Sri Lanka Institute of Information Technology

Status
Document 02
2024

GROUP ID: R24-059

Bandara A M S S

IT21103322

Information Technology

DECLARATION

We declare that this is our own work and this project report does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Fernando W.P.A.M	IT21078378	

Signature of the Supervisor
(Mr. Nelum Chathuranga)

Date

……11/09/2024……

Table of Contents

System diagram

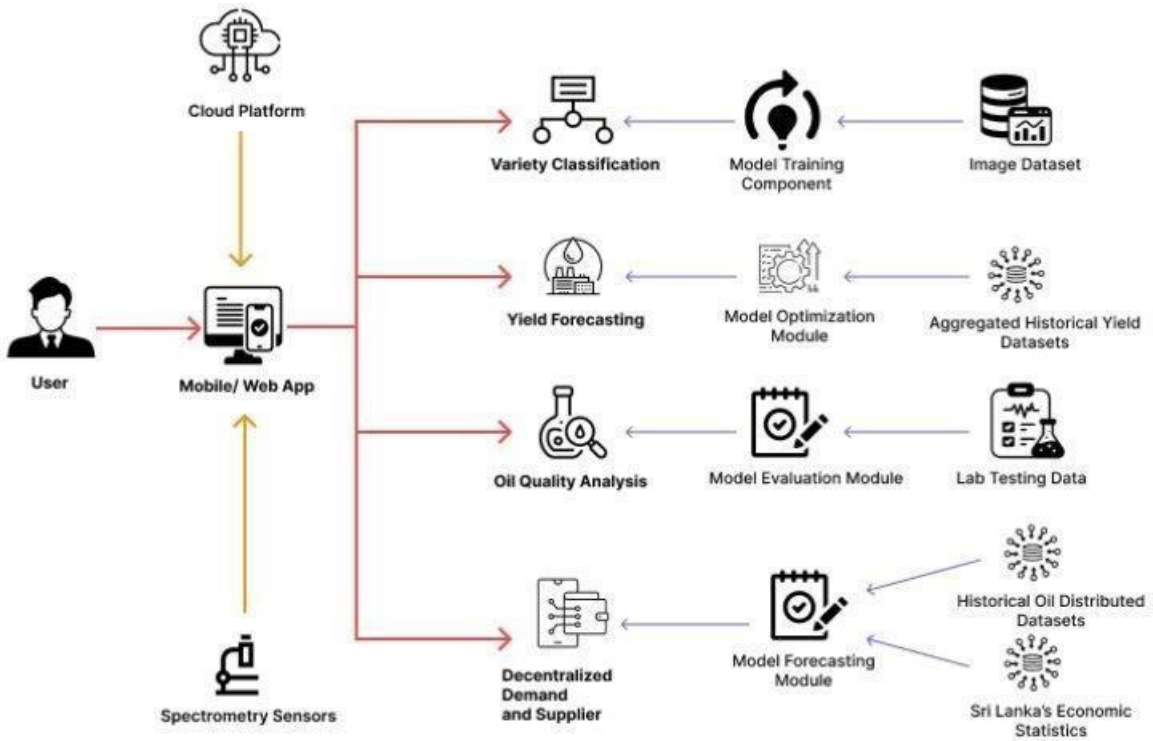


Figure 1 -System diagram

Figure 2 -System diagram

Component: live coconut oil quality measuring feature

Progress

Implementation of frontend

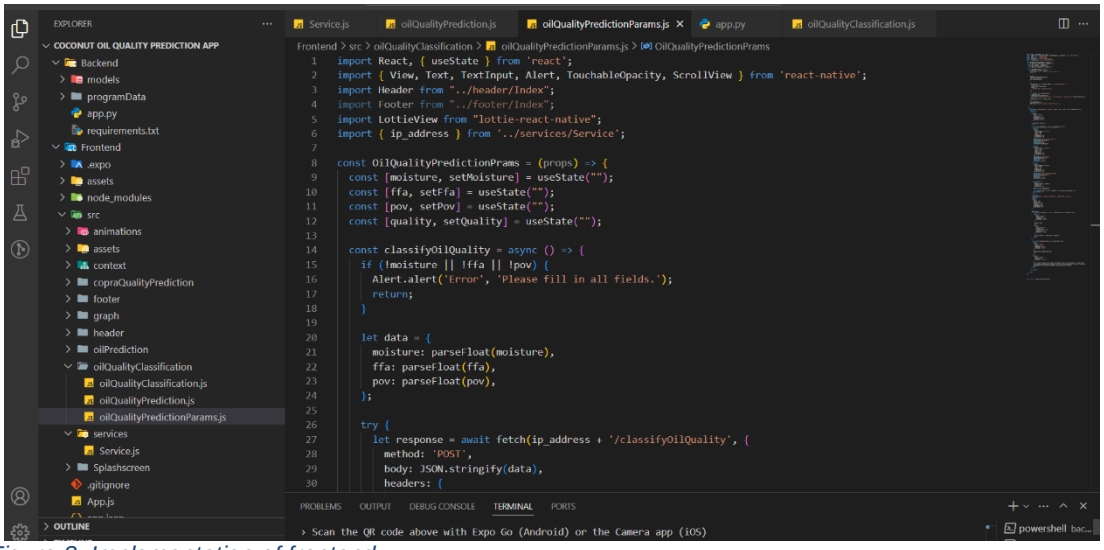


Figure 3 Implementation of frontend

Implementation of backend

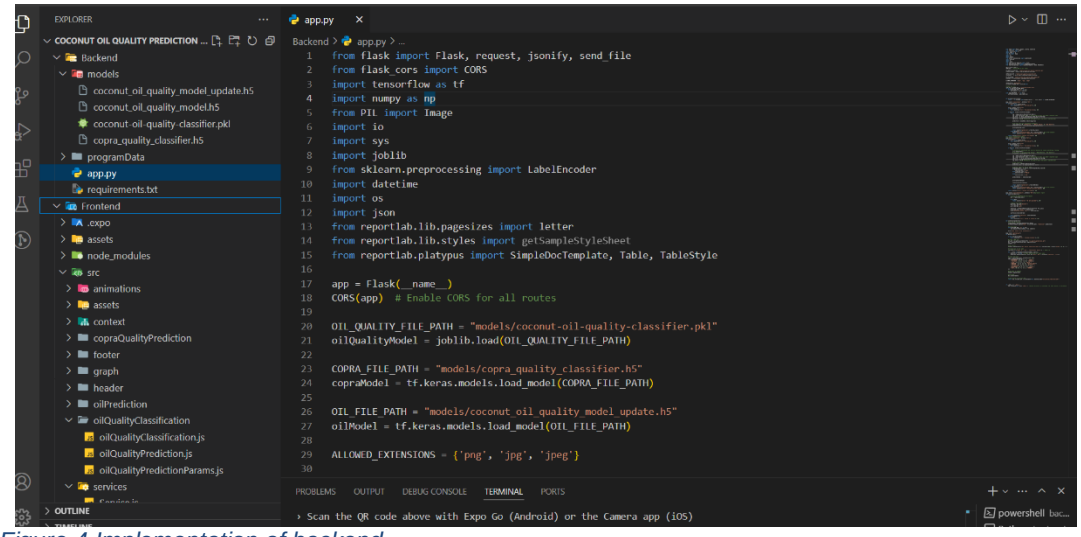


Figure 4 Implementation of backend

Cnn model

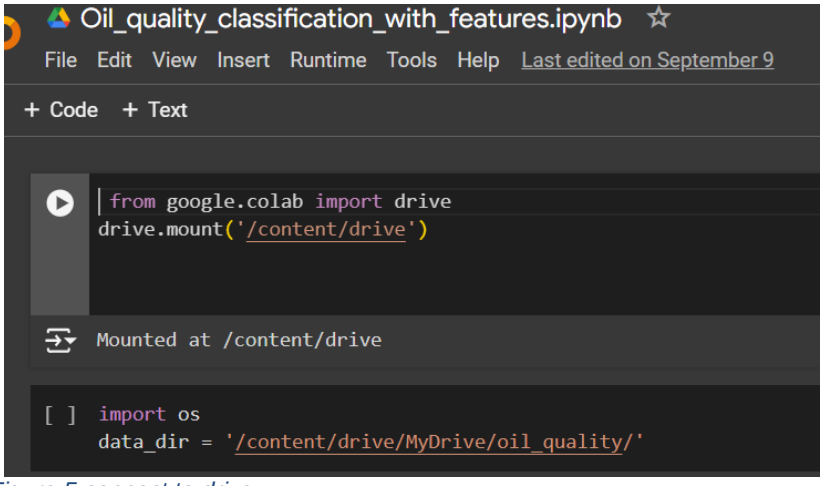


Figure 5 connect to drive

```
[ ] import cv2
    import numpy as np
    import tensorflow as tf
    from tensorflow.keras import layers, models
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    import matplotlib.pyplot as plt
```

Figure 6 import libraries

```
[ ]  def load_images_and_labels(data_dir):
         images = []
         labels = []
         for label in ['A', 'B', 'C']:
             label_dir = os.path.join(data_dir, label)
             for image_name in os.listdir(label_dir):
                 img_path = os.path.join(label_dir, image_name)
                 img = cv2.imread(img_path)
                 img = cv2.resize(img, (224, 224))  # Resize for uniformity
                 images.append(img)
                 labels.append(label)
         return np.array(images), np.array(labels)

     images, labels = load_images_and_labels(data_dir)


[ ]  le = LabelEncoder()
     labels = le.fit_transform(labels)
```

Figure 7 Data Preprocessing image data

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)
```

Figure 8 train and test data

Color Analysis (Lab* Color Space Conversion)

```
[ ]  def lab_color_analysis(img):
         lab = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)
         l, a, b = cv2.split(lab)
         return np.mean(l), np.mean(a), np.mean(b)

     color_features = np.array([lab_color_analysis(img) for img in X_train])
```

Clarity Analysis (Edge Detection & Histogram)

```
[ ]  def clarity_analysis(img):
         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         edges = cv2.Canny(gray, 100, 200)
         histogram = cv2.calcHist([edges], [0], None, [256], [0, 256])
         clarity_score = np.sum(histogram[128:])  # Focus on the high-end of the histogram
         return clarity_score

     clarity_scores = np.array([clarity_analysis(img) for img in X_train])
```

Impurity Detection (Image Segmentation)

```
[ ]  def impurity_detection(img):
         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         _, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY_INV)
         contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
         return len(contours)  # Number of detected impurities

     impurity_counts = np.array([impurity_detection(img) for img in X_train])
```

Figure 9 image analysis

Combine Features and Build the CNN Model

```
[ ]  X_train_features = np.hstack((color_features, clarity_scores.reshape(-1, 1), impurity_counts.reshape(-1, 1)))
```

```
[ ]  model = models.Sequential([
         layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(64, (3, 3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(128, (3, 3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Flatten(),
         layers.Dense(128, activation='relu'),
         layers.Dense(len(le.classes_), activation='softmax')
     ])

     model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

     model.fit(X_train, y_train, epochs=10, validation_split=0.2)
```

Figure 10 Combine Features and Build the CNN Model

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 76s 5s/step - accuracy: 0.5296 - loss: 348.9953 - val_accuracy: 1.0000 - val_loss: 3.0129e-06
Epoch 2/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 82s 5s/step - accuracy: 0.8803 - loss: 4.9466 - val_accuracy: 0.9612 - val_loss: 1.9083
Epoch 3/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 66s 4s/step - accuracy: 0.9487 - loss: 4.4450 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 69s 4s/step - accuracy: 0.9515 - loss: 5.8025 - val_accuracy: 0.9922 - val_loss: 0.6583
Epoch 5/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 66s 4s/step - accuracy: 0.9730 - loss: 5.3953 - val_accuracy: 0.9922 - val_loss: 2.4671
Epoch 6/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 86s 4s/step - accuracy: 0.9252 - loss: 13.2204 - val_accuracy: 0.6047 - val_loss: 18.7784
Epoch 7/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 83s 4s/step - accuracy: 0.8671 - loss: 5.9960 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 8/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 76s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 9/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 88s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/10
16/16 ━━━━━━━━━━━━━━━━━━━━ 76s 4s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
<keras.src.callbacks.history.History at 0x7c67e771e6e0>
```

Figure 11 prediction accuracy

```
[ ] test_loss, test_acc = model.evaluate(X_test, y_test)
    print(f"Test Accuracy: {test_acc}")

6/6 ━━━━━━━━━━━━━━━━━━━━ 5s 783ms/step - accuracy: 1.0000 - loss: 0.0000e+00
Test Accuracy: 1.0

[ ] model.save('coconut_oil_quality_model.h5')

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recomm

[ ] from google.colab import files
    files.download('coconut_oil_quality_model.h5')

[ ] Start coding or generate with AI.
```

Figure 12 download the trained model

Import Libraries

[10]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import joblib
```

Data Import

[2]:
```
# Load the dataset
data = pd.read_csv('coconut_quality_dataset.csv')
```

Data Cleaning

[4]:
```
# Data cleaning
# Check for any missing values
print("Missing values before cleaning:")
print(data.isnull().sum())
```

```
Missing values before cleaning:
MOISTURE    0
FFA         0
POV         0
QUALITY     0
dtype: int64
```

[5]:
```
data.head()
```

[5]:

	MOISTURE	FFA	POV	QUALITY
0	0.190	0.190	0.190	A
1	0.199	0.197	0.196	A
2	0.198	0.196	0.195	A
3	0.197	0.195	0.194	A
4	0.196	0.194	0.193	A

Figure 13

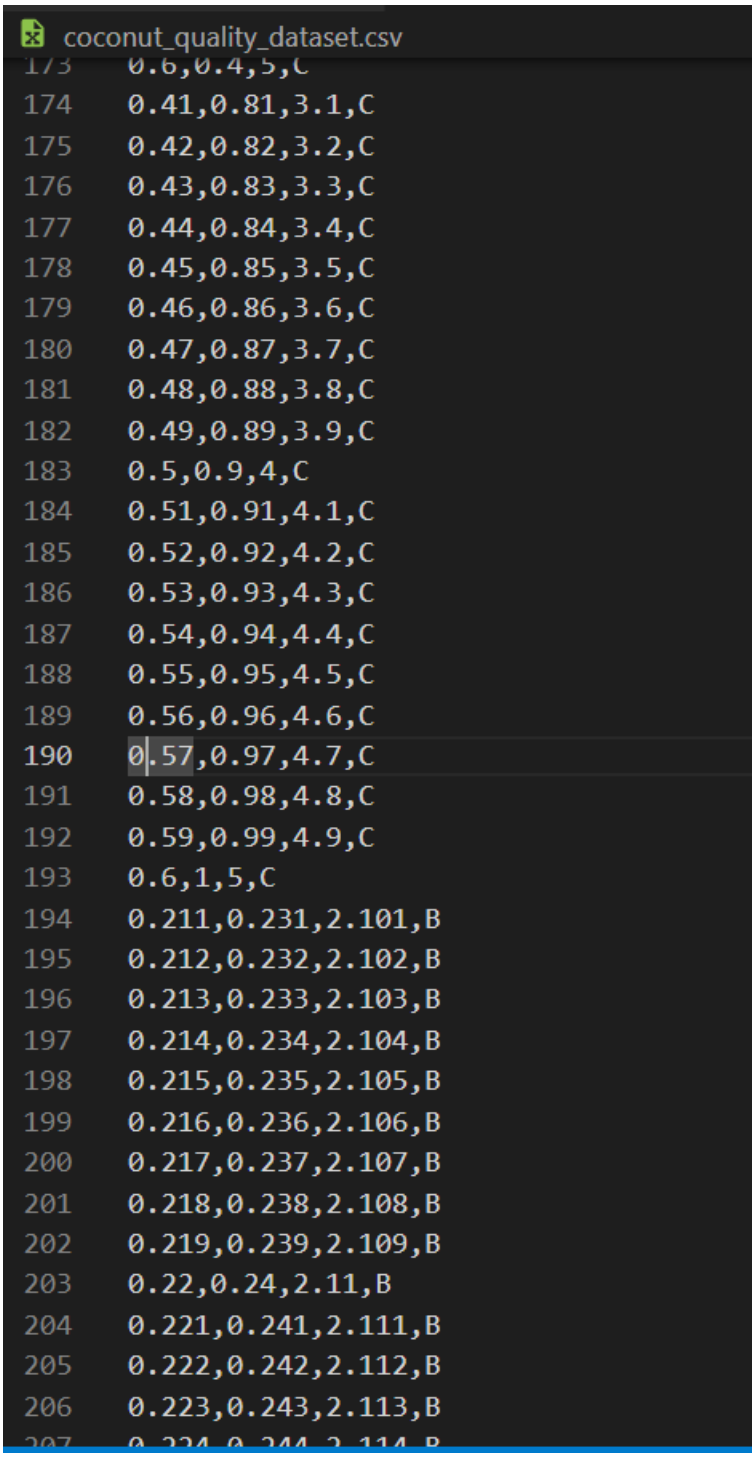

coconut_quality_dataset.csv
0.6,0.4,5,C
0.41,0.81,3.1,C
0.42,0.82,3.2,C
0.43,0.83,3.3,C
0.44,0.84,3.4,C
0.45,0.85,3.5,C
0.46,0.86,3.6,C
0.47,0.87,3.7,C
0.48,0.88,3.8,C
0.49,0.89,3.9,C
0.5,0.9,4,C
0.51,0.91,4.1,C
0.52,0.92,4.2,C
0.53,0.93,4.3,C
0.54,0.94,4.4,C
0.55,0.95,4.5,C
0.56,0.96,4.6,C
0.57,0.97,4.7,C
0.58,0.98,4.8,C
0.59,0.99,4.9,C
0.6,1,5,C
0.211,0.231,2.101,B
0.212,0.232,2.102,B
0.213,0.233,2.103,B
0.214,0.234,2.104,B
0.215,0.235,2.105,B
0.216,0.236,2.106,B
0.217,0.237,2.107,B
0.218,0.238,2.108,B
0.219,0.239,2.109,B
0.22,0.24,2.11,B
0.221,0.241,2.111,B
0.222,0.242,2.112,B
0.223,0.243,2.113,B

Figure 14

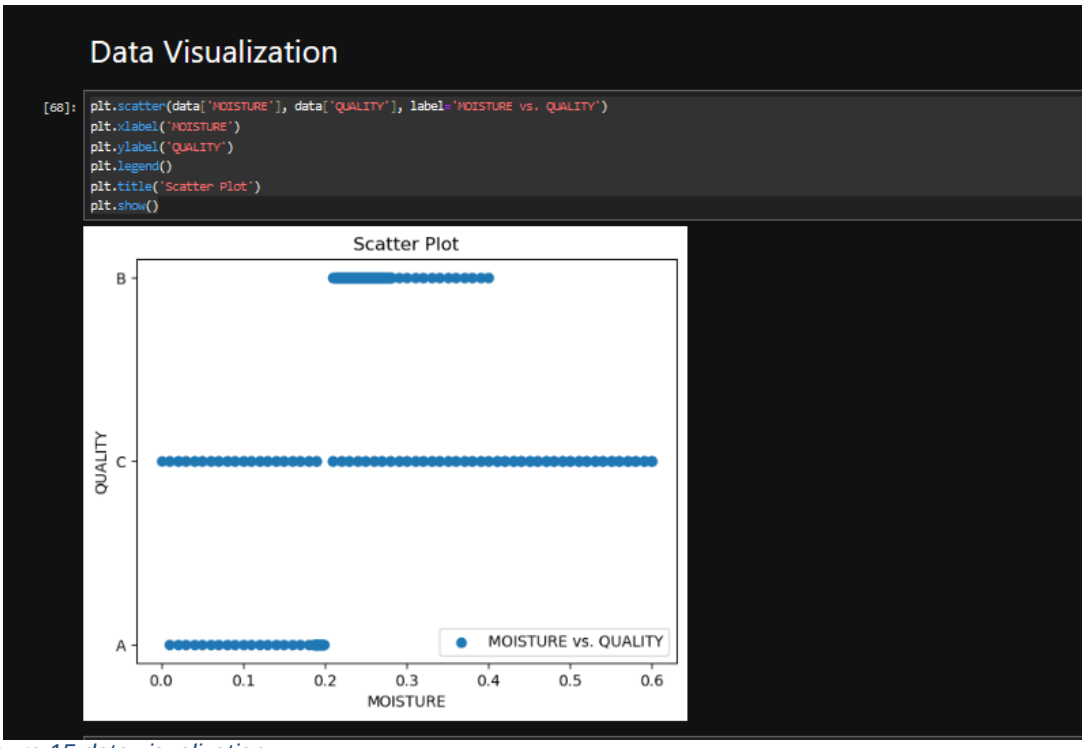
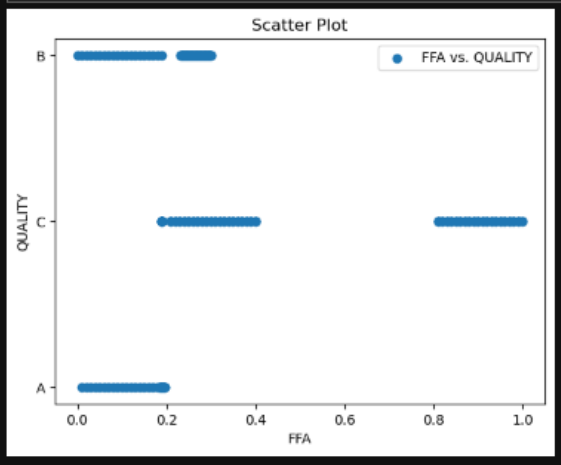


Figure 15 data visualization

```
[69]: plt.scatter(data['FFA'], data['QUALITY'], label='FFA vs. QUALITY')
      plt.xlabel('FFA')
      plt.ylabel('QUALITY')
      plt.legend()
      plt.title('Scatter Plot')
      plt.show()
```




```
[70]: plt.scatter(data['POV'], data['QUALITY'], label='POV vs. QUALITY')
      plt.xlabel('POV')
      plt.ylabel('QUALITY')
      plt.legend()
      plt.title('Scatter Plot')
      plt.show()
```

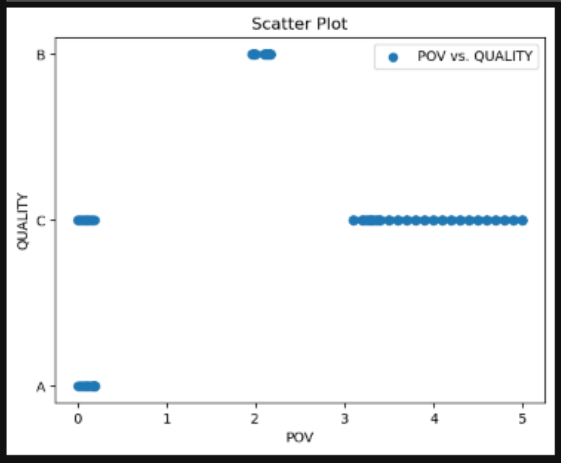
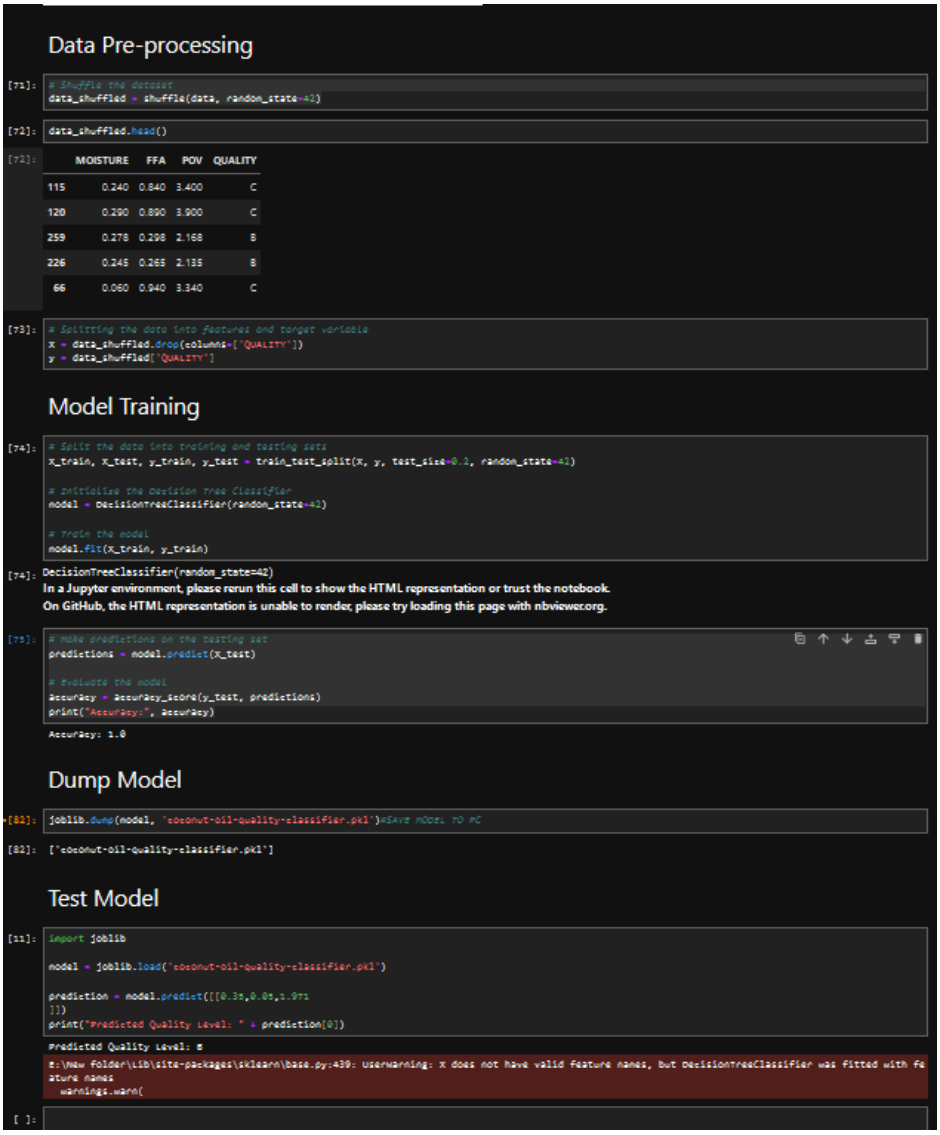


Figure 16 visulization.



Data Pre-processing

```
[71]: # Shuffle the dataset
data_shuffled = shuffle(data, random_state=42)
```

```
[72]: data_shuffled.head()
```

[72]:

	MOISTURE	FFA	POV	QUALITY
115	0.240	0.840	3.400	C
120	0.290	0.890	3.900	C
259	0.278	0.298	2.168	B
226	0.245	0.265	2.135	B
66	0.060	0.940	3.340	C

```
[73]: # Splitting the data into features and target variable
X = data_shuffled.drop(columns=['QUALITY'])
y = data_shuffled['QUALITY']
```

Model Training

```
[74]: # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Decision Tree Classifier
model = DecisionTreeClassifier(random_state=42)

# Train the model
model.fit(X_train, y_train)
```

[74]:
```
DecisionTreeClassifier(random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
[75]: # Make predictions on the testing set
predictions = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

Dump Model

```
[82]: joblib.dump(model, 'coconut-oil-quality-classifier.pkl')#Save model to PC
```

[82]:
```
['coconut-oil-quality-classifier.pkl']
```

Test Model

```
[11]: import joblib

model = joblib.load('coconut-oil-quality-classifier.pkl')

prediction = model.predict([[0.35,0.05,1.971
]])
print("Predicted Quality Level: " + prediction[0])
```

```
Predicted Quality Level: B
E:\New Folder\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with fe
ature names
  warnings.warn(
```

```
[ ]:
```

Figure 17

Test Cases

Test Case 1 to Verify Image Upload for Coconut Oil Quality Classification

Test Case Id	**1**
Test Case	**Verify Image Upload**
Test Scenario	**Verify whether the uploaded image is successfully processed for quality classification**
Input	**Coconut oil sample images**
Expected Output	**1. 200 status code should be displayed 2. The image should be processed and a prediction returned**
Actual Result	**1. 200 status code displayed 2. Image processed and quality prediction returned**
Status (Pass/Fail)	**Pass**

Test Case 2 to Classify Coconut Oil Quality Using Image Input (CNN Model)

Test Case Id	**2**
Test Case	**Classification of Coconut Oil Quality**
Test Scenario	**Testing images of coconut oil to classify and determine quality using the CNN model**
Precondition	**The coconut oil quality model must be trained and deployed**
Input	**Coconut oil sample images**
Expected Output	**High accuracy (prediction of 'Good', 'Medium', or 'Bad' quality)**
Actual Result	**Classification result with appropriate quality level**
Status (Pass/Fail)	**Pass**

Test Case 3 to Classify Coconut Oil Quality Using Parameter Input (Decision Tree)

Test Case Id	**3**
Test Case	**Classification of Coconut Oil Quality Using Parameters**
Test Scenario	**Testing coconut oil quality by providing**

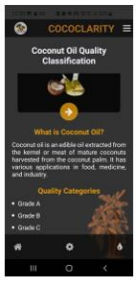
	Moisture, FFA, and POV parameters
Precondition	**Coconut oil quality prediction model trained and deployed**
Input	**Parameters: Moisture, Free Fatty Acid (FFA), Peroxide Value (POV)**
Expected Output	**Accurate classification of the coconut oil quality ('Grade A', 'Grade B', 'Grade C')**
Actual Result	**Quality prediction returned as 'Grade A', 'Grade B', or 'Grade C'**
Status (Pass/Fail)	**Pass**

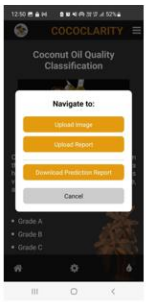
Test Case to 4 Generate a Report for Coconut Oil Quality Predictions

Test Case Id	**04**
Test Case	**Generate a Report for Oil Quality Predictions**
Test Scenario	**Testing the functionality to generate a PDF report based on past predictions**
Precondition	**Past oil quality prediction data stored**
Input	**Request for PDF generation**
Expected Output	**1. PDF report generated successfully 2. Report contains data about previous predictions**
Actual Result	**PDF generated with accurate prediction data**
Status (Pass/Fail)	**Pass**

UI


COCOCLARITY


COCOCLARITY
Coconut Oil Quality Classification
What is Coconut Oil?
Quality Categories
Grade A
Grade B
Grade C


COCOCLARITY
Coconut Oil Quality Classification
Navigate to:
Upload Image
Upload Report
Download Prediction Report
Cancel
Grade A
Grade B
Grade C


COCOCLARITY
Oil Quality Prediction


COCOCLARITY
Oil Quality Prediction
Enter Moisture Content (%)
Enter FFA (%)
Enter Peroxide Value (POV)
Machine Learning Model

Team Communication

The team chose Microsoft Teams as their primary communication channel, forming a dedicated Team with all four group members. We also used Zoom to communicate with supervisors, provide updates, and receive comments on the project's progress.

.

Teams Channel

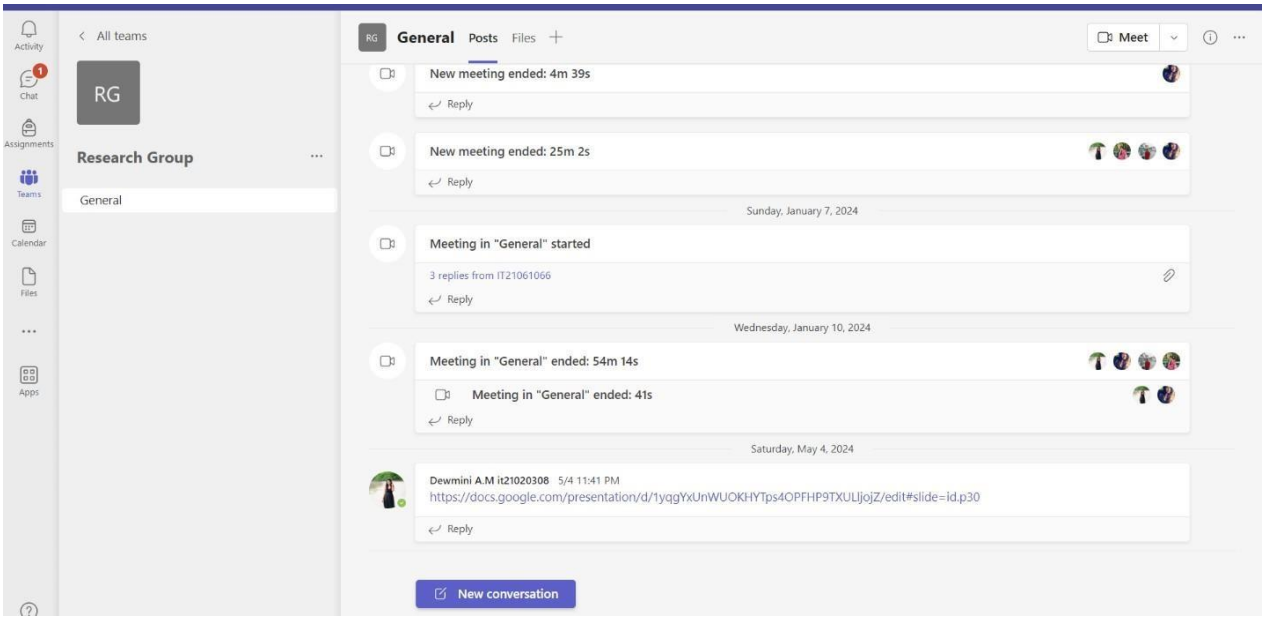


Figure 18 teams channel creation

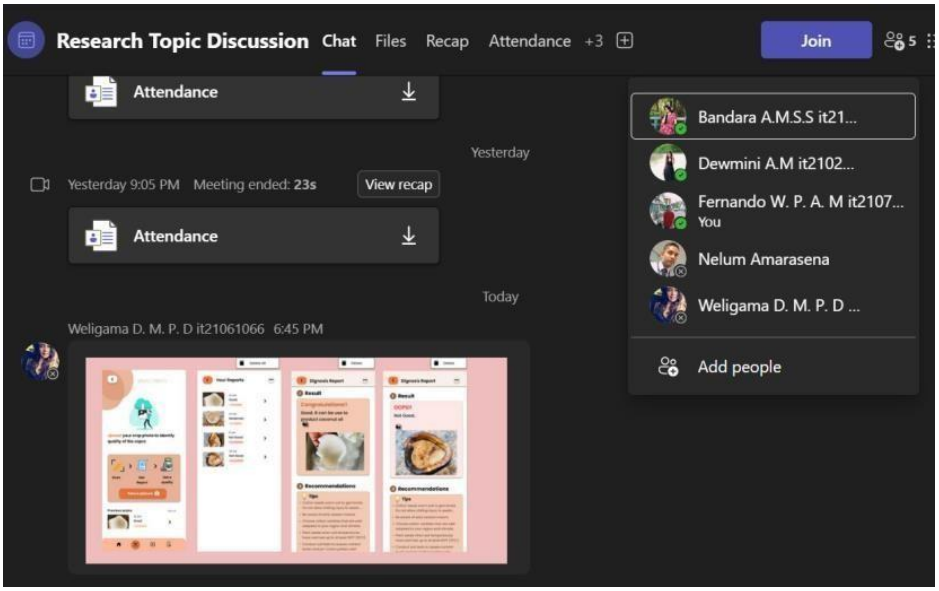


Figure 19 teams channel creation

Teams Calls with the Research Team

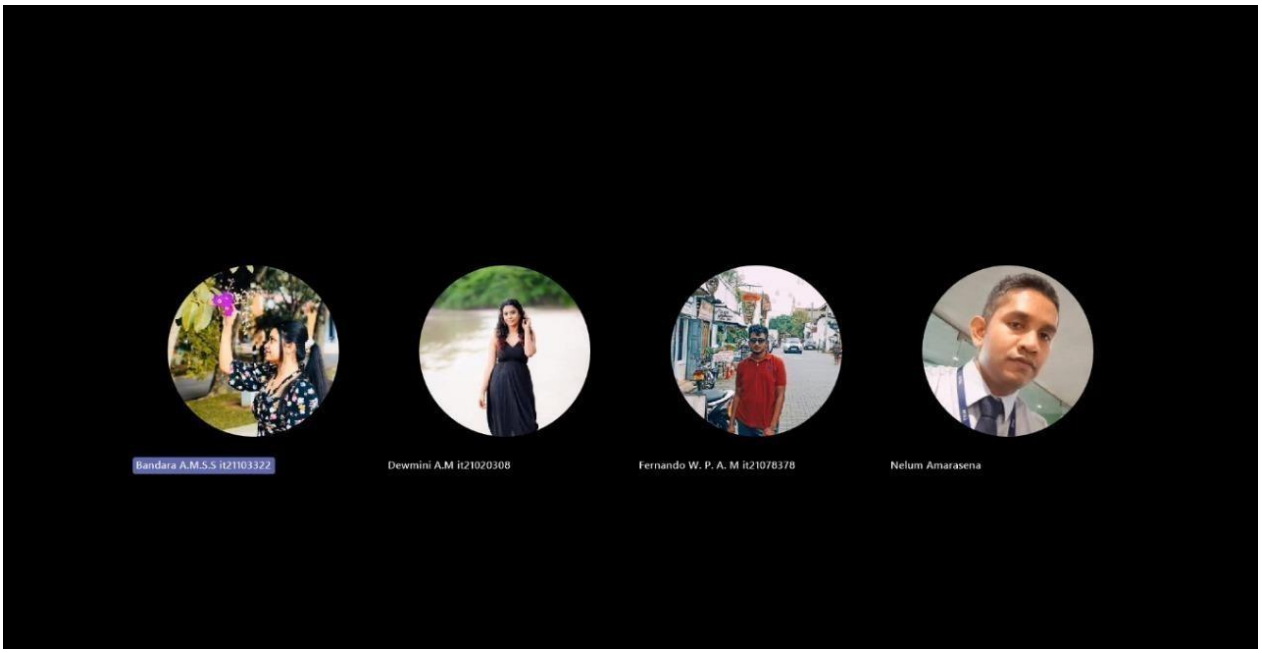


Figure 20 teams calls overview

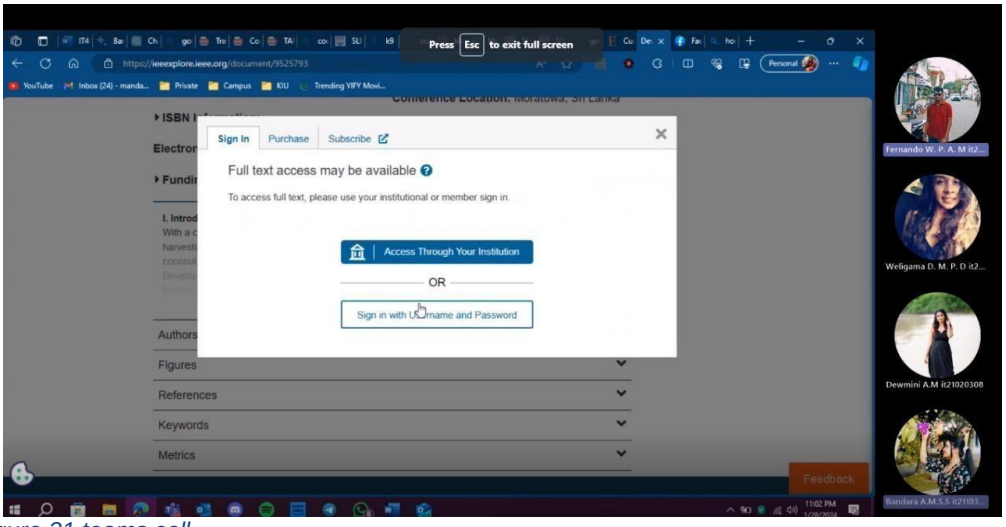


Figure 21 teams call

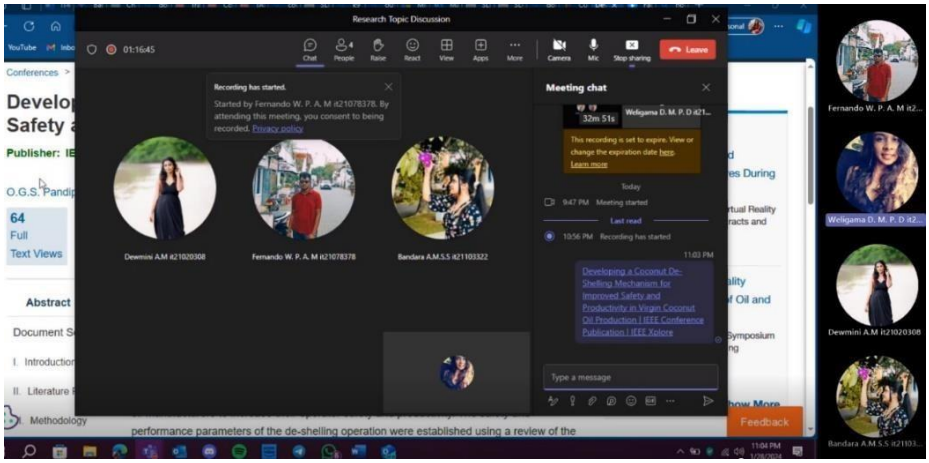


Figure 22 teams call with group members

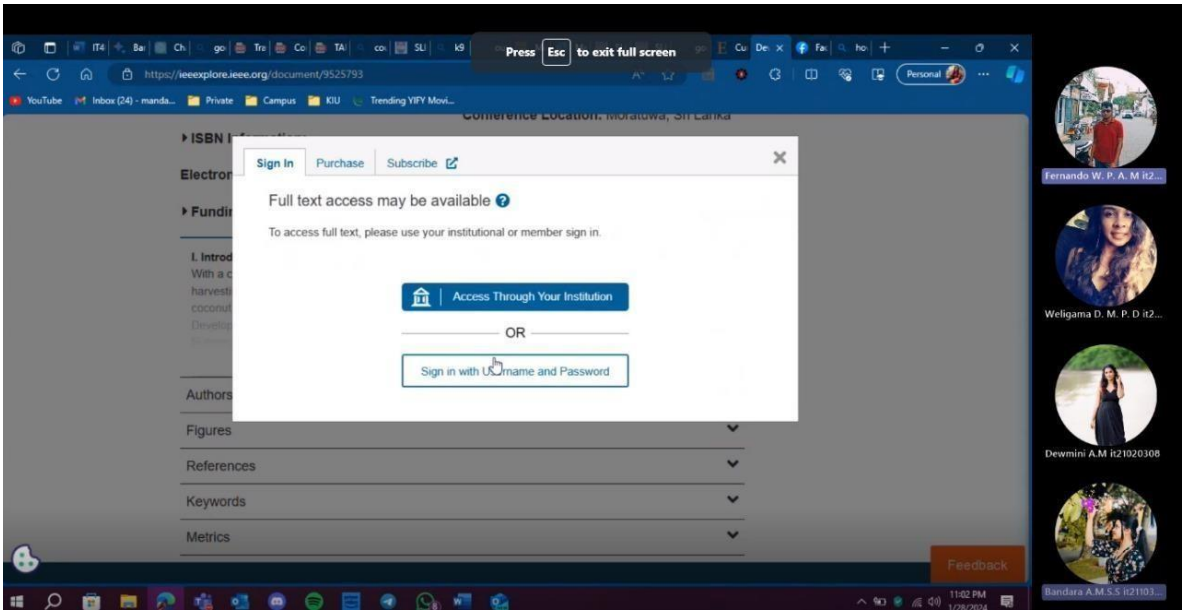


Figure 24 teams call with group members

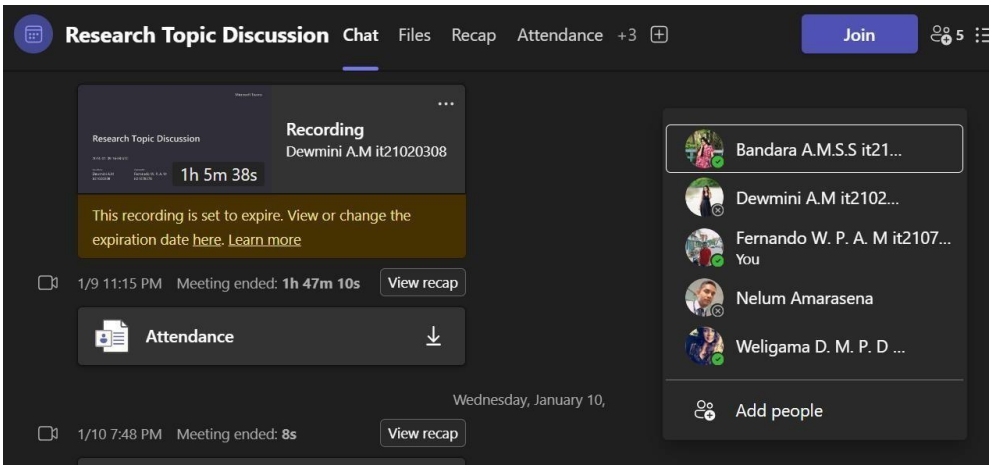


Figure 23 teams call with supervisor

WhatsApp Group Creation & meetings

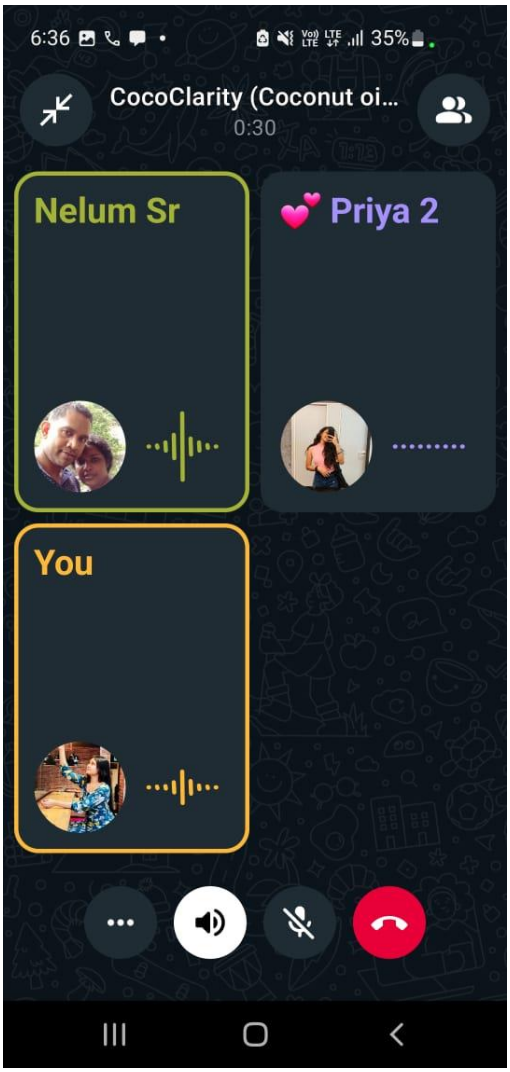


Figure 25 commiuncationwith supervisor

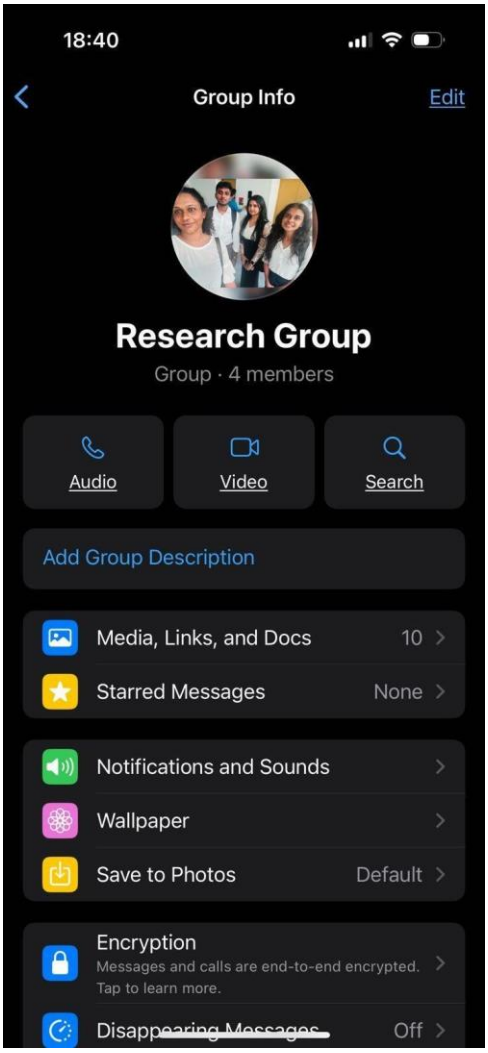


Figure 26 whatsapp group creation

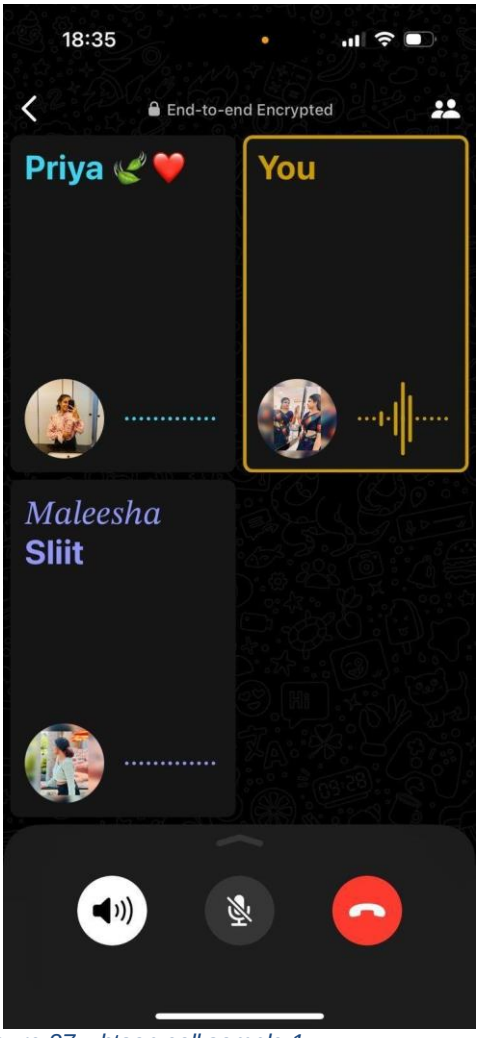


Figure 27 whtsap call sample 1

Figure 28 whtsapp call sample2

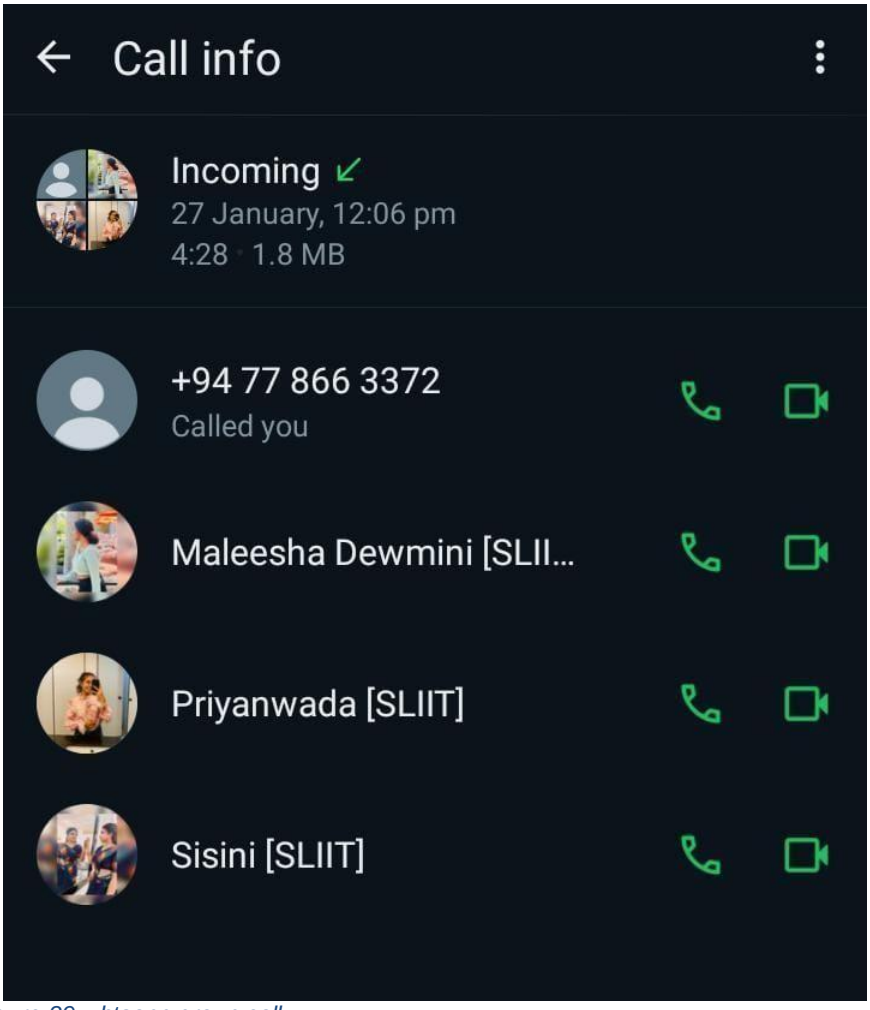


Figure 29 whtsapp group call

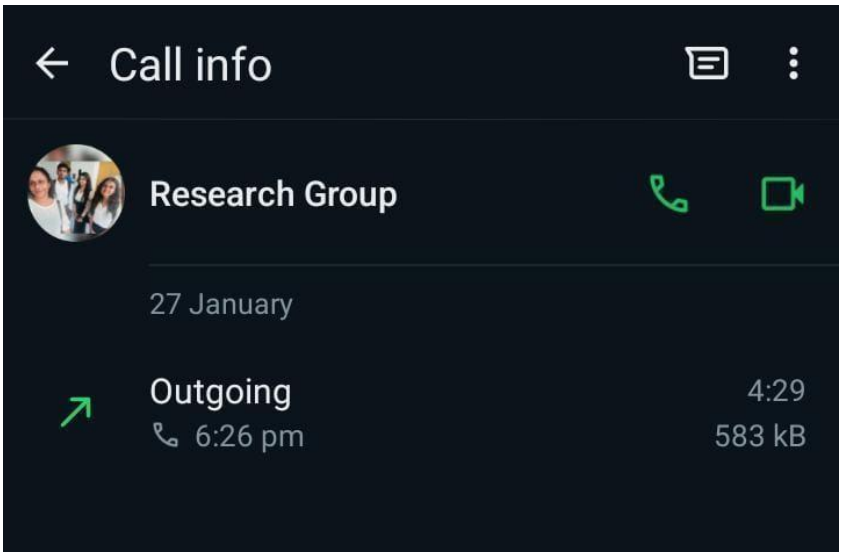


Figure 30 whtsapp call sample2

Online Calls with Supervisor

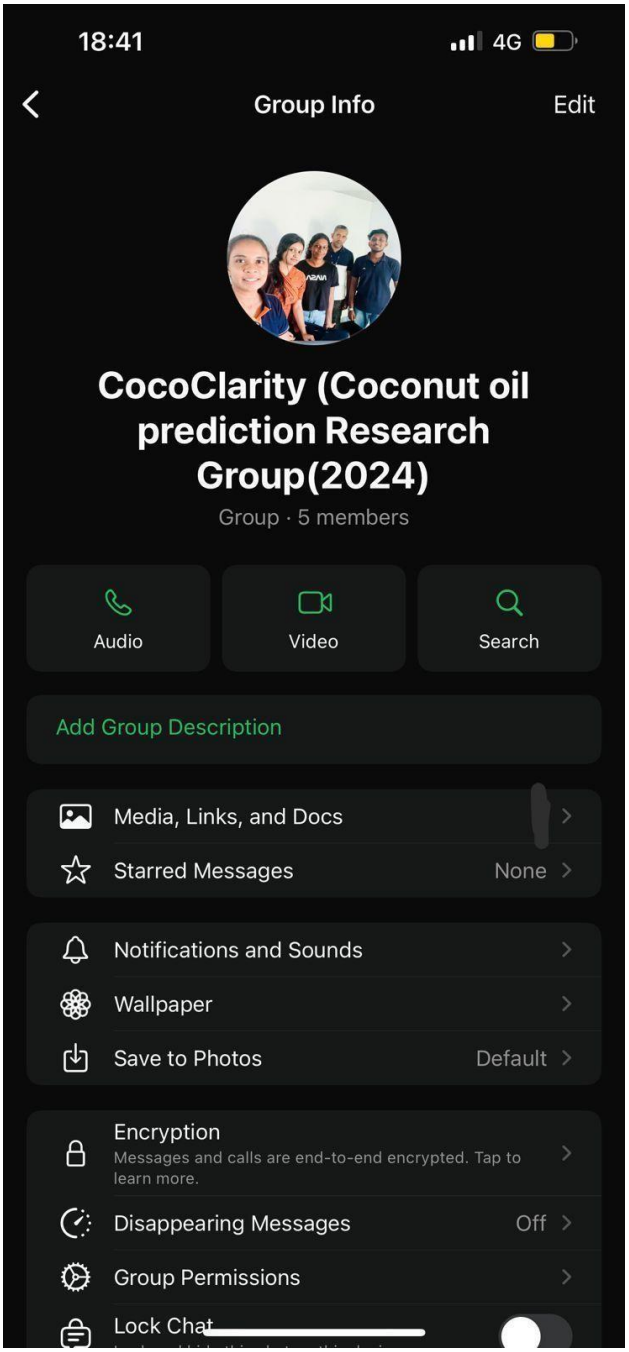


Figure 31 group creation with supervisor

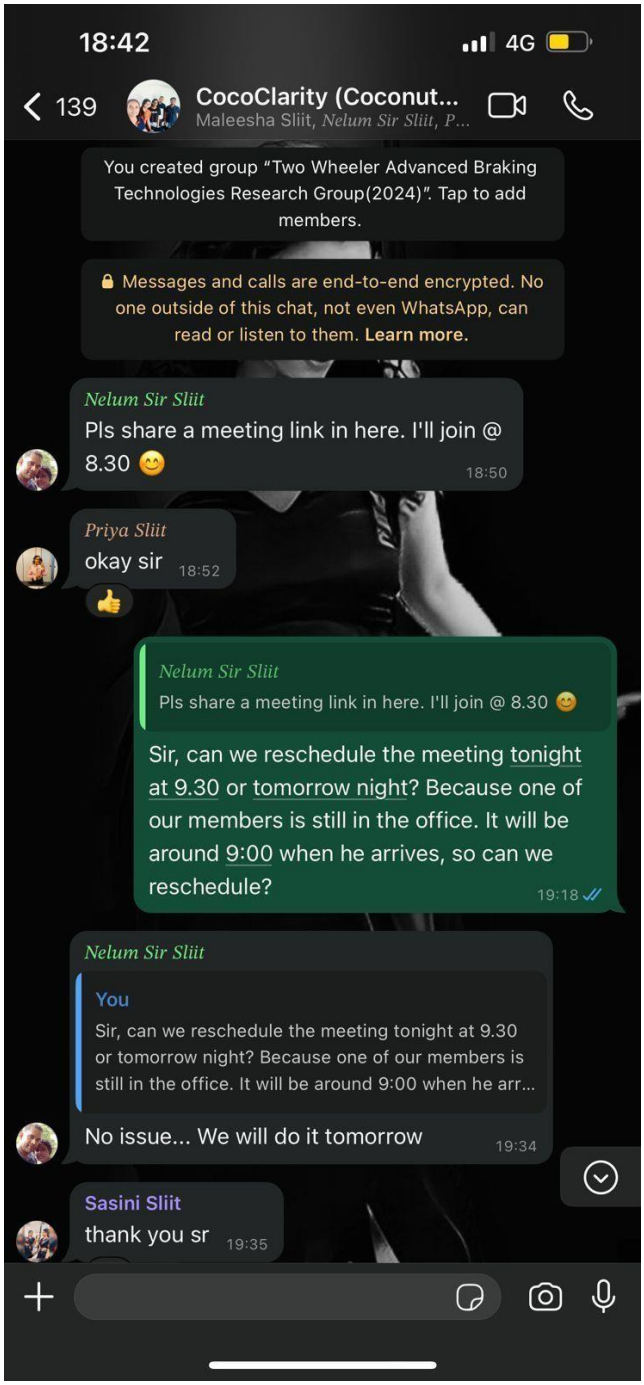


Figure 32 discusson with supervisor

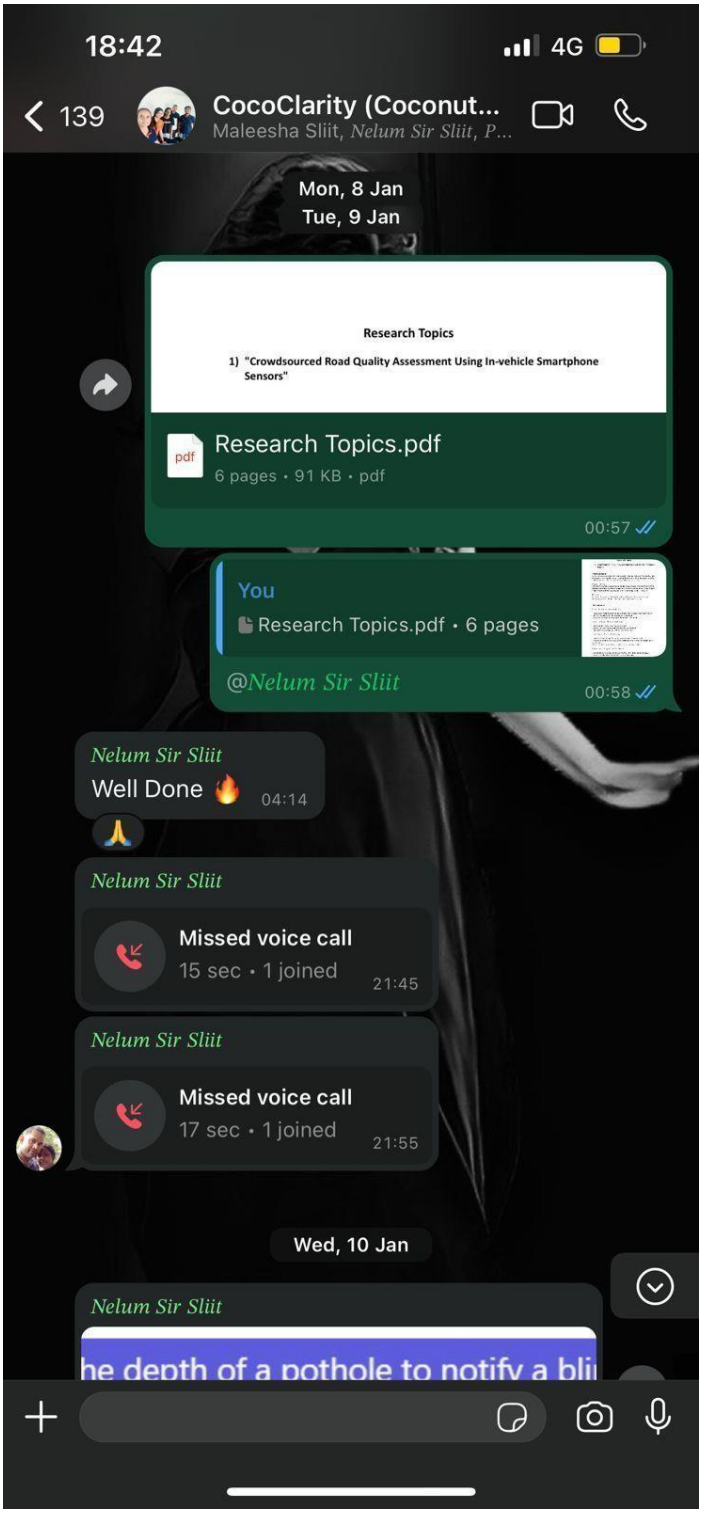


Figure 33 discusson with supervisor

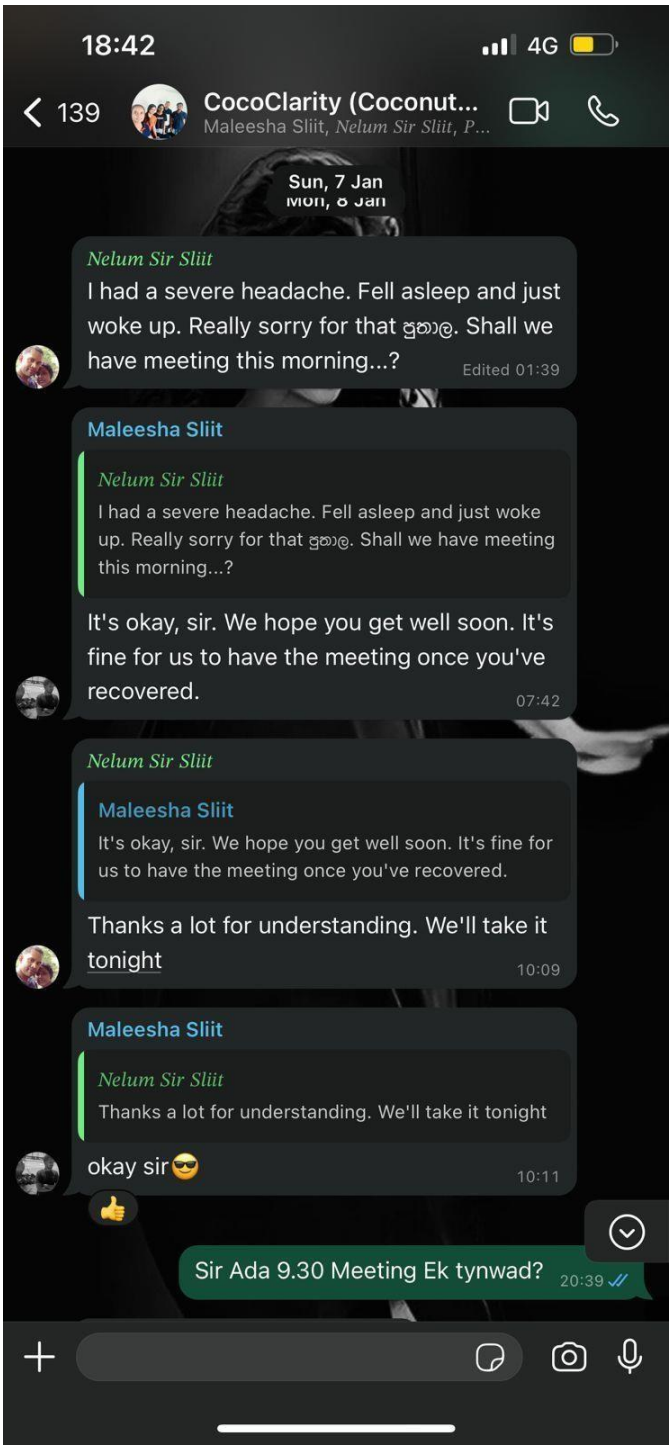


Figure 34 discusson with supervisor

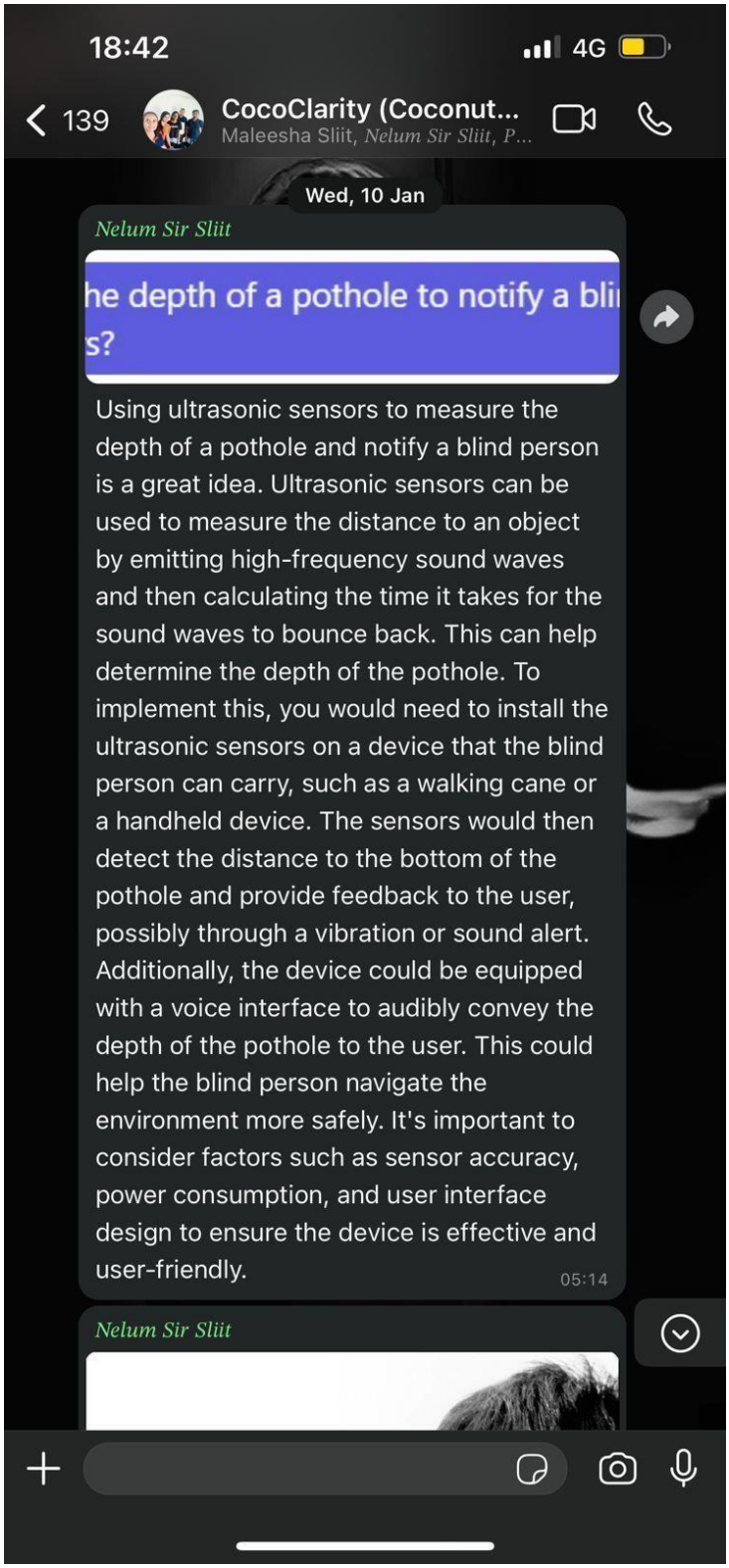


Figure 35 supervisor advising

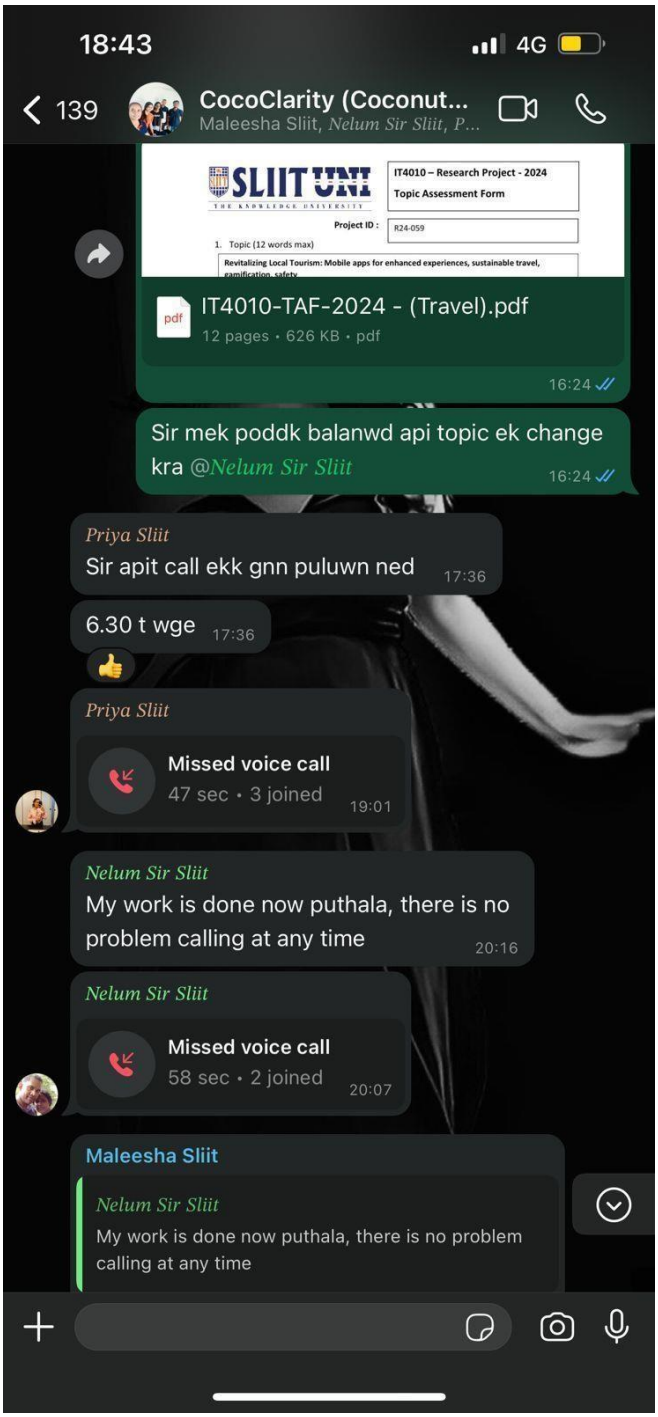


Figure 36 discusson

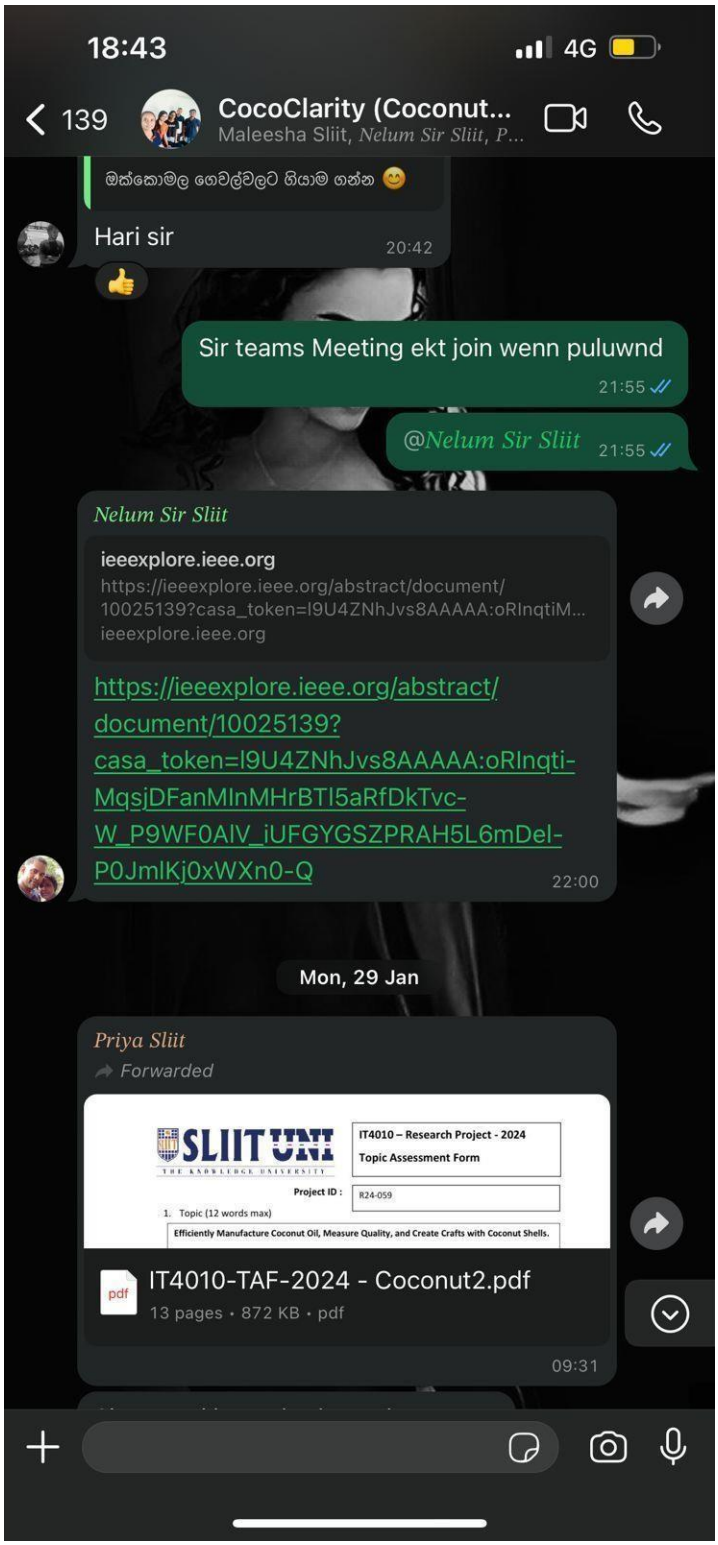


Figure 37 discussion with supervisor

Figure 38 call with supervisor

External supervisor meetings

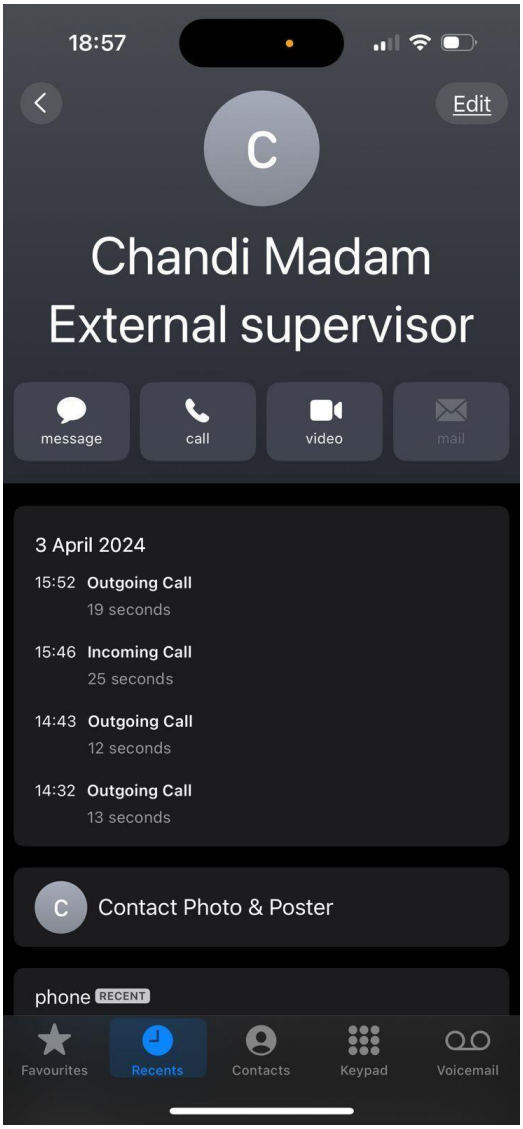


Figure 39 call with external supervisor

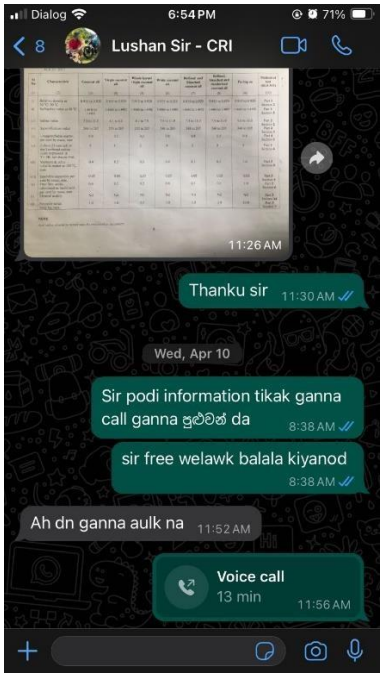


Figure 40 discuss information with CRI officer

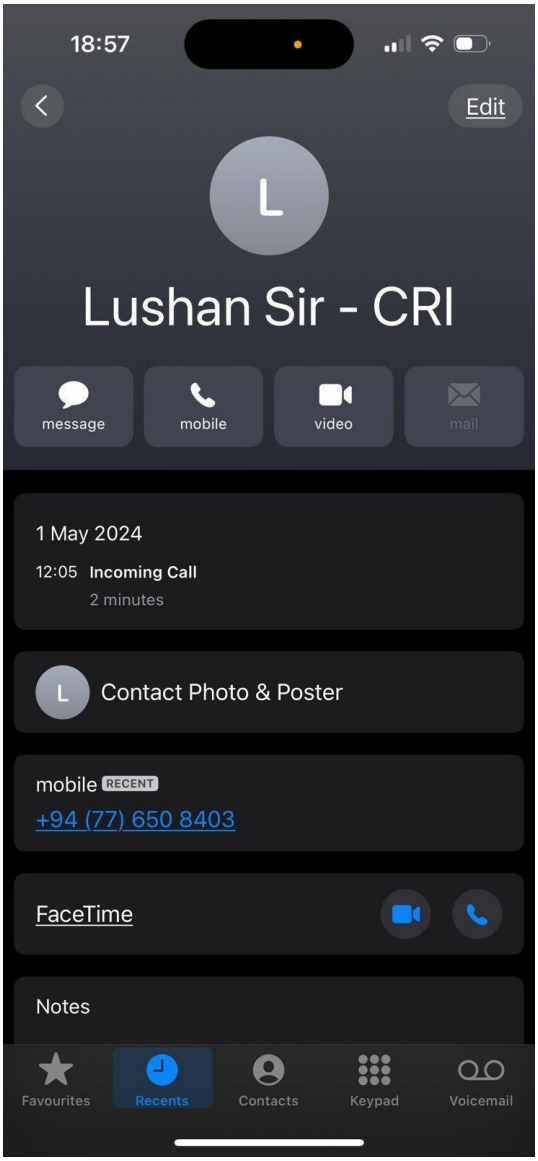


Figure 41 call with CRI officer

Figure 42 meetup with supervisor

Figure 43 meeting with group members

Figure 44 meet with CRI officer

Figure 45 CRI Lunuwila

Figure 46 CRI lab

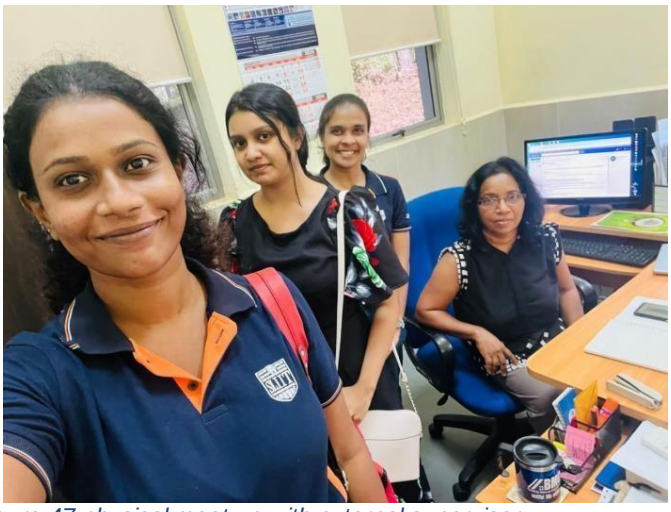


Figure 47 physical meet up with external supervisor

Teams' meetings with supervisor

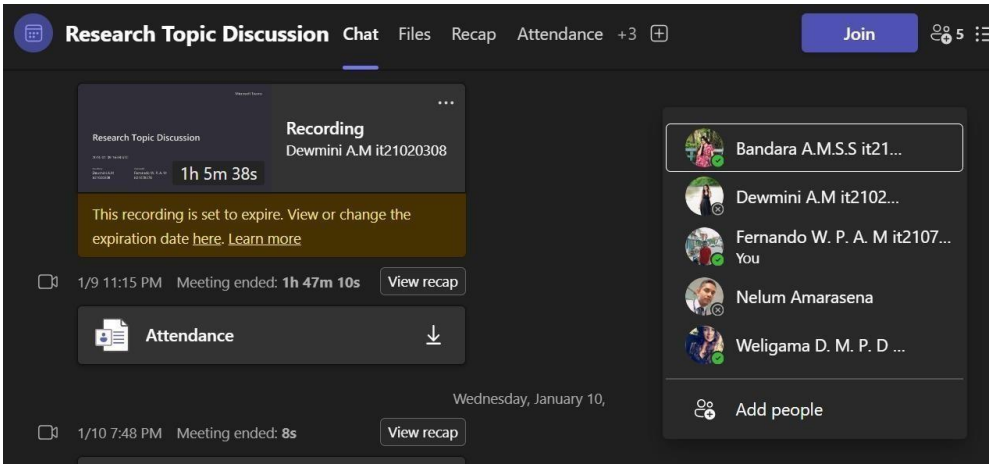


Figure 48 teams meeting with supervisor1

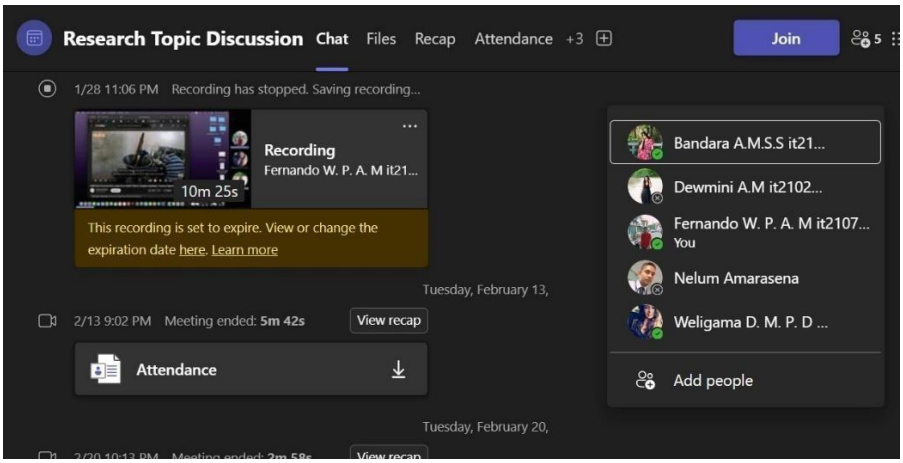


Figure 49 teams meeting with supervisor2

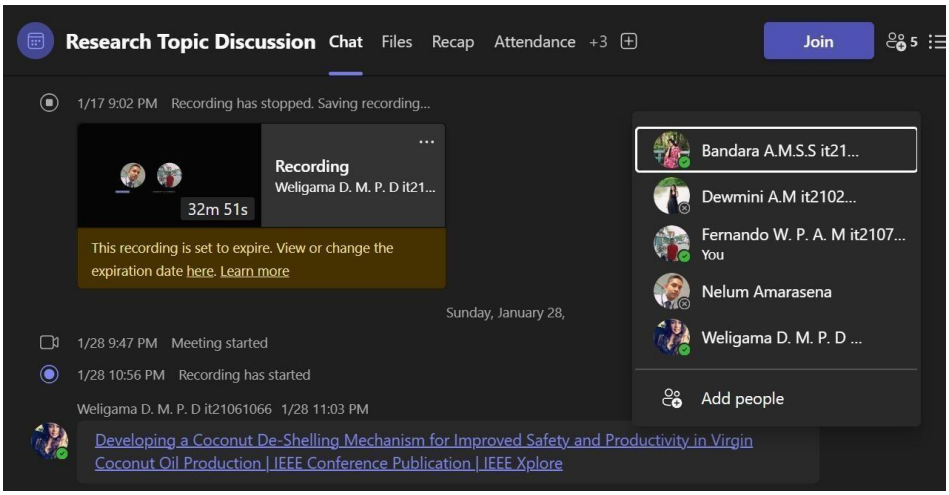


Figure 50 teams meeting with supervisor3

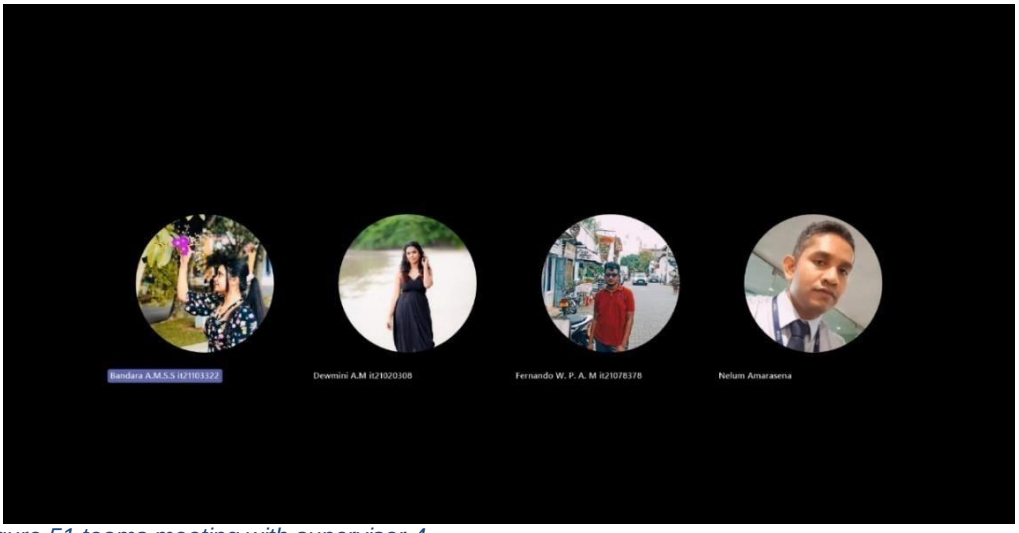


Figure 51 teams meeting with supervisor 4

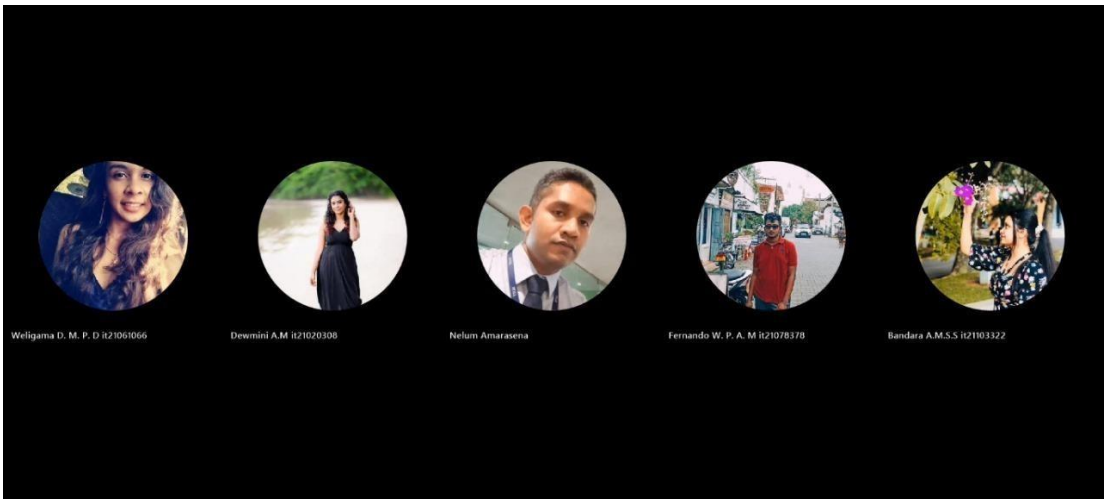


Figure 52 teams meeting with supervisor5

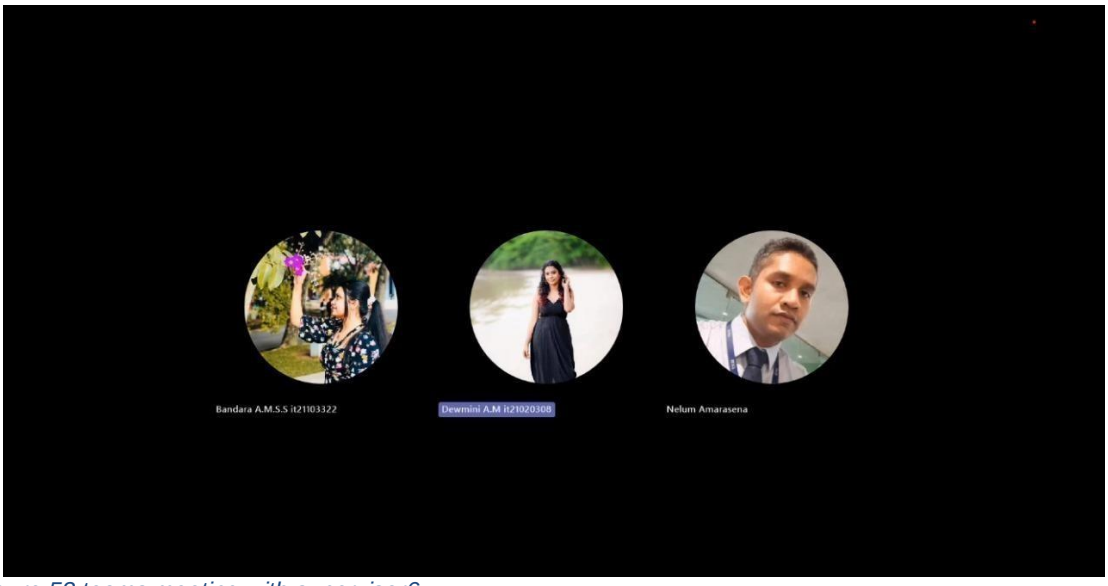


Figure 53 teams meeting with supervisor6

Project Timeline

A Gantt chart is a visual aid for project management that is used to display a project's chronology. It shows the beginning and ending dates of all project components, including tasks, milestones, and stages, along with the dependencies between them.

.

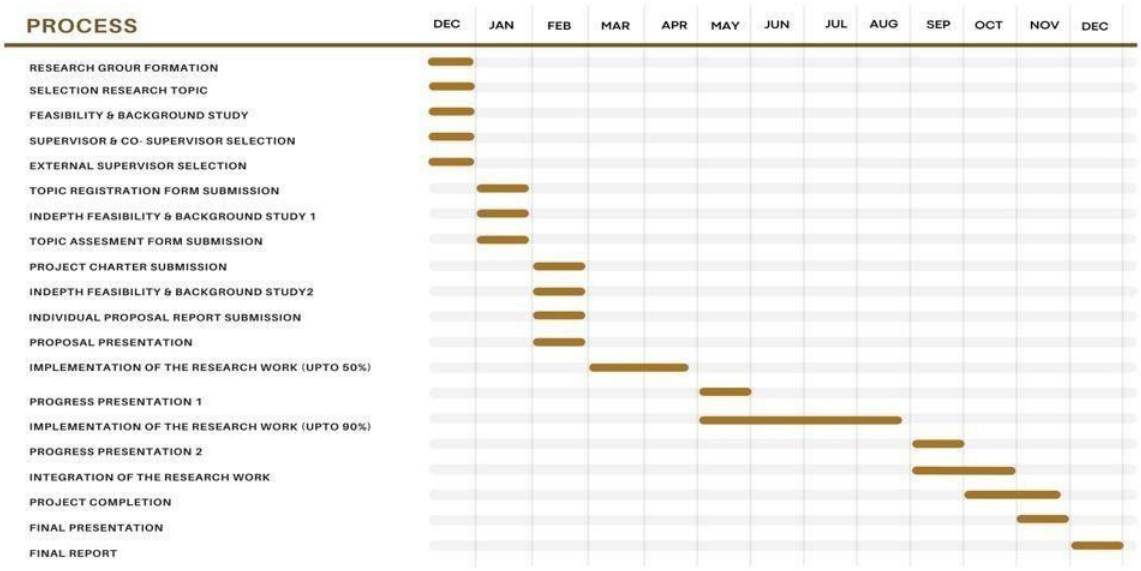


Figure 54 gantt chart

Figure 55 gantt chart

Figure 56 gantt chart

Figure 57 gantt chart

Work Break-Down

A project is broken down into smaller, easier-to-manage components using a hierarchical process called a work breakdown structure, or WBS. It facilitates project planning, execution, and control by breaking the project up into distinct deliverables and work packages.

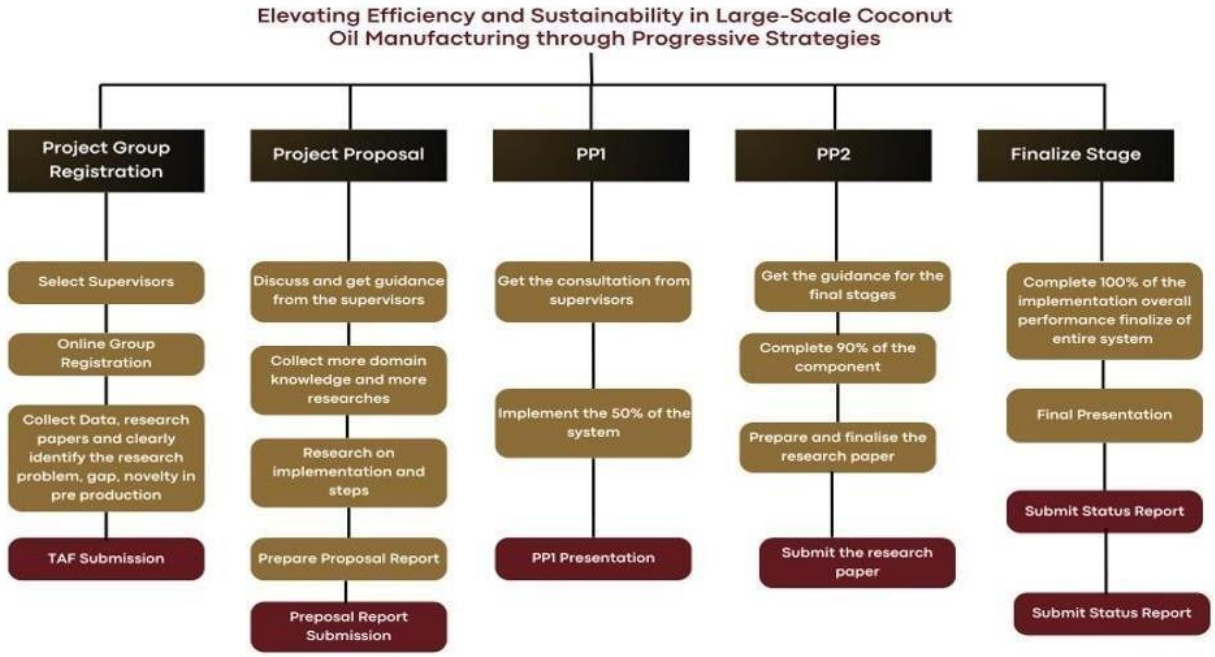


Figure 58 Work Break-Down Structure

Figure 59 Work Break-Down Structure

Figure 60 Work Break-Down Structure

Figure 61 Work Break-Down Structure