**ASSESSING THE SAFETY OF MUNICIPAL DRNKING WATER**

**PROJECT REPORT**

**Submitted by**

**Team Id NM2023TMID10894**

**MUTHUBHUVANESWARI**

**RANJITHA P**

**JANSI K**

**JOTHI S**

**CONTENT :**

1. **INTRODUCTION**

1.1 Project Overview

1.2 Purpose

2. **IDEATION & PROPOSED SOLUTION**

2.1 Problem Statement Definition

2.2 Empathy Map Canvas

2.3 Ideation & Brainstorming

2.4 Proposed Solution

3. **REQUIREMENT ANALYSIS**

3.1 Functional requirement

3.2 Non-Functional requirements

**4. PROJECT DESIGN**

4.1 Data Flow Diagrams

4.2 Solution & Technical Architecture

4.3 User Stories

**5. CODING & SOLUTIONING (Explain the features added in the project along with code)**

5.1 Feature 1

5.2 Feature 2

**6. RESULTS**

6.1 Performance Metrics

**7. ADVANTAGES & DISADVANTAGES**

**8. CONCLUSION**

**9. FUTURE SCOPE**

**10. APPENDIX**

Source Code

GitHub & Project Video Demo Link

1. INTRODUCTION
   1. PROJECT OVERVIEW

Assessing the safety of a municipal drinking water project involves evaluating various aspects to ensure that the water provided to the community is of high quality and meets health and safety standards. Here's an overview of the key factors to consider in such an assessment:

1. Water Source: Identify the source of the drinking water, whether it comes from surface water (rivers, lakes) or groundwater (wells, aquifers). Evaluate the quality and potential contaminants associated with the chosen water source.

2. Treatment Process: Assess the treatment process employed to purify the water and make it safe for consumption. This may involve processes such as coagulation, sedimentation, filtration, disinfection (chlorination, UV treatment), and possibly advanced treatments like reverse osmosis or activated carbon filtration.

3. Infrastructure: Evaluate the infrastructure and equipment used in the water treatment and distribution system. This includes examining the condition of pipes, pumps, storage tanks, and any other components that may impact water quality. Ensure that the infrastructure is well-maintained and capable of delivering clean water to the community.

4. Compliance with Regulations: Verify that the project adheres to local, national, and international regulations and standards for drinking water quality. These standards may include limits for various contaminants such as bacteria, viruses, chemicals, heavy metals, and disinfection by-products.

5. Monitoring and Testing: Establish a robust system for monitoring and testing the water quality at various stages of the treatment process and distribution network. Regular sampling and analysis should be conducted to identify any potential issues or deviations from the desired water quality standards.

6. Emergency Preparedness: Develop contingency plans and protocols to address potential emergencies or incidents that may impact the water supply. This includes having backup systems, alternative water sources, and response plans for natural disasters, equipment failures, or waterborne disease outbreaks.

7. Community Engagement: Involve the local community and stakeholders in the assessment process. Seek their input, address concerns, and provide transparent information about the safety measures in place. Encourage public participation and awareness to ensure community trust and confidence in the drinking water project.

8. Long-Term Sustainability: Consider the long-term sustainability of the project, including factors like water resource management, energy efficiency, waste management, and environmental impact. Ensure that the project is designed to meet the community's needs for the foreseeable future.

9. Regular Review and Upgrades: Implement a system for periodic review and evaluation of the drinking water project. This includes assessing new technologies, scientific advancements, and regulatory changes to continuously improve the safety and quality of the drinking water provided to the community.

Overall, assessing the safety of a municipal drinking water project requires a comprehensive approach that encompasses water source evaluation, treatment processes, infrastructure, compliance, monitoring, emergency preparedness, community engagement, sustainability, and continuous improvement. By addressing these aspects, the project can provide safe and clean drinking water to the community.

* 1. PURPOSE

The purpose of assessing the safety of a municipal drinking water project is to ensure that the water provided to the community meets the required standards and is safe for consumption. This assessment is crucial to protect public health and prevent waterborne diseases.

Here are some specific purposes for assessing the safety of a municipal drinking water project:

1. Public Health Protection: The primary purpose is to safeguard the health and well-being of the community by ensuring that the water supplied meets the necessary quality and safety standards. This involves assessing the presence of harmful contaminants, such as bacteria, viruses, chemicals, heavy metals, pesticides, and other substances that could pose health risks when consumed.

2. Compliance with Regulations: Municipal drinking water projects must comply with local, national, and international regulations and guidelines. Assessing the safety of the project ensures that it meets all the regulatory requirements related to water quality, treatment processes, distribution systems, and monitoring protocols.

3. Risk Identification and Mitigation: The assessment process helps identify potential risks and hazards that may impact the safety of the drinking water supply. It involves evaluating the water sources, treatment methods, infrastructure, and operational procedures to identify vulnerabilities and implement appropriate mitigation measures.

4. Water Treatment Evaluation: Assessing the safety of the municipal drinking water project involves evaluating the effectiveness of the water treatment processes employed. This includes assessing the efficiency of filtration, disinfection, and other treatment methods in removing or reducing contaminants to acceptable levels.

5. Monitoring and Surveillance: Regular monitoring and surveillance of the water supply are essential to ensure ongoing safety. The assessment process establishes protocols for routine testing, sampling, and analysis of water samples to detect any deviations from the desired standards. It helps identify trends, detect potential issues, and take corrective actions promptly.

6. Emergency Preparedness and Response: The assessment also includes evaluating the preparedness and response plans in case of emergencies or incidents that could impact the safety of the drinking water supply. This involves assessing the infrastructure's resilience, contingency plans, and communication protocols to ensure timely and effective response to any water-related emergencies.

Overall, the purpose of assessing the safety of a municipal drinking water project is to protect public health, comply with regulations, identify and mitigate risks, ensure effective treatment processes, maintain continuous monitoring, and be prepared for emergencies.

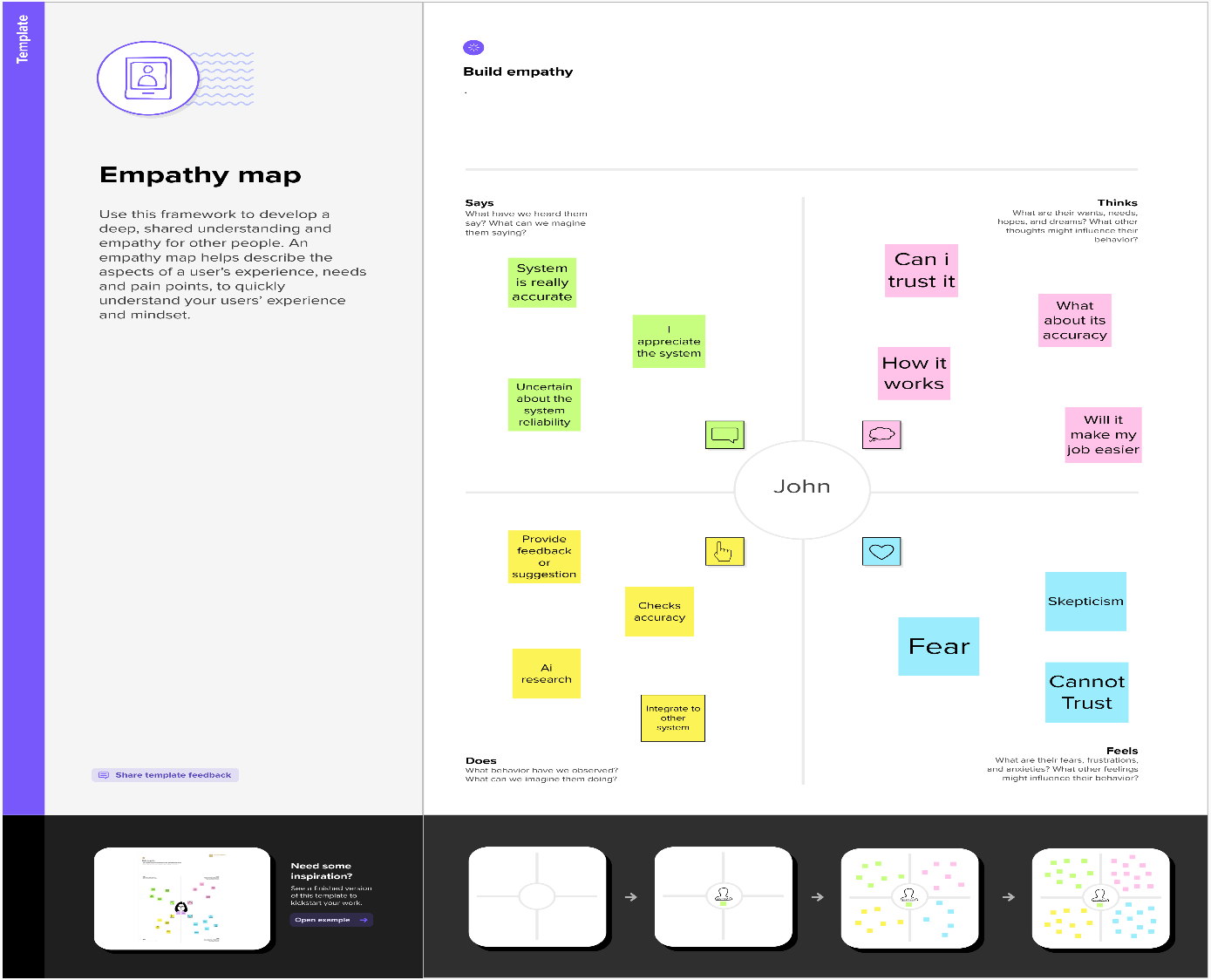
1. IDEATION & PROPOSED SOLUTION
   1. PROBLEM STATEMENT DEFENITION

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.



* 1. EMPATHY MAP

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.

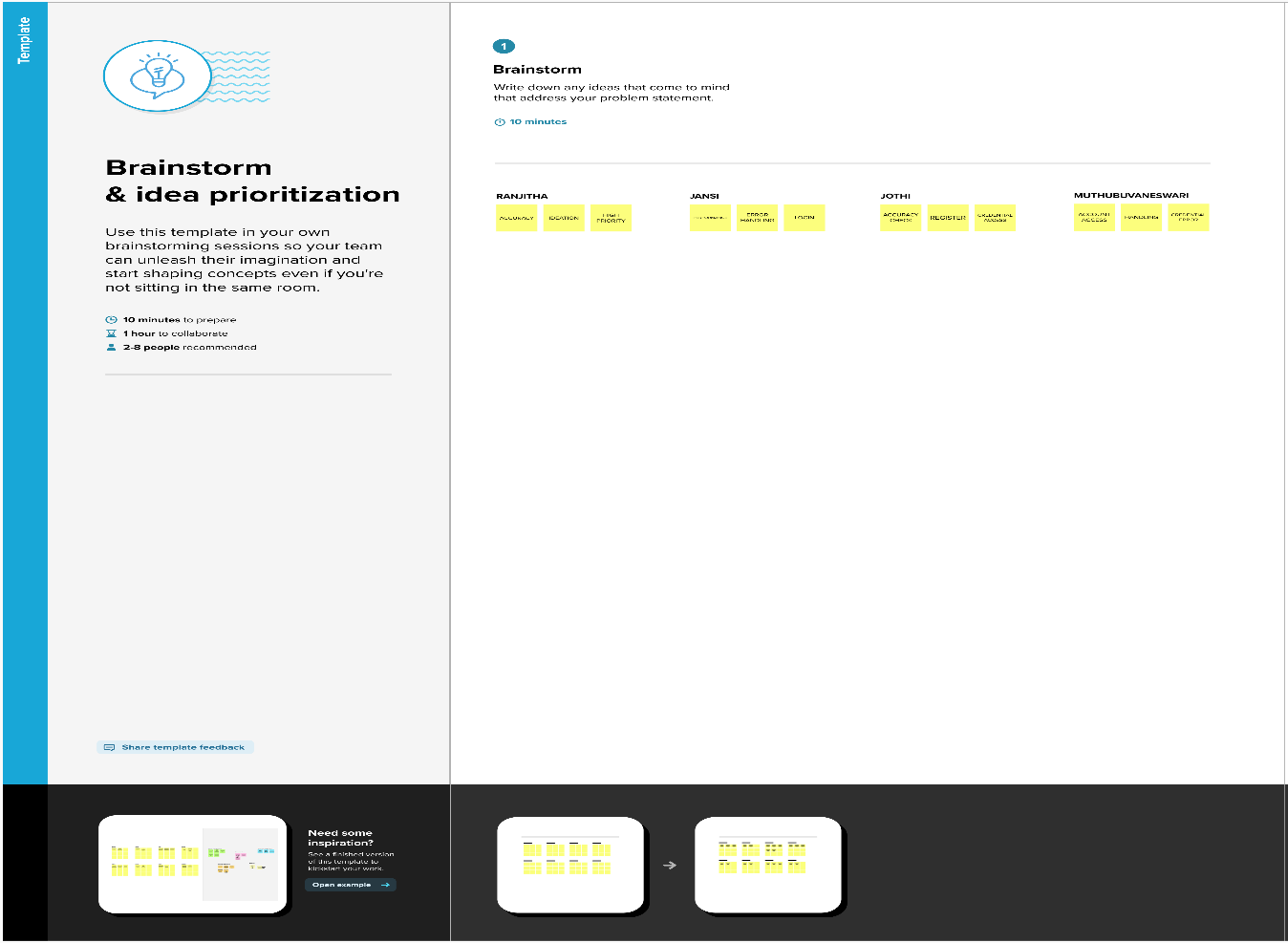


* 1. BRAINSTORM & IDEATION

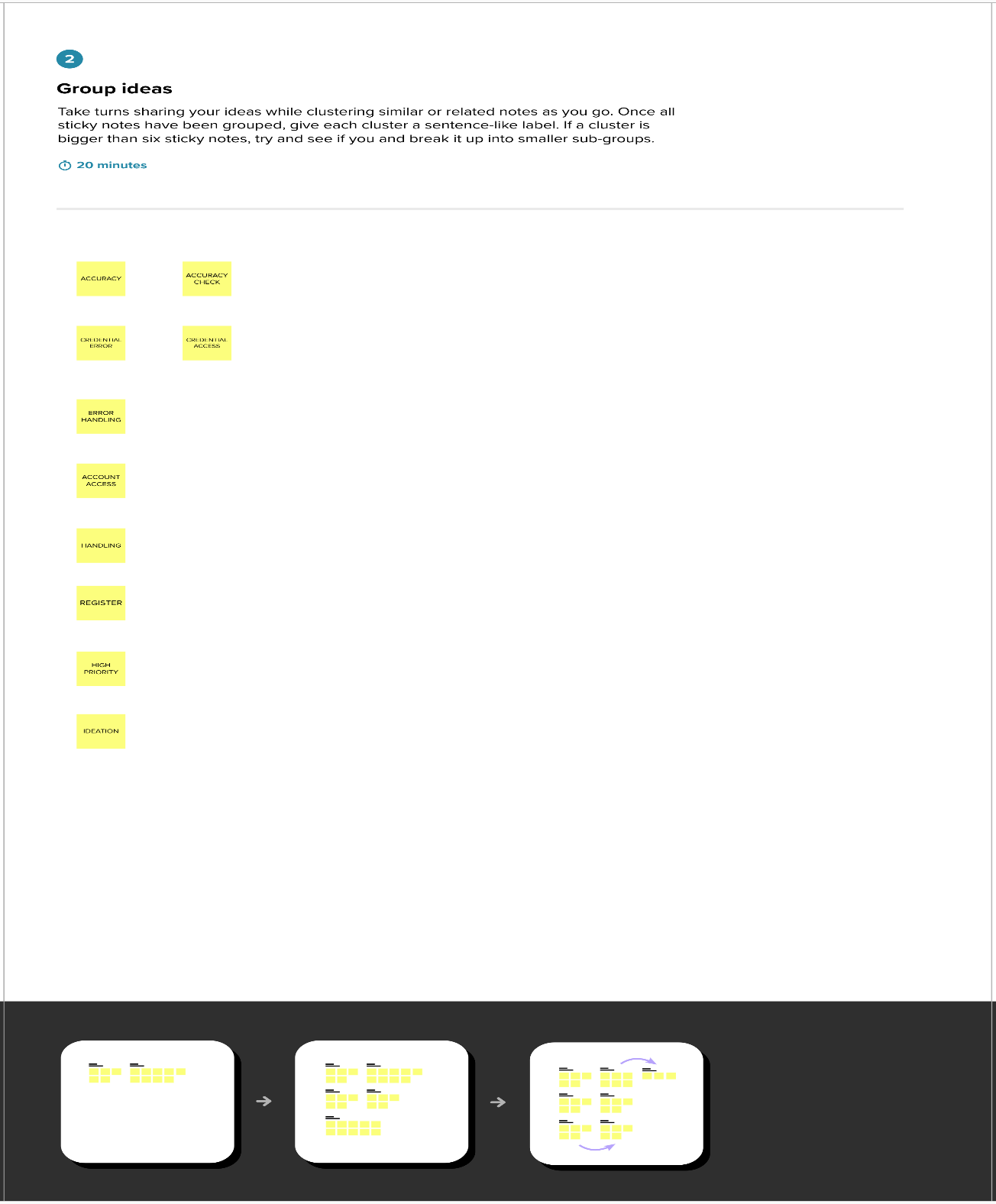
**Brainstorm & Idea Prioritization :**

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.

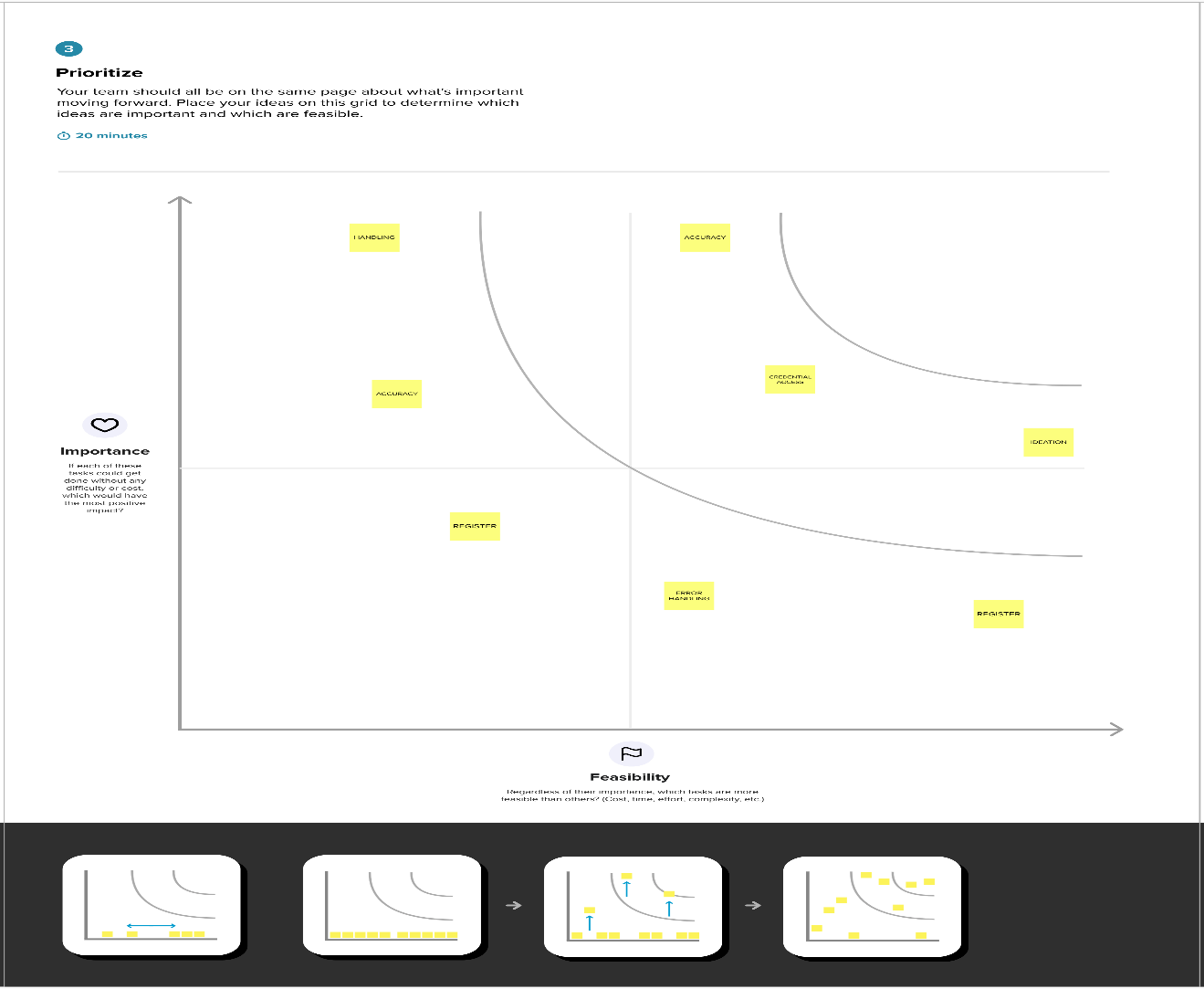
**Step-1: Team Gathering, Collaboration and Select the Problem Statement**



**Step-2: Brainstorm, Grouping**



**Step-3: Idea Prioritization**



* 1. PROPOSED SYSTEM

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | The problem is the lack of an efficient and comprehensive system for assessing the safety of municipal drinking water. Current methods are often manual, time-consuming, and prone to errors, leading to potential health risks for the population |
|  | Idea / Solution description | The system will continuously monitor key parameters such as pH levels, chemical contaminants, bacterial presence, and turbidity. |
|  | Novelty / Uniqueness | Our solution stands out due to its integration of IoT sensors, data analytics, and machine learning algorithms to provide a holistic and automated approach to water quality assessment. The real-time monitoring, predictive analytics, and automated alert system offer a unique advantage over traditional manual methods |
|  | Social Impact / Customer Satisfaction | implementing this solution, we aim to significantly improve the safety of municipal drinking water. Real-time monitoring and early detection of water quality issues will enable prompt action, minimizing health risks and ensuring public safety. |
|  | Business Model (Revenue Model) | business model revolves around a subscription-based service. Municipalities and water authorities can subscribe to our system and pay a recurring fee based on the size of their water distribution network and the number of sensors deployed. |
|  | Scalability of the Solution | The proposed solution is highly scalable. It can accommodate water distribution networks of varying sizes, from small towns to large cities. The system can easily be expanded by deploying additional sensors and scaling up the data management infrastructure. |

3 REQUIREMENT ANALYSIS

# Functional Requirements:

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | Water Quality Testing | contaminants, including bacteria, viruses, parasites, heavy metals |
| FR-2 | Monitoring and Surveillance | Install appropriate sensors, meters, and data logging systems to collect and analyze |
| FR-3 | Infrastructure and Treatment | water treatment plants, distribution networks, and storage facilities |
| FR-4 | Emergency Preparedness and Response | authorities, stakeholders, and the public |
| FR-5 | Public Education and Outreach | test results, water treatment processes |
| FR-6 | Compliance and Reporting | water quality tests, monitoring data, maintenance activities |

Non – Functional Requirenments

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | technical and non-technical personnel involved in assessing the safety |
| NFR-2 | **Security** | confidentiality, integrity, and availability of the information. |
| NFR-3 | **Reliability** | detect and handle errors, ensuring minimal downtime |
| NFR-4 | **Performance** | quick responses to queries and generate reports within acceptable timeframes |
| NFR-5 | **Availability** | assess the safety of municipal drinking water |
| NFR-6 | **Scalability** | easy expansion and resource allocation, ensuring optimal performance |

4 PROJECT DESIGN

4.1 DATA FLOW DIAGRAM

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 right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Diagram

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional**  **Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Team Member** |
| Weather Analyst | Weather Classification | USN-1 | The system should be able to classify weather conditions based on input data using transfer learning. | The classification results in a clear and understandable format | High | ArjunYuvanesh |
| Our-door event organizer |  | USN-2 | As an outdoor event organizer, I want to be able to predict weather conditions for my upcoming event so that I can make informed decisions about scheduling, venue setup, and other logistical considerations. | Available in all the time | High | Revanthramesh |
| Emergency Management Team |  | USN-3 | I want to be able to identify potential natural disasters before they occur so that we can take appropriate action to mitigate the risks and protect the public.so the system should be able to classify weather conditions based on input image. | Performance and accuracy | High | Dhanuskar |
| Agricultural Scientist |  | USN-4 | As an agricultural scientist, I want to be able to predict weather patterns and their impact on crops so that I can optimize crop yield and minimize losses due to weather-related factors. | Efficient Classification | High | Sachin |
| Web user | User Friendly | USN-5 | I can easily access the application using my web browsers in all devices. | User Friendly | Medium | Rajapandian |
| Mobile User | Easy Access | USN-6 | I can access the service using a mobile application | Easy Access | Low | Sachin |
| Administrator | Update and monitoring | USN-7 | I can update a model and monitor its performance. | Monitoring and enhancement | Medium | Dhanuskar |

4.2 USER STORIES

5. CODING & SOLUTIONING

5.1 Model Training Code Explanation

Training code :

from google.colab import drive

drive.mount("/content/drive")

!pip install tensorflow

from tensorflow.keras.applications.vgg19 import VGG19, preprocess\_input

from tensorflow.keras.layers import Flatten, Dense

from tensorflow.keras.models import Model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from PIL import ImageFile

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

train\_datagen = ImageDataGenerator( rescale = 1./255,

shear\_range = 0.2,

zoom\_range = [.99,1.01],

brightness\_range = [0.8,1.2],

data\_format = "channels\_last",

fill\_mode ="constant",

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale=1./255)

training\_set = train\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM/dataset/train',

target\_size = (180, 180), batch\_size = 64,

class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM/dataset/test',

target\_size = (180, 180), batch\_size = 64,

class\_mode = 'categorical')

IMAGE\_SIZE =[180,180]

VGG19 = VGG19(input\_shape=IMAGE\_SIZE + [3], weights='imagenet',include\_top=False)

for layer in VGG19.layers:

layer.trainable = False

x = Flatten() (VGG19.output)

prediction = Dense(5, activation='softmax')(x)

model = Model(inputs=VGG19.input, outputs=prediction)

model.summary()

model.compile(

loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'] )

r = model.fit(

training\_set,

validation\_data = test\_set,

epochs = 50,

steps\_per\_epoch=len(training\_set),

validation\_steps = len(test\_set)

)

loss, accuracy = model.evaluate(test\_set,

steps=11,

verbose=2,

use\_multiprocessing=True,

workers=2)

print(f'Model performance on test images:\nAccuracy = {accuracy}\nLoss = {loss}')

model.save('wcv.h5')

model = load\_model("/content/wcv.h5")

img = image.load\_img(r"/content/drive/MyDrive/IBM/dataset/train/sunrise/sunrise1.jpg",target\_size= (180,180))

x = image.img\_to\_array(img)

x = np.expand\_dims(x,axis = 0)

preds=model.predict(x)

pred=np.argmax(preds,axis=1)

index=['cloudy','foggy','rainy','shine','sunrise']

result=str(index[pred[0]])

result

Explanation :

1. Mounting Google Drive: It mounts the Google Drive to the Colab environment using the `drive.mount` function. This allows access to files stored in Google Drive.

2. Installing TensorFlow: It installs the TensorFlow library using the `!pip install tensorflow` command.

3. Data Preparation:

- The code sets up image data generators (`train\_datagen` and `test\_datagen`) for data augmentation and rescaling.

- The training and test datasets are generated using `flow\_from\_directory` from the `ImageDataGenerator` class. The data is loaded from the specified directories (`'/content/drive/MyDrive/IBM/dataset/train'` and `'/content/drive/MyDrive/IBM/dataset/test'`) and preprocessed by resizing to `(180, 180)` and scaling pixel values between 0 and 1.

4. Model Setup:

- The VGG19 model is loaded with pre-trained weights from the ImageNet dataset using `VGG19` from `tensorflow.keras.applications.vgg19`.

- The VGG19 layers are set to be non-trainable, and a custom classification head is added.

- The model is compiled with the categorical cross-entropy loss function, the Adam optimizer, and accuracy as the evaluation metric.

5. Model Training:

- The `fit` function is called to train the model on the training set (`training\_set`) with validation on the test set (`test\_set`).

- The training is performed for 50 epochs with the specified steps per epoch and validation steps.

- The training progress and evaluation metrics are printed during training.

6. Model Evaluation:

- The `evaluate` function is used to calculate the loss and accuracy of the model on the test set (`test\_set`).

- The results are printed, showing the model's performance on test images.

7. Saving and Loading the Model:

- The trained model is saved to a file named `'wcv.h5'` using `model.save`.

- The model is then loaded from the saved file using `load\_model`.

8. Image Prediction:

- An example image (`sunrise1.jpg`) is loaded using `image.load\_img` from the specified path.

- The image is preprocessed by converting it to a numpy array, expanding its dimensions, and then passed through the loaded model to get the predicted probabilities.

- The index list of weather conditions is used to map the predicted class index to its corresponding label, which is stored in the `result` variable.

Overall, this code performs weather classification using a VGG19 model trained on the provided dataset. It includes data preparation, model setup, training, evaluation, saving/loading the model, and making predictions on new images.

5.2 Flask Web Application Code Explanation

Web Application Code :

import numpy as np

import os

from flask import Flask, request, render\_template

from tensorflow.keras.models import load\_model

from PIL import Image

model=load\_model("wcv.h5")

app=Flask (\_\_name\_\_, template\_folder='templates/')

app.config['UPLOAD\_FOLDER'] = 'uploads/'

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/about')

def about():

return render\_template("about.html")

@app.route('/images')

def image():

return render\_template("images.html")

@app.route('/input')

def input():

return render\_template("predict.html")

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

def res():

if request.method=="POST":

f=request.files['image']

filepath = os.path.join(app.config['UPLOAD\_FOLDER'], f.filename)

f.save(filepath)

img = Image.open(filepath)

img = img.resize((180, 180))

x = np.array(img)

x = np.expand\_dims(x, axis=0)

preds=model.predict(x)

pred=np.argmax(preds,axis=1)

index=['cloudy','foggy','rainy','shine','sunrise']

result=str(index[pred[0]])

return render\_template('result.html', prediction=result, path = f.filename)

if \_\_name\_\_== "\_\_main\_\_":

app.run(debug=True)

Explanation :

The provided code is a Flask web application that uses a trained model (`wcv.h5`) to classify weather conditions based on user-uploaded images. Here are the main features of the code:

1. Flask Setup: The code starts by importing the necessary dependencies, including Flask, numpy, os, and PIL (Python Imaging Library). It also loads the trained model using `load\_model` from `tensorflow.keras.models`.

2. App Routes: The code defines various routes that handle different functionalities of the web application:

- `'/'`: Renders the `index.html` template, which likely serves as the homepage.

- `'/about'`: Renders the `about.html` template, which provides information about the application.

- `'/images'`: Renders the `images.html` template, which might display some sample images or a gallery.

- `'/input'`: Renders the `predict.html` template, where users can upload an image to classify.

- `'/predict'`: Handles the prediction request when a user uploads an image. It saves the image, preprocesses it, and passes it through the loaded model to get the prediction. The predicted label is then rendered using the `result.html` template.

3. Model Prediction:

- After the user uploads an image, it is saved in the designated `UPLOAD\_FOLDER` using the `os.path.join` and `f.save` methods.

- The image is then opened using PIL's `Image.open` and resized to the desired input size of the model.

- The preprocessed image is converted to a numpy array and expanded to include a batch dimension using `np.expand\_dims`.

- The model's `predict` method is called on the preprocessed image to obtain the predicted probabilities for each weather class.

- `np.argmax` is used to find the index of the class with the highest probability, and the corresponding label is retrieved from the `index` list.

- The prediction result and the path to the uploaded image file are passed to the `result.html` template for display.

4. Flask App Execution: The `if \_\_name\_\_ == "\_\_main\_\_"` block ensures that the Flask app runs only when the script is executed directly (not imported as a module). It starts the Flask development server with the `app.run` method, enabling the debug mode for easier troubleshooting.

Overall, this code sets up a Flask web application that allows users to upload images for weather classification using a pre-trained model. The predicted weather condition is displayed to the user on a result page.

6. RESULTS

6.1 Performance Metrics

Based on the provided performance metrics:

For the training dataset:

- Validation loss (val\_loss): 0.0228

- Validation accuracy (val\_accuracy): 0.9967

For the test dataset:

- Accuracy: 0.9969879388809204

- Loss: 0.021998772397637367

These metrics indicate that the model achieves high accuracy and low loss on both the training and test datasets. The high accuracy values suggest that the model is able to accurately classify the weather conditions in the given images. The low loss values indicate that the model's predictions are close to the actual labels, further confirming its effectiveness in classifying the images.

Overall, the performance metrics demonstrate that the model trained using this project performs exceptionally well in weather classification tasks.

7. ADVANTAGES & DISADVANTAGES :

Advantages of automated weather classification using transfer learning:

1. Improved Accuracy: By leveraging pre-trained models and their learned features, automated weather classification can achieve higher accuracy compared to traditional rule-based or manually designed classification systems.

2. Efficiency: Transfer learning reduces the need for training models from scratch, saving time and computational resources. It allows for faster development and deployment of weather classification systems.

3. Adaptability: Transfer learning enables the model to adapt to different weather conditions and environments, making it more robust and capable of handling diverse datasets.

4. Scalability: The project can handle large volumes of weather data efficiently, enabling real-time or near-real-time processing and analysis. This scalability is important in applications that require continuous monitoring or forecasting.

5. Broad Applicability: The automated weather classification system can be applied in various fields such as weather forecasting, agriculture, renewable energy, transportation, and disaster management, providing valuable insights for decision-making and planning.

Disadvantages of automated weather classification using transfer learning:

1. Dataset Limitations: The performance of transfer learning heavily relies on the availability and quality of the training dataset. If the dataset used for pre-training the model does not adequately represent the target weather conditions or lacks diversity, it may limit the system's accuracy and generalization.

2. Overfitting: If the transfer learning process is not carefully managed, there is a risk of overfitting, where the model becomes too specialized in the pre-training domain and fails to generalize well to the target weather classification task.

3. Domain Adaptation Challenges: Transferring knowledge from one domain to another may encounter challenges due to differences in data distributions. Weather conditions can vary across regions or time, and the model may struggle to adapt to new or unseen patterns if the training data is not representative of these variations.

4. Interpretability: Deep learning models, including those used in transfer learning, can be complex and difficult to interpret. It may be challenging to understand the specific features or characteristics the model uses for weather classification, limiting transparency and interpretability.

5. Resource Requirements: Although transfer learning reduces training time compared to training models from scratch, it still requires substantial computational resources, especially for fine-tuning and optimization. High-performance hardware or cloud infrastructure may be necessary to handle the computational demands.

6. Dependence on Pre-trained Models: The success of the project relies on the availability of suitable pre-trained models. If there is a lack of appropriate pre-trained models for weather classification or if existing models become outdated, it may hinder the project's effectiveness.

It's important to consider these advantages and disadvantages when implementing automated weather classification using transfer learning and address any potential limitations to ensure the system's reliability and accuracy.

8. CONCLUSION

In conclusion, automated weather classification using transfer learning offers several advantages in accurately and efficiently categorizing weather conditions. By leveraging pre-trained models and their learned features, the project enhances accuracy, efficiency, adaptability, and scalability in weather classification systems.

The project's primary advantages include improved accuracy due to the utilization of pre-trained models, increased efficiency through faster development and deployment, adaptability to different weather conditions and environments, scalability to handle large volumes of data, and broad applicability across various fields.

However, the project also has its limitations, such as the reliance on representative and diverse datasets, potential risks of overfitting, challenges in domain adaptation, limited interpretability of complex models, resource requirements, and dependence on the availability and relevance of pre-trained models.

Despite these limitations, automated weather classification using transfer learning holds significant potential for weather forecasting, climate research, agriculture, renewable energy optimization, transportation, urban planning, and disaster management.

By addressing the project's challenges and leveraging its advantages, stakeholders can make informed decisions, improve predictions, optimize resource utilization, enhance safety, and support planning and decision-making processes in various industries and sectors reliant on accurate weather information.

Overall, automated weather classification using transfer learning contributes to advancing the field of weather analysis and prediction, ultimately benefiting society through improved weather forecasting, disaster preparedness, and sustainable decision-making.

9. FUTURE SCOPE

The future scope of the project on automated weather classification using transfer learning is promising and offers several potential avenues for further development and enhancement. Here are some areas of future exploration:

1. Advanced Model Architectures: Researchers can explore more advanced deep learning architectures specifically tailored for weather classification. Novel architectures, such as attention mechanisms, graph neural networks, or recurrent neural networks, can be investigated to capture temporal dependencies, spatial relationships, and complex weather patterns more effectively.

2. Fine-grained Weather Classification: While the project focuses on general weather classification (e.g., sunny, cloudy, rainy), future work can aim to achieve fine-grained classification of specific weather phenomena, such as different types of clouds, precipitation intensities, or wind patterns. This level of granularity can provide more detailed information for specialized applications and research.

3. Ensemble Methods: Combining multiple weather classification models or utilizing ensemble methods can potentially improve the overall accuracy and robustness of the system. Ensemble techniques, such as model averaging, stacking, or boosting, can be explored to leverage the strengths of different models and mitigate individual model biases.

4. Real-time Weather Monitoring: Expanding the project to support real-time or near-real-time weather monitoring is a valuable direction. By incorporating streaming data sources, such as weather sensors, satellite imagery, or IoT devices, the system can provide up-to-date and dynamic weather classification information for timely decision-making.

5. Transfer Learning for Extreme Weather Events: Transfer learning can be specifically applied to address the challenges of classifying extreme weather events, such as hurricanes, tornadoes, or severe storms. Developing specialized models that are trained on diverse and representative datasets of extreme weather patterns can enhance the system's ability to accurately identify and predict such events.

6. Integration with Decision Support Systems: The project can be integrated with decision support systems used in various industries. For example, integrating weather classification with agricultural decision support systems, renewable energy management platforms, or urban planning tools can provide valuable insights for optimized decision-making, resource allocation, and risk mitigation.

7. Explainable AI in Weather Classification: Enhancing the interpretability and explainability of the weather classification system is an important area for future research. Developing methods to understand and visualize the features or patterns that contribute to the classification decisions can increase the trustworthiness and adoption of the system.

8. Cross-Domain Transfer Learning: Exploring the potential for transferring knowledge and models from related domains, such as climate modeling, satellite imagery analysis, or remote sensing, can further improve the weather classification system's performance and generalization capabilities.

9. Integration with Forecasting Models: Integrating the automated weather classification system with weather forecasting models can create a more comprehensive and accurate forecasting system. The classified weather data can be used as input to improve the accuracy and precision of weather prediction models, leading to better forecasts and early warnings.

These future directions have the potential to advance automated weather classification using transfer learning, pushing the boundaries of accuracy, applicability, and usefulness in various fields that rely on weather information. Continuous research and development in these areas can contribute to more precise weather analysis, forecasting, and decision-making processes in the future.

10. APPENDIX :

Source Code :

Dataset Link :

https://www.kaggle.com/datasets/vijaygiitk/multiclass-weather-dataset?resource=download

Training Source Code :

from google.colab import drive

drive.mount("/content/drive")

!pip install tensorflow

from tensorflow.keras.applications.vgg19 import VGG19, preprocess\_input

from tensorflow.keras.layers import Flatten, Dense

from tensorflow.keras.models import Model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from PIL import ImageFile

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

train\_datagen = ImageDataGenerator( rescale = 1./255,

shear\_range = 0.2,

zoom\_range = [.99,1.01],

brightness\_range = [0.8,1.2],

data\_format = "channels\_last",

fill\_mode ="constant",

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale=1./255)

training\_set = train\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM/dataset/train',

target\_size = (180, 180), batch\_size = 64,

class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM/dataset/test',

target\_size = (180, 180), batch\_size = 64,

class\_mode = 'categorical')

IMAGE\_SIZE =[180,180]

VGG19 = VGG19(input\_shape=IMAGE\_SIZE + [3], weights='imagenet',include\_top=False)

for layer in VGG19.layers:

layer.trainable = False

x = Flatten() (VGG19.output)

prediction = Dense(5, activation='softmax')(x)

model = Model(inputs=VGG19.input, outputs=prediction)

model.summary()

model.compile(

loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'] )

r = model.fit(

training\_set,

validation\_data = test\_set,

epochs = 50,

steps\_per\_epoch=len(training\_set),

validation\_steps = len(test\_set)

)

loss, accuracy = model.evaluate(test\_set,

steps=11,

verbose=2,

use\_multiprocessing=True,

workers=2)

print(f'Model performance on test images:\nAccuracy = {accuracy}\nLoss = {loss}')

model.save('wcv.h5')

model = load\_model("/content/wcv.h5")

img = image.load\_img(r"/content/drive/MyDrive/IBM/dataset/train/sunrise/sunrise1.jpg",target\_size= (180,180))

x = image.img\_to\_array(img)

x = np.expand\_dims(x,axis = 0)

preds=model.predict(x)

pred=np.argmax(preds,axis=1)

index=['cloudy','foggy','rainy','shine','sunrise']

result=str(index[pred[0]])

result

Flask Source Code :

import numpy as np

import os

from flask import Flask, request, render\_template

from tensorflow.keras.models import load\_model

from PIL import Image

model=load\_model("wcv.h5")

app=Flask (\_\_name\_\_, template\_folder='templates/')

app.config['UPLOAD\_FOLDER'] = 'uploads/'

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/about')

def about():

return render\_template("about.html")

@app.route('/images')

def image():

return render\_template("images.html")

@app.route('/input')

def input():

return render\_template("predict.html")

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

def res():

if request.method=="POST":

f=request.files['image']

filepath = os.path.join(app.config['UPLOAD\_FOLDER'], f.filename)

f.save(filepath)

img = Image.open(filepath)

img = img.resize((180, 180))

x = np.array(img)

x = np.expand\_dims(x, axis=0)

preds=model.predict(x)

pred=np.argmax(preds,axis=1)

index=['cloudy','foggy','rainy','shine','sunrise']

result=str(index[pred[0]])

return render\_template('result.html', prediction=result, path = f.filename)

if \_\_name\_\_== "\_\_main\_\_":

app.run(debug=True)

Demo Video Link :

<https://drive.google.com/drive/folders/1g41s_Mx0mAzJTLRPh2lpnqCaFpA7qg5T?usp=sharing>

Git-Hub Link : <https://github.com/naanmudhalvan-SI/PBL-NT-GP--5908-1680839598>