PROJECT REPORT FORMAT

Team- 592449

1. INTRODUCTION

- **1.1 Project overview:**
  Our project, "Fake/Real Logo Detection using Deep Learning," is a collaborative effort by a skilled team consisting of Badri, Vidya, Pranit, and Vivek. This innovative solution combines front-end development, machine learning integration, and a diverse tech stack to address the rising issue of counterfeit logos in the digital space.

- **1.2 Purpose:**
  The primary purpose of our project is to create a robust system capable of distinguishing between authentic and fake logos. Beyond logo authentication, we aim to provide additional features, such as brand identification, to enhance user experience and contribute to a safer online environment.

2. LITERATURE REVIEW

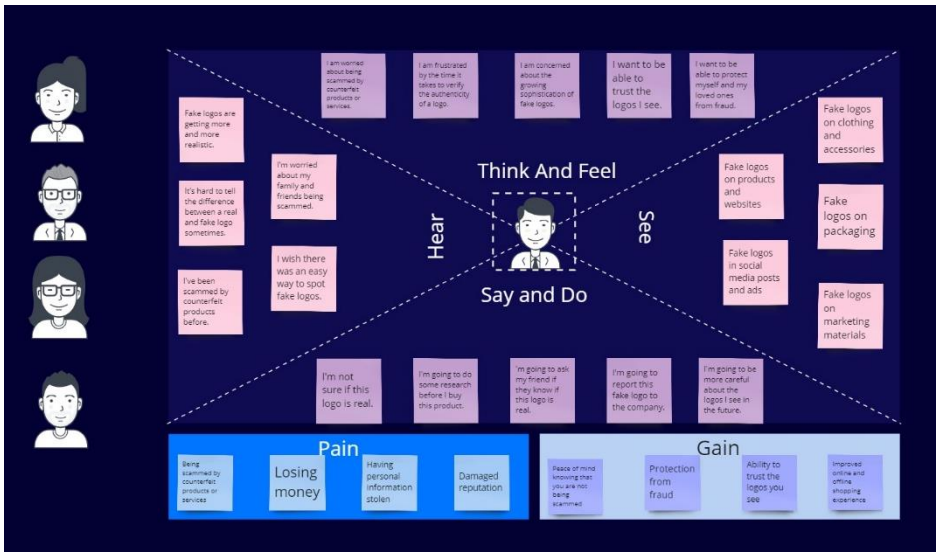
- **2.1 Existing Problem:**
  The digital space faces a significant challenge with the proliferation of counterfeit logos, leading to trust and authenticity issues. Our literature survey delves into existing problems related to logo authentication and explores the current landscape of solutions.

- **2.2 References:**
  The successful completion of our project was facilitated by consulting various online sources. GitHub served as a valuable resource for referencing and understanding coding aspects integral to our project. For a comprehensive range of datasets and to gain insights from public codes relevant to our project's objective, we utilized Kaggle.com. Additionally, Pinterest and Google Images were instrumental in inspiring front-end design ideas and collecting real and fake logos, respectively.

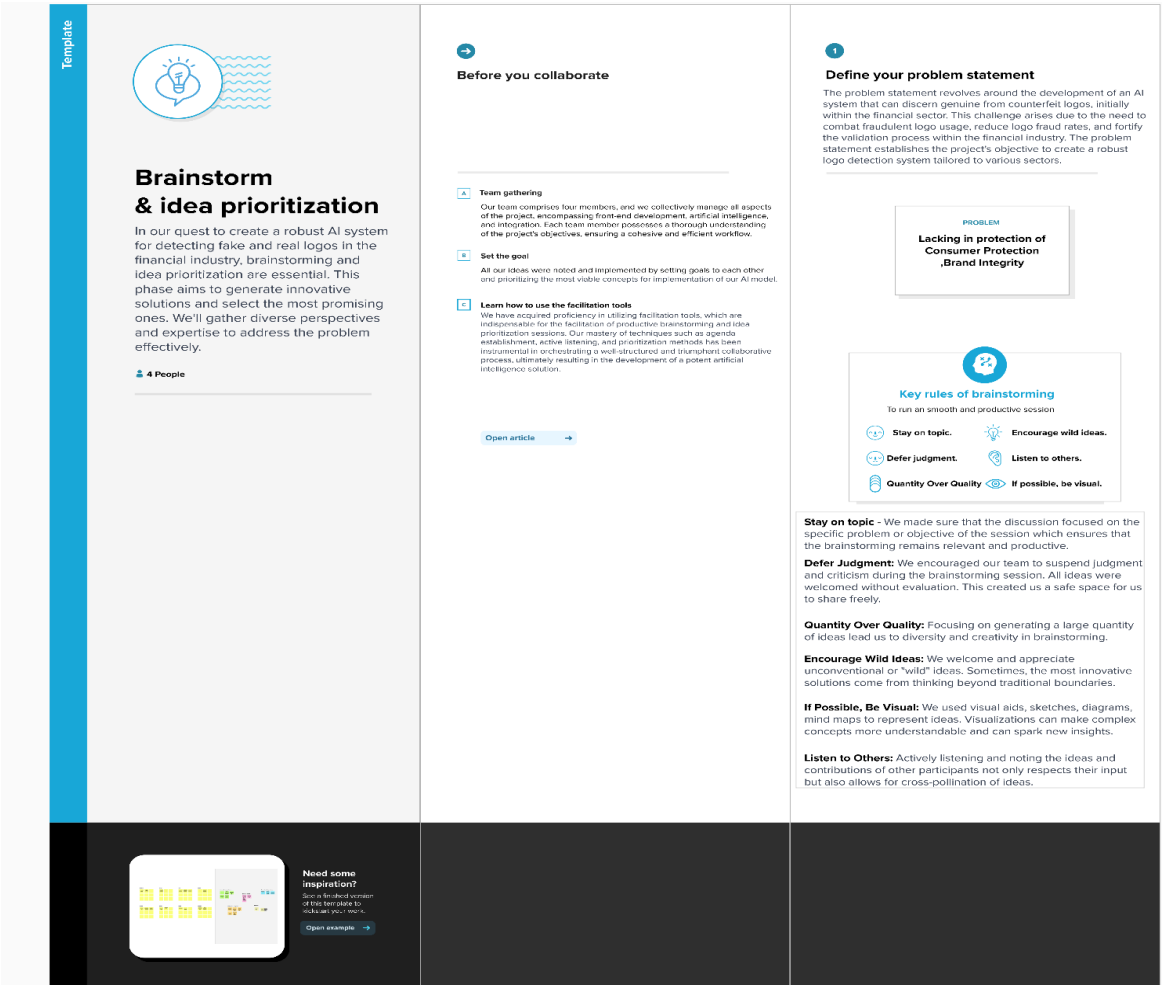
- **2.3 Problem Statement Definition:**
  The successful completion of our project involved extensive consultation of online resources. GitHub served as a valuable reference for coding aspects, providing insights and solutions crucial to our project. Kaggle.com played a pivotal role, offering a diverse range of datasets and serving as a resource for reviewing public codes relevant to our project's objectives. Pinterest and Google Images were instrumental in gathering front-end design ideas and a repository of both fake and real logos, respectively. To foster collaboration and idea generation, Mural.com was utilized for creating an empathy map and facilitating brainstorming sessions. Google Drive served as our central platform for organizing and storing files in a folder-wise structure. Additionally, GitHub was employed for the official upload of all project-related documents.

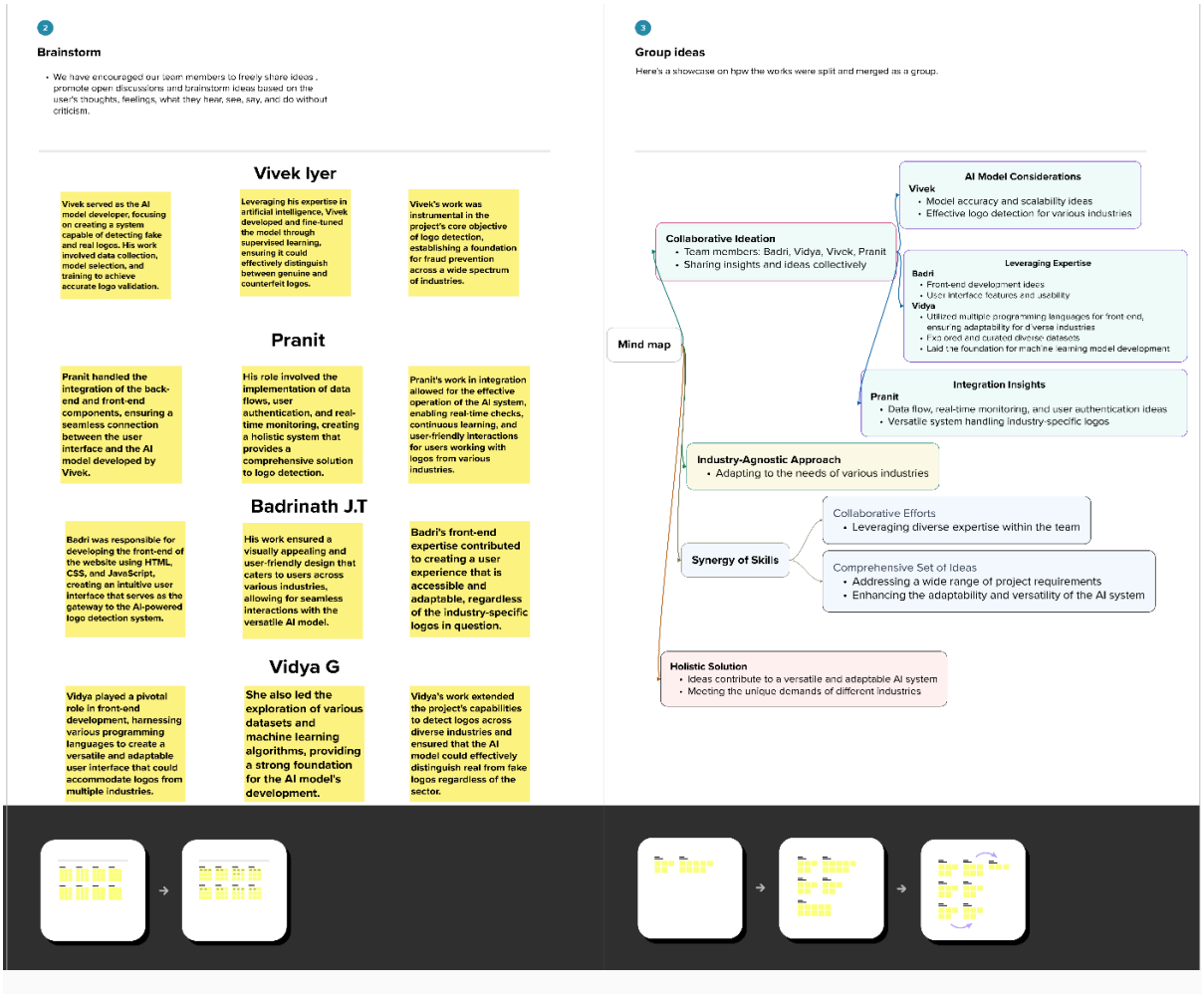
3. IDEATION & PROPOSED SOLUTION

- **3.1 Empathy Map Canvas:**



- **3.2 Ideation and Brainstorming:**





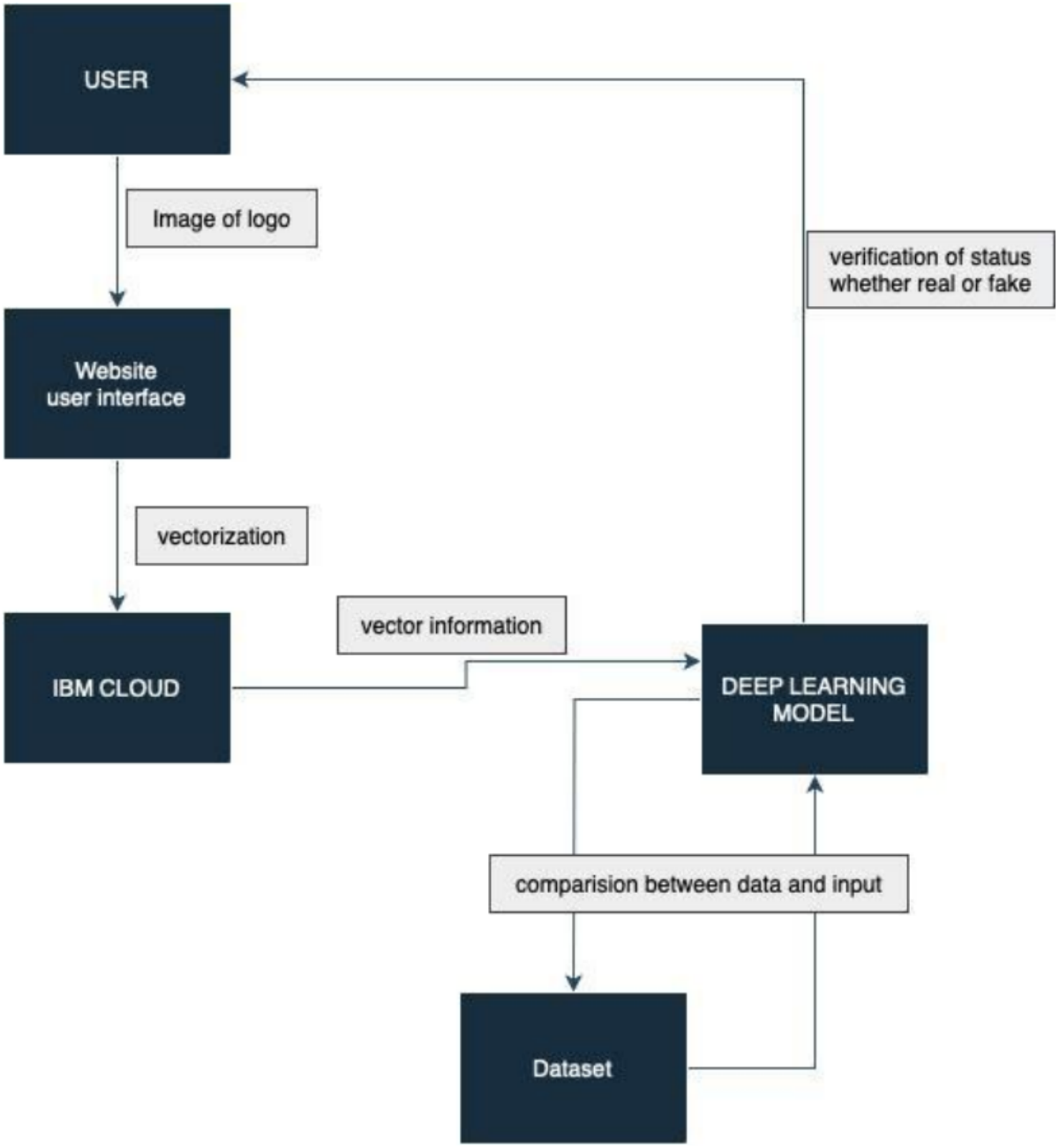
4. REQUIREMENT ANALYSIS

- **4.1 Functional requirement:**
  Functional requirements include the ability to detect fake and real logos, a user-friendly front-end interface, brand identification features, and seamless integration of the various project components.

- **4.2 Non-functional requirements:**
  Non-functional requirements encompass aspects like system reliability, scalability, and optimal memory utilization. Additionally, the system must adhere to industry-standard security protocols.

5. PROJECT DESIGN

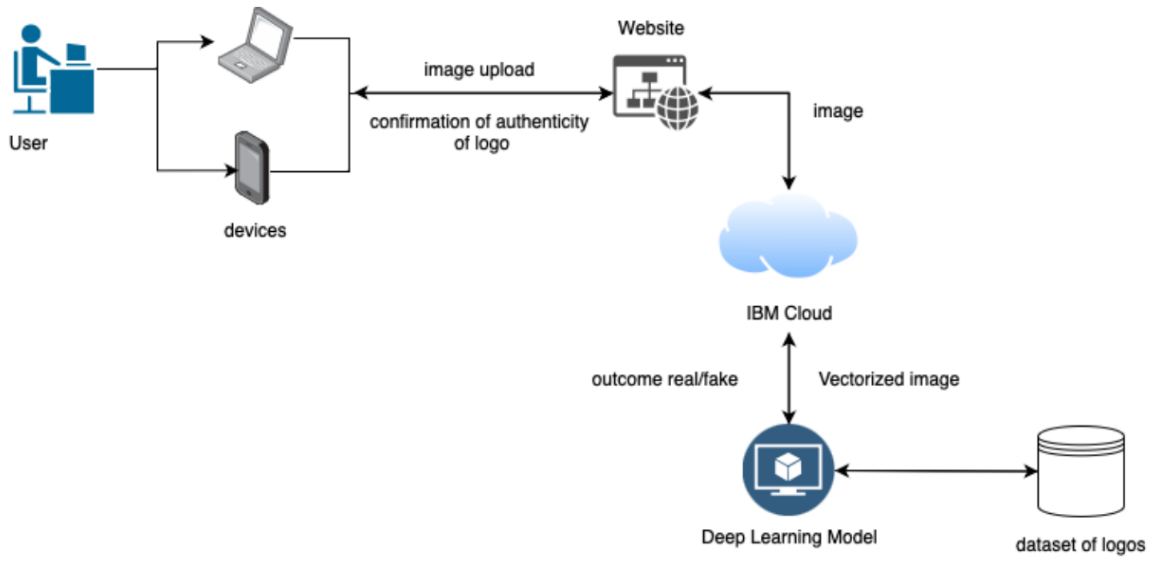
- **5.1 Data Flow Diagrams and User Stories:**


USER
Image of logo
Website
user interface
vectorization
IBM CLOUD
vector information
DEEP LEARNING
MODEL
verification of status
whether real or fake
comparision between data and input
Dataset

User Stories

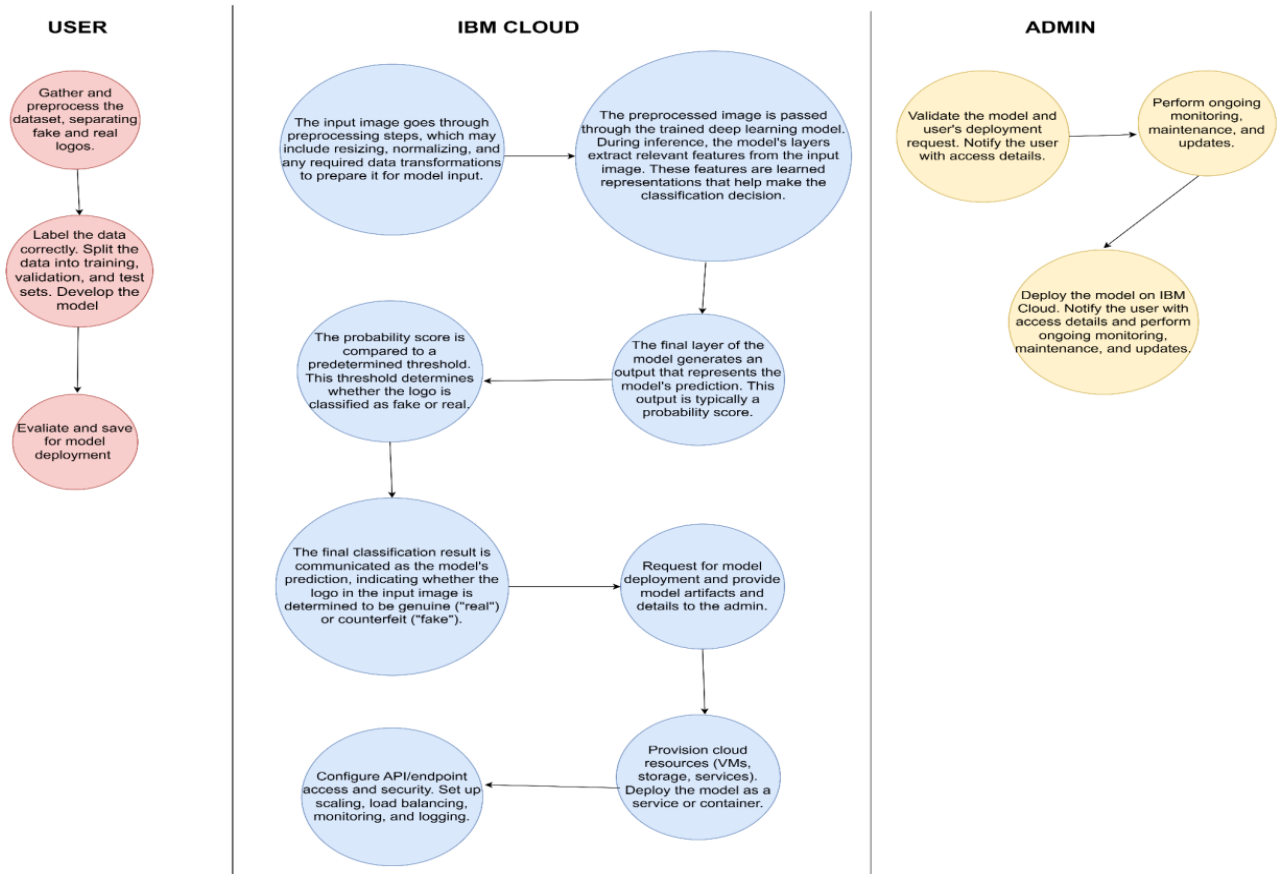
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
User (Web user)	Image Upload	USN-1	Upload a logo image to the website UI for verification.	Select & upload an image with acceptable formats.	High	Sprint-1
User (Web user)	Verification Feedback	USN-2	Receive feedback on the verification status of logo.	Get a response on logo authenticity after upload.	High	Sprint-2
System	AI Model Verification	USN-3	Use CNN neural network on IBM Cloud for verification.	Process image and provide verification output.	High	Sprint-3
Admin	Dataset Update	USN-4	Update dataset for verification.	Reflect updated data for verification after modification.	Medium	Sprint-4
System	Connection to IBM Cloud	USN-5	Connect to CNN neural network on IBM Cloud.	Send and process verification requests on IBM Cloud.	High	Sprint-5
User (Web user)	UI Feedback & Error Handling	USN-6	Receive clear error messages for issues.	Display clear error messages for unsupported formats.	Medium	Sprint-6
Admin	Monitoring & Logging	USN-7	View logs of verification attempts and errors.	Capture activities and errors in accessible logs.	Medium	Sprint-7
User (Web user)	Data Security & Privacy	USN-8	Assure secure processing and non-storage of logos.	Ensure no storage of user-uploaded images after verify.	High	Sprint-8
System	Scalability & Load Handling	USN-9	Be scalable for multiple simultaneous verifications.	Respond promptly under high load.	Medium	Sprint-9
Admin	System Health & Maintenance Dash.	USN-10	Monitor system health and issues in real-time.	Provide real-time insights through a dashboard.	Medium	Sprint-10

- **5.2 Solution Architecture:**



6. PROJECT PLANNING AND SCHEDULING

- **6.1 Technical Architecture:**



- **6.2&6.3 Sprint planning & estimation and delivery schedule**

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection and Preprocessing	USN-1	Gather and preprocess a dataset of fake and real logos for model training.	5	High	Vivek
Sprint-2	Model Development	USN-2	Design and develop a deep learning model for logo detection.	8	High	Vivek
Sprint-3	Model Training and Validation	USN-3	Train the deep learning model on the dataset and validate its performance.	5	Low	Vivek
Sprint-4	Website Front-End Design	USN-4	Create a user-friendly front-end interface for users to interact with the system.	5	Medium	Badri, Vidya
Sprint-5	Website Back-End Development	USN-5	Develop the back-end of the website to handle user requests and model integration.	8	High	Badri, Vidya
Sprint-6	Model Integration with Flask	USN-6	Integrate the trained model with the website using Flask for serving predictions..	5	High	Pranit
Sprint-7	User Interface Testing	USN-7	Perform user interface testing to ensure the website functions correctly and is user-friendly.	5	Medium	Vivek, Badri, Vidya, Pranit
Sprint-8	System Testing and Optimization	USN-8	Conduct system testing and optimization to enhance performance and reliability.	8	High	Vivek, Badri, Vidya, Pranit
Sprint-9	Documentation	USN-9	Create comprehensive project documentation, including user guides and technical documentation.	3	Low	Vivek, Badri,

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	1 Days	27 Oct 2023	28 Oct 2023	20	27 Oct 2023
Sprint-2	20	2 Days	29 Oct 2023	31 Oct 2023	20	29 Oct 2023
Sprint-3	20	1 Days	01 Nov 2023	02 Nov 2023	20	01 Nov 2023
Sprint-4	20	2 Days	03 Nov 2023	05 Nov 2023	20	03 Nov 2023
Sprint-5	20	2 Days	06 Nov 2023	08 Nov 2023	20	06 Nov 2023
Sprint-6	20	2 Days	09 Nov 2023	11 Nov 2023	20	09 Nov 2023
Sprint-7	20	1 Days	14 Nov 2023	15 Nov 2023	20	14 Nov 2023
Sprint-8	20	1 Days	16 Nov 2023	17 Nov 2023	20	16 Nov 2023
Sprint-9	20	1 Days	18 Nov 2023	19 Nov 2023	20	18 Nov 2023

We have a 17-day sprint duration, and the velocity of the team is 20 (points per sprint). The team's average velocity (AV) per iteration unit (story points per day)

Average Velocity (AV) = sprint duration/velocity = 20/17 = 1.176

7. CODING & SOLUTION

- **7.1 Feature 1 – Detecting the authenticity of the logos:**
  One key feature is the accurate detection of fake and real logos using state-of-the-art deep learning algorithms. Vivek, our machine learning developer, played a pivotal role in fine-tuning these algorithms for optimal performance which was integrated by Pranit, with the front-end made my Vidya and Badri.

```
import os
import cv2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
import tensorflow as tf
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.mobilenet import preprocess_input
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Input, Flatten, Dense, GlobalAveragePooling2D
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
import pandas as pd

# Load the CSV file
df = pd.read_csv('/kaggle/input/fakereal-logo-detection-dataset/file_mapping.csv')

# Display the columns to check if 'Unnamed: 0' is present
print("Columns before dropping 'Unnamed: 0':", df.columns)

# Drop 'Unnamed: 0' if it exists
if 'Unnamed: 0' in df.columns:
    df.drop('Unnamed: 0', axis=1, inplace=True)
    print("Column 'Unnamed: 0' dropped.")
else:
    print("Column 'Unnamed: 0' not found in DataFrame.")

# Display the first few rows of the DataFrame
df.head()
```

```
# Preprocess the image
label_encoder = preprocessing.LabelEncoder()
df['Label'] = label_encoder.fit_transform(df['Label'])
```

```
img = cv2.imread(os.path.join('/kaggle/input/fakereal-logo-detection-dataset', df['Filename'][6].replace('\\', '/')), cv2.COLOR_RGB2BGR)
plt.imshow(img)
```

```
IMG_SIZE = 70
images = []
labels = []

for _, row in df.iterrows():
    img = cv2.imread(os.path.join('/kaggle/input/fakereal-logo-detection-dataset',
                                  row['Filename'].replace('\\', '/')), )
    try:
        img_arr = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        image = img_to_array(img_arr)
        image = preprocess_input(image)
        images.append(image)
        labels.append(row['Label'])
    except:
        continue
image_shape = images[0].shape

for image in images:
    if image.shape != image_shape:
        raise ValueError("All images must have the same shape.")

images = np.array(images)
labels = np.array(labels)
```

```
# Normalize pixel values to the range [0, 1]
images = np.array(images) / 255.0
```

```
# Split the dataset into train and test sets

train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size=0.2, random_state=42)
test_labels
```

```
# Build the model
base_model = tf.keras.applications.MobileNet(input_shape=(IMG_SIZE, IMG_SIZE, 3), include_top=False, weights='imagenet')
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(128, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=output)
```

```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

```
# Train the model with EarlyStopping
early_stopping = EarlyStopping(monitor="val_loss", patience=5, verbose=1, restore_best_weights=True)
model.fit(train_images, train_labels, batch_size=32, epochs=10, validation_data=(test_images, test_labels), callbacks=[early_stopping])
```

```
# Evaluate the model
_, accuracy = model.evaluate(test_images, test_labels)
print('Test Accuracy:', accuracy)
```

```python
# Predictions
predictions = model.predict(test_images)
y_pred = (predictions > 0.5).astype(int).reshape(-1)

print(classification_report(y_pred, test_labels))

image_path="/kaggle/input/bmw-image"
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    try:
        img_arr = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        image = img_to_array(img_arr)
        image = preprocess_input(image)
        return image
    except Exception as e:
        print(f"Error preprocessing image: {e}")
        return None

def predict_image(model, image_path):
    # Preprocess the image
    image = preprocess_image(image_path)
    if image is not None:
        # Expand dimensions to match the model's expected input shape
        image = np.expand_dims(image, axis=0)

        # Make prediction
        prediction = model.predict(image)

        # Convert prediction to binary (0 or 1)
        predicted_class = int(prediction > 0.5)

        return predicted_class
    else:
        return None
```

```python
# Test the model with user input
user_image_path = input("Enter the path of the image to test: ")
predicted_class = predict_image(model, user_image_path)

if predicted_class is not None:
    if predicted_class == 1:
        print("The model predicts the image is Genuine.")
    else:
        print("The model predicts the image is Fake.")
else:
    print("Error processing the image.")
```

This Python script utilizes TensorFlow and OpenCV to build and train a MobileNet-based model for detecting fake and real logos. Below is a step-by-step explanation of the code:

1. Import Libraries:

- The script starts by importing necessary libraries, including TensorFlow, OpenCV, and other essential modules for data processing and visualization.

2. Load and Preprocess Dataset:

- The script loads a dataset from a CSV file (`file_mapping.csv`) that contains file mappings and labels. It preprocesses the images, resizes them to a specified dimension, and normalizes pixel values.

3. Build and Train the Model:

- A MobileNet model is built with a custom output layer for binary classification (fake or genuine logos). The model is compiled, and training is executed using the dataset. The training process includes early stopping to prevent overfitting.

4. Evaluate the Model:

- The trained model is evaluated on the test dataset, and the accuracy is printed. Additionally, a classification report is generated, providing detailed metrics such as precision, recall, and F1-score.

5. Predictions on User Input:

- The script defines functions (`preprocess_image` and `predict_image`) to preprocess an input image and make predictions using the trained model. It prompts the user to input the path of an image for testing and prints whether the model predicts it as genuine or fake.

The script is organized into sections, each serving a specific purpose, from data preprocessing to model training and user testing. It effectively demonstrates the process of building, training, and using a machine learning model for logo detection.

- **7.2 Feature 2- Identifying the logos:**
  An additional feature involves the identification of the brand associated with logos in uploaded images. Vidya, with her expertise in front-end and integration, ensured a seamless user experience for this feature.

```
from flask import Flask, render_template, request
from keras.models import load_model
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input
import numpy as np

app = Flask(__name__)

# Load the pre-trained model
VGG19_model = load_model('fakelogo.h5')

# Define class names
class_names = [
    'Adidas', 'Amazon', 'Android', 'Apple', 'Ariel', 'BMW', 'Bic', 'Burger King',
    'Cadbury', 'Chevrolet', 'Chrome', 'Coca Cola', 'Cowbell', 'Dominos', 'Fila',
    'Gillette', 'Google', 'Goya oil', 'Guinness', 'Heinz', 'Honda', 'Hp', 'Huawei',
    'Instagram', 'Kfc', 'Krisspy Kreme', 'Lays', 'Levis', 'Lg', 'Lipton', 'Mars', 'Marvel', 'McDonald',
    'Mercedes Benz', 'Microsoft', 'MnM', 'Mtn', 'Mtn dew', 'NASA', 'Nescafe', 'Nestle', 'Nestle milo',
    'Netflix', 'Nike', 'Nutella', 'Oral b', 'Oreo', 'Pay pal', 'Peak milk', 'Pepsi', 'PlayStation',
    'Pringles', 'Puma', 'Reebok', 'Rolex', 'Samsung', 'Sprite', 'Starbucks', 'Tesla', 'Tiktok',
    'Twitter', 'YouTube', 'Zara'
]

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # Get the uploaded image file
        file = request.files['file']
        if file:
            # Save the uploaded image
            img_path = 'uploads/uploaded_image.png'
            file.save(img_path)

            # Preprocess the image
            img = image.load_img(img_path, target_size=(244, 244))
            x = image.img_to_array(img)
            x = np.expand_dims(x, axis=0)
            x = preprocess_input(x)

            # Make a prediction
            preds = VGG19_model.predict(x)

            # Get the predicted class
            predicted_class = class_names[np.argmax(preds)]

            return render_template('index.html', prediction=predicted_class, image_path=img_path)

    return render_template('index.html', prediction=None, image_path=None)

if __name__ == '__main__':
    app.run(debug=True)
```

This Python code represents a Flask web application for logo detection using a pre-trained VGG19 model. Here is a simple summary of the code:

1. Import Libraries:
- The code begins by importing necessary libraries, including Flask for web development and Keras for deep learning.

2. Load Pre-trained Model:
- The pre-trained VGG19 model for logo detection (`fakelogo.h5`) is loaded. VGG19 is a popular deep learning model known for image classification.

3. Define Class Names:
- A list of class names is defined, representing various brand logos that the model can recognize.

4. Flask Web App Setup:
- A Flask web application is created using the `Flask` class.

5. Define Route for Image Upload:
- The `/` route is defined to handle both GET and POST requests. On POST, it receives an uploaded image, saves it, preprocesses it for the VGG19 model, and makes a prediction.

6. Prediction and Display:
- The model predicts the logo in the uploaded image, and the predicted class (brand name) is displayed on the web page. The processed image and prediction are rendered using the `render_template` function.

7. Run the Application:
- The application runs when the script is executed directly (`if __name__ == '__main__':`). The `debug=True` parameter enables debugging during development.

In essence, this Flask app allows users to upload an image containing a brand logo. The VGG19 model then processes the image, predicts the brand, and displays the result on a web page. The code integrates image handling, model prediction, and web development to create an interactive logo detection tool.

8. PERFORMANCE TESTING

- **8.1 Performance metrics:**
  - ***Model Compilation***: In the model compilation step, the script uses the following metrics:
    - ❖ Loss Metric: Binary Crossentropy ('binary_crossentropy')
    - ❖ Accuracy Metric: ‘accuracy’

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
```

  - ***Model Evaluation:*** After training , the script evaluates the model on the dataset. The evaluation includes the accuracy metric, which is printed.

```
_, accuracy = model.evaluate(test_images, test_labels)
print('Test Accuracy:', accuracy)
```

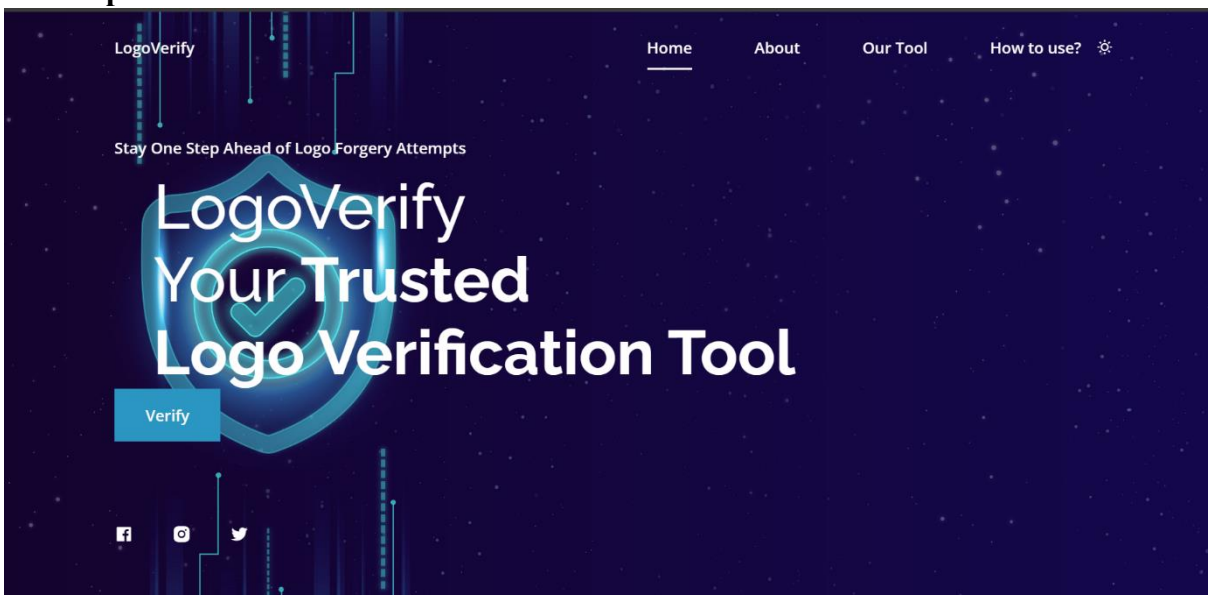
  - ***Classification Report:*** The script generates a classification report, providing detailed metrics such as precision, recall and F1-score. This report is based on the predictions made by the model on the test dataset.

```
predictions = model.predict(test_images)
y_pred = (predictions > 0.5).astype(int).reshape(-1)
print(classification_report(y_pred, test_labels))
```

These are the key metrics used in our codes. The accuracy is a common metric for classification tasks, and the classification report provides a more comprehensive view, including precision, recall, and F1-score.

9. RESULTS

- **9.1 Output Screenshots:** Screenshots from our website



Logo Classifier

Choose File No file chosen

SUBMIT

Prediction: Genuine

Logo Classifier

Choose File No file chosen

SUBMIT

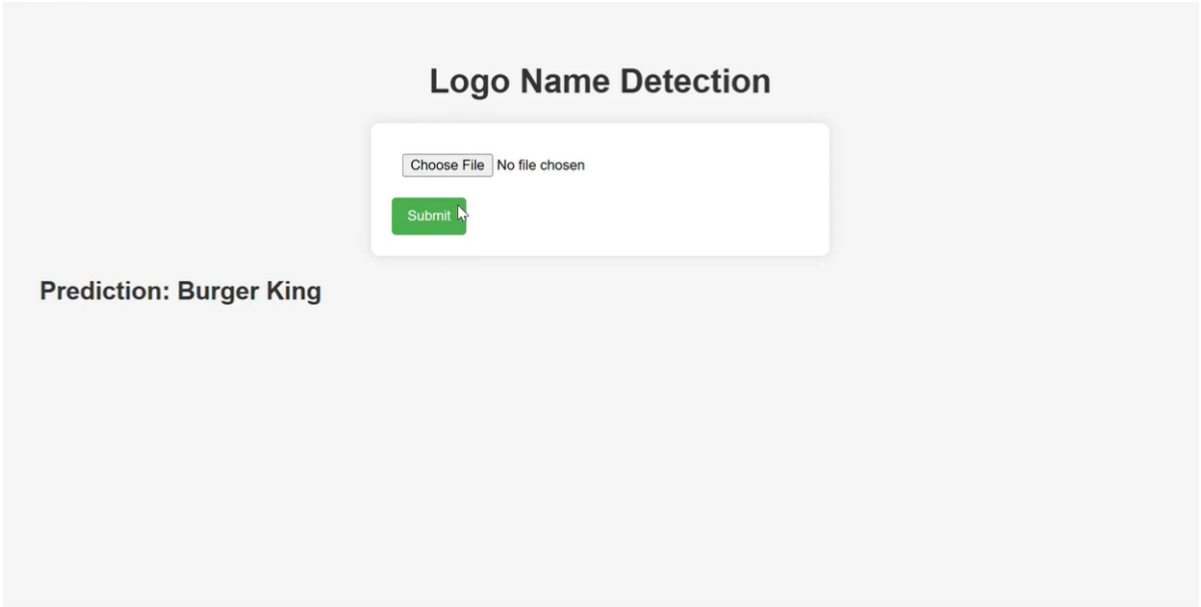
Prediction: Fake

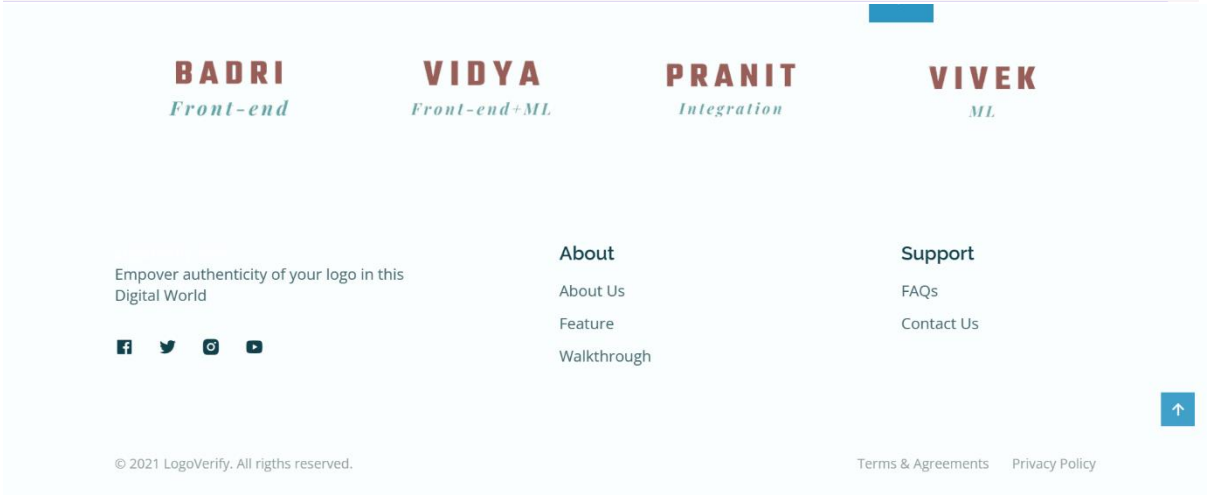
Logo Name Detection

Choose File burgerking.jpg

Submit

Selected Image:





10. ADVANTAGES & DISADVANTAGES

- **Advantages:**
    a. Accurate logo detection.
    b. Enhanced user experience with brand identification.
    c. Scalable and maintainable architecture.

- **Disadvantages:**
    a. Potential dependency on robust internet connectivity for real-time processing.

11. CONCLUSION

In conclusion, our project successfully addresses the challenge of counterfeit logos by combining front-end development and machine learning. The comprehensive solution not only enhances logo authentication but also provides valuable insights into associated brands.

12. FUTURE SCOPE

- The future scope of our project includes:
    1. Refinement of machine learning models for even greater accuracy.
    2. Integration of additional features based on user feedback.
    3. Collaboration with industry stakeholders for real-world application and validation.

13. APPENDIX

- **Source code –**
  https://drive.google.com/drive/folders/1tIw8aSL_STutb2BoatFhsm5AlWLm62cw?usp=sharing

- **GitHub link –**
  https://github.com/smartinternz02/SI-GuidedProject-600227-1698070743

- **Project Demo Link –**
  https://clipchamp.com/watch/qNf18aoxvg8