================] - 100s 3s/step - loss: 6.5715 - acc:
0.2716 - val_loss: 7.2315 - val_acc: 0.3086
Epoch 3/30
39/39 [==============================] - 100s 3s/step - loss: 6.7892 - acc:
0.2647 - val_loss: 6.5556 - val_acc: 0.2435
Epoch 4/30
39/39 [==============================] - 97s 2s/step - loss: 6.6160 - acc:
0.2335 - val_loss: 6.6227 - val_acc: 0.2240
Epoch 5/30
39/39 [==============================] - 98s 3s/step - loss: 6.6442 - acc:
0.2326 - val_loss: 6.6841 - val_acc: 0.2266
Epoch 6/30
39/39 [==============================] - 101s 3s/step - loss: 6.6060 - acc:
0.2452 - val_loss: 6.7107 - val_acc: 0.2826
Epoch 7/30
39/39 [==============================] - 100s 3s/step - loss: 6.5619 - acc:
0.2611 - val_loss: 6.6324 - val_acc: 0.2005
Epoch 8/30
39/39 [==============================] - 99s 3s/step - loss: 6.6423 - acc:
0.2261 - val_loss: 6.7685 - val_acc: 0.2266
Epoch 9/30
39/39 [==============================] - 99s 2s/step - loss: 6.6822 - acc:
0.2335 - val_loss: 6.3158 - val_acc: 0.2539
Epoch 10/30
39/39 [==============================] - 98s 3s/step - loss: 6.6277 - acc:
0.2318 - val_loss: 6.8581 - val_acc: 0.2292
Epoch 11/30
39/39 [==============================] - 99s 3s/step - loss: 6.6433 - acc:
0.2359 - val_loss: 6.4943 - val_acc: 0.2578
Epoch 12/30
39/39 [==============================] - 100s 3s/step - loss: 6.6492 - acc:
0.2351 - val_loss: 6.6126 - val_acc: 0.2461
Epoch 13/30
```

```
39/39 [==============================] - 99s 3s/step - loss: 6.6206 - acc:
0.2352 - val_loss: 7.0745 - val_acc: 0.2305
Epoch 14/30
39/39 [==============================] - 100s 3s/step - loss: 6.6186 - acc:
0.2371 - val_loss: 6.6327 - val_acc: 0.2305
Epoch 15/30
39/39 [==============================] - 98s 3s/step - loss: 6.6725 - acc:
0.2343 - val_loss: 6.5969 - val_acc: 0.2318
Epoch 16/30
39/39 [==============================] - 98s 3s/step - loss: 6.6019 - acc:
0.2351 - val_loss: 6.3088 - val_acc: 0.2305
Epoch 17/30
39/39 [==============================] - 100s 3s/step - loss: 6.5992 - acc:
0.2359 - val_loss: 6.6669 - val_acc: 0.2292
Epoch 18/30
39/39 [==============================] - 98s 3s/step - loss: 6.6192 - acc:
0.2343 - val_loss: 6.6865 - val_acc: 0.2500
Epoch 19/30
39/39 [==============================] - 100s 3s/step - loss: 6.6389 - acc:
0.2363 - val_loss: 6.3719 - val_acc: 0.2305
Epoch 20/30
39/39 [==============================] - 99s 3s/step - loss: 6.6332 - acc:
0.2383 - val_loss: 6.6542 - val_acc: 0.2331
Epoch 21/30
39/39 [==============================] - 98s 3s/step - loss: 6.6270 - acc:
0.2814 - val_loss: 7.0311 - val_acc: 0.2057
Epoch 22/30
39/39 [==============================] - 97s 2s/step - loss: 6.6268 - acc:
0.2322 - val_loss: 6.5411 - val_acc: 0.2318
Epoch 23/30
39/39 [==============================] - 98s 3s/step - loss: 6.5970 - acc:
0.2367 - val_loss: 6.2787 - val_acc: 0.2799
Epoch 24/30
39/39 [==============================] - 98s 3s/step - loss: 6.5837 - acc:
0.2858 - val_loss: 6.3961 - val_acc: 0.2812
Epoch 25/30
39/39 [==============================] - 99s 3s/step - loss: 6.5251 - acc:
0.3642 - val_loss: 6.3526 - val_acc: 0.3997
Epoch 26/30
39/39 [==============================] - 98s 3s/step - loss: 6.5248 - acc:
0.3735 - val_loss: 6.5666 - val_acc: 0.3841
Epoch 27/30
39/39 [==============================] - 100s 3s/step - loss: 6.5299 - acc:
0.3780 - val_loss: 6.2806 - val_acc: 0.3815
Epoch 28/30
39/39 [==============================] - 98s 3s/step - loss: 6.5160 - acc:
0.3894 - val_loss: 6.7936 - val_acc: 0.3906
Epoch 29/30
39/39 [==============================] - 99s 3s/step - loss: 6.4948 - acc:
0.3930 - val_loss: 6.8273 - val_acc: 0.3529
Epoch 30/30
39/39 [==============================] - 100s 3s/step - loss: 6.4496 - acc:
0.4060 - val_loss: 6.3682 - val_acc: 0.3919
```

**8.Save the Model**

In [50]:

```
model.save('Garbage1.h5')
```

### 9.Test the Model

In [52]:

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("Garbage1.h5")
```

In [53]:

```
img=image.load_img(r"/content/drive/MyDrive/Dataset/Garbage
classification/Garbage
classification/glass/glass17.jpg",target_size=(128,128))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
```

In [54]:

```
a=np.argmax(model.predict(x), axis=1)
```

```
1/1 [==============================] - 0s 161ms/step
```

In [55]:

```
index=['0','1','2','3','4','5']
result=str(index[a[0]])
result
```

Out[55]:

```
'4'
```

In [57]:

```
index1=['cardboard','glass','metal','paper','plastic','trash']
result1=str(index1[a[0]])
result1
```

Out[57]:

```
'plastic'
```

### 5.2 Feature 2

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building server side script

# 6. RESULTS

## 6.1 Performance Metrics

### Model description:

The model used here is a reduced version on VGG network with height=96, width=96, depth=3 and class=2 (organic/recyclable).

Find the model in model_O_R directory.

| Train Accuracy | Train Loss | Validation Accuracy | Validation Loss |
|---|---|---|---|
| 0.92 | 0.23 | 0.900 | 0.248 |

### Model 2 (Six type classification)

The **Second model** is a six type garbage classification model. The different classes into which the input image is classified are 1. Cardboard 2. Glass 3. Metal 4. Paper 5. Plastic 6. Trash.

In this, the ImageDataGenerator keras.preprocessing.image is used to create the train set and validation set.

```
train = ImageDataGenerator(horizontal_flip = True, vertical_flip = True,
                         validation_split = 0.1, rescale = 1./255,
                         shear_range = 0.2, zoom_range = 0.2,
                         width_shift_range = 0.1, height_shift_range = 0.1,)

test = ImageDataGenerator(rescale = 1/255, validation_split = 0.1)

train_generator = train.flow_from_directory(dir_path,
                                          target_size = (300,300),
                                          batch_size = 32,
                                          class_mode = 'categorical',
                                          subset = 'training')
```

```
valid_generator = test.flow_from_directory(dir_path,
                                        target_size = (300,300),
                                        batch_size = 32,
                                        class_mode = 'categorical',
                                        subset = 'validation')
```

The train and validation split is 90:1. After that, the base model is InceptionV3 with pretrained weights. From this weights, current classification model is trained. Due to low number of train images the Accuracy of the model is low.

| Train Accuracy | Train Loss | Validation Accuracy | Validation Loss |
|---|---|---|---|
| 0.8226 | 0.5107 | 0.6941 | 0.6847 |

## 8. CONCLUSION

The application is integrated to a camera or you can add a captured video/image. The model detects the type of garbage in the image and outputs the message indicating the type of garbage to the user. This can be integrated into an automated machine to seperate trash based on it's type. This project focuses on the software implementation and not the hardware.

## 9. FUTURE SCOPE

In future ,We will add more features like live camera detection and modification. It will help customer to find all the services on the same application.

**10. APPENDIX**

**Source code:**

**#index.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

  <link rel="stylesheet" href="https://unpkg.com/swiper@7/swiper-bundle.min.css" />
  <link rel="stylesheet" href="index.css">
  <title>Home</title>

</head>



<body>
  <header>
    <div class="title1">
      <h1>GARBAGE CLASSIFICATION</h1>
    </div>
      <div class="main">
      <ul>
        <li><a href="#Home" class="btn">Home</a></li>
        <li><a href="#About" class="btn">About</a></li>
        <li><a href="#Contact" class="btn">Contact</a></li>
        <li><a href="#Image Prediction" class="btn">Image
Predicition</a></li>
      </ul>
    </div>


  </header>
 <section id="Home">
   <div class="img1">
    <div class="name">
      <h3>Welcome to <span>Garbage Classification</span> </h3>


      <a href="" class="btn">ReadMore</a>


    </div>
  </div>
 </section>
 <section id="About" class="about">
  <div class="about1">About</div>
  <div class="about2"><h1>ABOUT PROJECT</h1></div>
  <div class="container">
  <div class="about3">


    <h3>Problem:</h3>
```

```
        <p>"The accumulation of solid waste in the urban area is becoming a
great concern, and it would result in environmental pollution and may be
hazardous to human health if It is not properly managed. It is important to have
an advanced/intelligent waste management system to manage a variety of waste
materials. One of the most important steps of waste management is the
separation of the waste into the different components and this process is
normally done manually by hand-picking. To simplify the process, we propose
an intelligent waste material classification system, which is developed by using
the 50-layer residual net pre-train (ResNet-50) Convolutional Neural Network
model which is a machine learning tool and serves as the extractor, and Support
Vector Machine (SVM) which is used to classify the waste into different
groups/types such as glass, metal, paper, and plastic etc. The proposed system is
tested on the trash image dataset which was developed by Gary Thung and
Mindy Yang, and is able to achieve an accuracy of 87% on the dataset. The
separation process of the waste will be faster and intelligent using the proposed
waste material classification system without or reducing human
involvement."</p>


    </div>


    <div class="about4">
        <h3>Solution:</h3>


        <p>"The present way of separating waste/garbage is the hand-picking
method, whereby someone is employed to separate out the different
objects/materials. The person, who separate waste. is prone to diseases due to
the harmful substances in the garbage. With this in mind, it motivated us to
develop an automated system which is able to sort the waste, and this system
can take short time to sort the waste, and it will be more accurate in sorting than
the manual way. With the system in place, the separated waste can still be
recycled and converted to energy and fuel for the growth of the economy. The
system that is developed for the separation of the accumulated waste is based on
the combination of Convolutional Neural Network."</p>
        <a href="" class="btn1">ReadMore</a>
    </div>
```

```
</div>
 </section>


 <section id="Contact" class="contact">
    <div class="contact1">Contact</div>
    <div class="contact2"><h1>CONTACT US</h1></div>
    <div id="services" class="row">
        <div class="meaw">
            <h3>Our Address</h3>
            <p>Plot No 132, 2nd floor ,Above DCB<br>
                Bank,HMT Nagar , Nacharam Main <br>Road , Hyderabad -
500076</p>
        </div>


      <div class="meaw">
          <h3>Email Us </h3>
          <p>info@thesmartbridge.com</p>
      </div>
          <div class="meaw">
              <h3>Call Us</h3>
              <p>+914035112535</p>
          </div>
 </section>


 <section class="footer">
 <div class="box-container1">
    <div class="box">
        <h3>Smart Bridge</h3>
    <p> <i class="fas fa-phone"></i>+914035112535 </p>
    <p> <i class="fas fa-envelope"></i>info@thesmartbridge.com </p>
    <p> <i class="fas fa-map"></i> Plot No 132, 2nd floor ,Above DCB<br>
      Bank,HMT Nagar , Nacharam Main <br>Road , Hyderabad - 500076</p>

  </div>
  <div class="box">
    <h3>Quick Links</h3>
    <a class="links" href="#Home">Home</a>
    <a class="links" href="#About">About</a>
    <a class="links" href="#Contact" >Contact</a>
  </div>

</div>
</section>

</body>
</html>
```

Index.css:

```
*{
  margin: 0;
  padding: 0;

}
html{
  scroll-behavior: smooth;
  background-color: #ffffff;


}


header{
  height: 10vh;
  background-size:cover;
  background-position: center;
  background-attachment: fixed;
  background-color: rgb(0, 0, 0);
}
.title1{
  font-family: "Amazon Ember",sans-serif;
  float: left;
  color: #ffffff;
  margin-top: 20px;
  padding-left: 60px;


}


ul{
  float: right;
  list-style-type: none;
  margin-top: 30px;
  font-family: Century Gothic ;
}
```

```
ul li{
    display: inline-block;


}
.img1{


    background-repeat: no-repeat;
    background-size: cover;
    background-image: linear-gradient(rgba(0, 0, 0, 0.488),rgba(0, 0, 0,
0.756)),url(images/garbagebg.jpeg);
    height:660px ;




}



.img1 .name h3{
    font-size: 2rem;
    color: #ffffff;
    position: absolute;
    top: 50%;
    left: 25%;
    transform: translate(-50%,-50%);
    padding-left: 5px;
    font-family: "Amazon Ember",sans-serif;
}
```

```
.btn{
   text-decoration: none;
   border-radius: 0.6rem;
   color: #ffffff;
   padding: 5px 25px;
   border: 1px solid transparent;
   transition: 1s ease;
   font-family: "Amazon Ember",sans-serif;

}
.btn:hover{


   border-radius: 0.6rem;
   background-color:rgb(255, 111, 0);;
   color: #000000;


}
.btn:focus {
   border-color: rgb(255, 255, 255);
   box-shadow: rgba(78, 78, 78, 0.5) 0 2px 5px 0;
   outline: 0;
  }
  .img1 .name .btn{
   position: absolute;
   font-size: 20px;
   background-color: rgb(230, 39, 10);
   color: #ffff;
```

```
    top: 55%;
    left:7%;
    font-family: "Amazon Ember",sans-serif;

  }
  .img1 .name .btn:hover{
    background-color:rgb(255, 111, 0);
    color: #000000;
    transition-duration: 1s ;
  }

  .img1 .name span{


    color:rgb(255, 111, 0);;
  }
.about{
    padding: 3rem 8%;
    display: inline-block;
    background-color: aliceblue;
}
.about1 {
    text-align:left;
    margin-bottom: 2rem;
    position: relative;
    font-family: "Amazon Ember",sans-serif;
  }
  .about1::before {
    content: '';
    position: absolute;
    top: 50%;
    height: 0.1rem;
    width: 10%;
    left: 4.5%;
    background: rgb(255, 111, 0);
    z-index: 2;
  }
  .about2{
color: rgb(45, 40, 90);
font-family: "Amazon Ember",sans-serif;
margin-top: 1%;
  }

.container{


  margin-top: 2%;
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: (minmax(29rem, 1fr))[auto-fit];
      grid-template-columns: repeat(auto-fit, minmax(29rem, 1fr));
  gap: 2rem;

}

.about3 p{
  font-size: 1rem;
}

 .about4 p{
  font-size: 1rem;
}
.btn1{
  position: absolute;
  margin-top: 5%;
  text-decoration: none;
    border-radius: 0.6rem;
    color: #fcfcfc;
    background-color: rgb(255, 0, 0);
    padding: 5px 25px;
    border: 1px solid transparent;
    transition: 1s ease;
    font-family: "Amazon Ember",sans-serif;

}
 .btn1:hover{
  border-radius: 0.6rem;
    background-color:rgb(255, 111, 0);;
    color: #000000;
 }
 .contact{
  justify-content: center;
  padding: 3rem 8%;
  background-color:aliceblue;
 }
.contact1{
  text-align:left;
    margin-bottom: 2rem;
    position: relative;
    font-family: "Amazon Ember",sans-serif;
}
.contact1::before{
  content: '';
    position: absolute;
    top: 50%;
    height: 0.1rem;
    width: 10%;
    left: 5.5%;
    background: rgb(255, 111, 0);
    z-index: 2;
}
.contact2{
  color: rgb(45, 40, 90);
font-family: "Amazon Ember",sans-serif;
margin-top: 1%;
}
.services{
  width:80%;
  margin: auto;
  text-align: center;
  padding-top: 100px;
}

h1{
  font-size: 38px;
  font: weight 6px;
}
p{
  color: #777;
  font-size: 14px;
  font-weight: 300;
  line-height: 22px;
  padding: 10px;
  text-align: center;
}
.row{
  margin-top: 1%;
  display: flex ;
  justify-content: space-between;
}


.meaw{
  flex-basis: 31%;
  background: rgba(0, 213, 255, 0.159);
  border-radius: 10px;
  margin-bottom: 5%;
  padding: 20ps 12px;
  box-sizing: border-box;
  transition: 0.5s;
}
h3{
  text-align: center;
  font-weight: 600;
  margin: 10px 0;
}
.meaw:hover{
   box-shadow: 0 0 20px 0px rgba(244, 8, 8, 0.5);
}


.footer .box-container1 {
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: (minmax(25rem, 1fr))[auto-fit];
      grid-template-columns: repeat(auto-fit, minmax(25rem, 1fr));
  gap: 1.5rem;
  background-color: #000000;
}

.footer .box-container1 .box h3 {
  font-size: 2rem;
  color:#f00;
  padding: 1rem 0;
```

```
}

.footer .box-container1 .box .links {
  font-size: 1.5rem;
  display: block;
  color: #aaa;
  padding: 1rem 0;
}

.footer .box-container1 .box .links:hover {
  color: #f00;
}

.footer .box-container1 .box p {
  font-size: 1.5rem;
  color: #aaa;
  padding: 1rem 0;
}

.footer .box-container1 .box p i {
  padding-right: .5rem;
  color: #f00;
}

.footer .box-container1 .box .share {
  padding: 1rem 0;
}
```

```
.footer .box-container1 .box .share a {
  height: 4.5rem;
  width: 4.5rem;
  line-height: 4.5rem;
  font-size: 1.7rem;
  color: #fff;
  background: #111;
  border-radius: 50%;
  margin-right: 2rem;
  text-align: center;
}

.footer .box-container1 .box .share a:hover {
  background: #f00;
}

.footer .box-container1.box form .email {
  margin-bottom: 1rem;
  width: 100%;
  background: #111;
  padding: 1.2rem;
  font-size: 1.5rem;
  color: #fff;
  text-transform: none;
}
```

## Index.py:

```
from __future__ import division, print_function

#coding-utf-8

import sys

import os

import glob

import numpy as np

from tensorflow.keras.preprocessing import image

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Dropout

from tensorflow.keras.applications.imagenet_utils import
preprocess_input, decode_predictions

from tensorflow.keras.models import load_model

from tensorflow.keras import backend

from tensorflow.keras import backend

from tensorflow import keras

import tensorflow as tf

from skimage.transform import resize


from flask import Flask, redirect, url_for, request, render_template

from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
```

In [ ]:

```
from __future__ import division, print_function
import sys
```

In [ ]:

```
@app.route('/',methods=['GET'])

def index () :

  return render_template ('index.html')
```

```
@app.route('/Image', methods=['POST', 'GET'])\
 def prediction ():

return render_template('base.html')
```

In [ ]:

```
@app.route('/predict', methods=['GET', 'POST'])

def upload():

if request.method == 'POST':
   f= request.files ['image']


basepath= os.path.dirname('/content/drive/MyDrive/Dataset/Garbage
classification/Garbage classification')

file_path = os.path.join(

basepath, 'predictions', f.filename)

f.save(file_path)

img=image.load_img(file_path, target_size=(128, 128))

x = image.img_to_ray (img)
x= np.expand_dims (x, axis=0)

preds=model.predict_classes (x)

index = ['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']

text = "The Predicted Garbage is : "+str(index [preds[0]])


return text
```

In [ ]:

```
img-image.load_img(r"/content/drive/MyDrive/Dataset/Garbage
classification/Garbage classification/glass/glass17.jpg",
target_size=(128,128))

x=image.img_to_array(img) #converting in to array format

x=np.expand_dims(x,axis-0) #changing its dimensions as per our
requirement

#img_dato-preprocess_input(x)

#img_data.shape

a-np.argmax(model.predict(x), axis=1)
```

In [ ]:

```
@app.route('/predict',methods=['GET', 'POST'])
```

```
def upload():
```

Github link: https://github.com/naanmudhalvan-SI/PBL-NT-GP-14343-1682679985/tree/main