

Sri Lanka Institute of Information Technology



R24-002

FERNANDO R.D.S.A.

IT21011948

Information System Engineering

Table of Contents

Table of Contents	i
Table of Figures	ii
Project Component and Current Status	1
Progress	1
Team Communication	9
Teams Channel.....	9
Teams Calls with the Research Team	10
Online Calls with Supervisors	12
Physical Meetings with Group Members.....	14
WhatsApp Group Creation	15
Project Timeline.....	17
Work Break-Down	18
Version Controlling.....	19
GitHub Commits	20
GitHub Graph of Commits	22
GitHub Contributors	22
Contribution Charts.....	23

Table of Figures

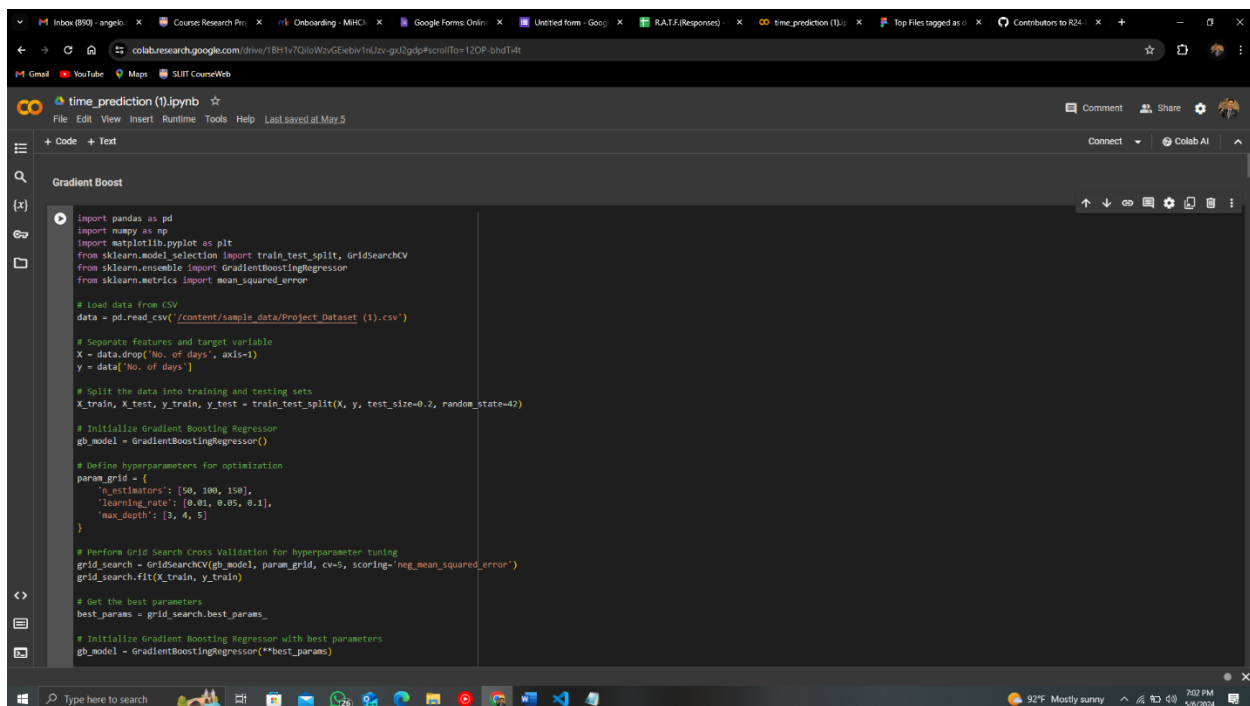
Figure 1 Machine Learning modeling.	1
Figure 2- Actual vs predicted graph	2
Figure 3-Confusion Matrix.....	3
Figure 4- ML Model.....	3
Figure 5-Random Forest model	4
Figure 6-SVG exact date model	4
Figure 7-Random Forest exact date	5
Figure 8- RF implementation	5
Figure 9-Random Forest changed.....	6
Figure 10-RF without scale	7
Figure 11 - Teams Channel Creation	9
Figure 12 - Overview of Team Calls and Communication.....	10
Figure 13 - Team Member Teams Calls (Example 01)	10
Figure 14 - Team Member WhatsApp Calls (Example 02)	11
Figure 15 - Teams Meeting with the Co-Supervisor (Example 01)	12
Figure 16 - Teams Meeting with the Co-supervisor (Example 02)	12
Figure 17 - WhatsApp Call with External Supervisor (LSEG)	13
Figure 18 - Physical Meetings with team members.....	14
Figure 19 - WhatsApp Group Creation with Co-Supervisor	15
Figure 20 - WhatsApp Group Creation with Supervisor	16
Figure 21 - Gantt Chart (Project Timeline)	17
Figure 22 - Work Break-Down Structure	18
Figure 23 - GitHub Repository	19
Figure 24 - GitHub Repository (Component 04 Branch)	20
Figure 25 - GitHub Commits.....	21
Figure 26 - Git Graph	22
Figure 27 - GitHub Contributors	23
Figure 28 - Contributors Charts	23

Project Component and Current Status

Component: Risk-Adjusted Time Forecast

Progress

- The system mainly focuses on giving time prediction for a project and measure the severity of risks identified in a certain project.
- A Time estimation model was trained after working on more than 6 Machine learning models to predict the time taken for a project which spent a very high effort to develop.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error

# Load data from CSV
data = pd.read_csv('/content/sample_data/Project_Dataset (1).csv')

# Separate features and target variable
X = data.drop("No. of days", axis=1)
y = data["No. of days"]

# 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 Gradient Boosting Regressor
gb_model = GradientBoostingRegressor()

# Define hyperparameters for optimization
param_grid = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5]
}

# Perform Grid Search Cross Validation for hyperparameter tuning
grid_search = GridSearchCV(gb_model, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_

# Initialize Gradient Boosting Regressor with best parameters
gb_model = GradientBoostingRegressor(**best_params)
```

Figure 1 Machine Learning modeling.

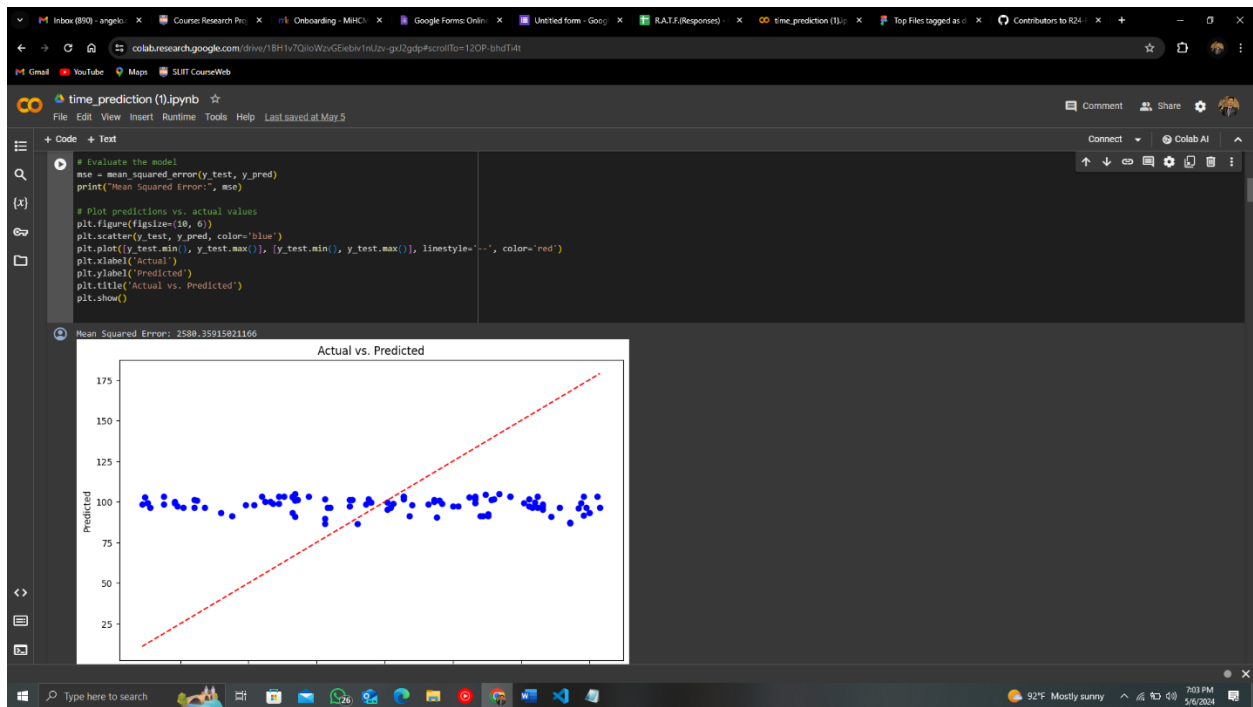


Figure 2- Actual vs predicted graph

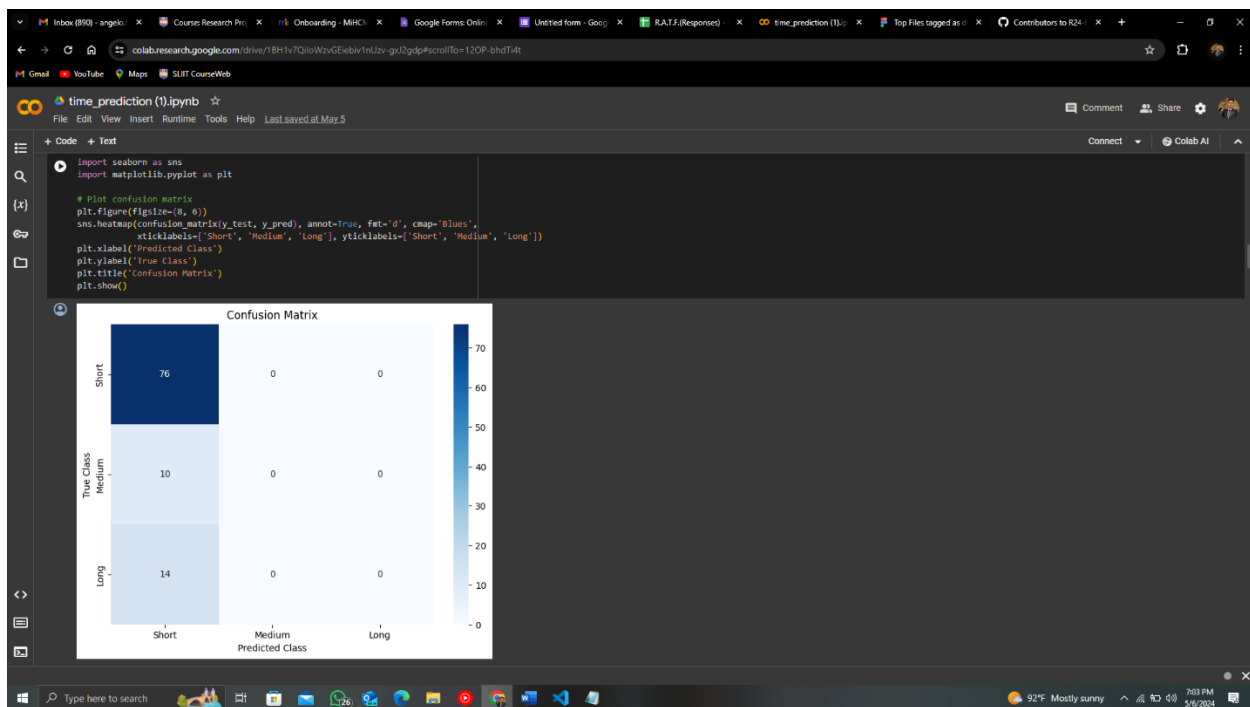


Figure 3-Confusion Matrix

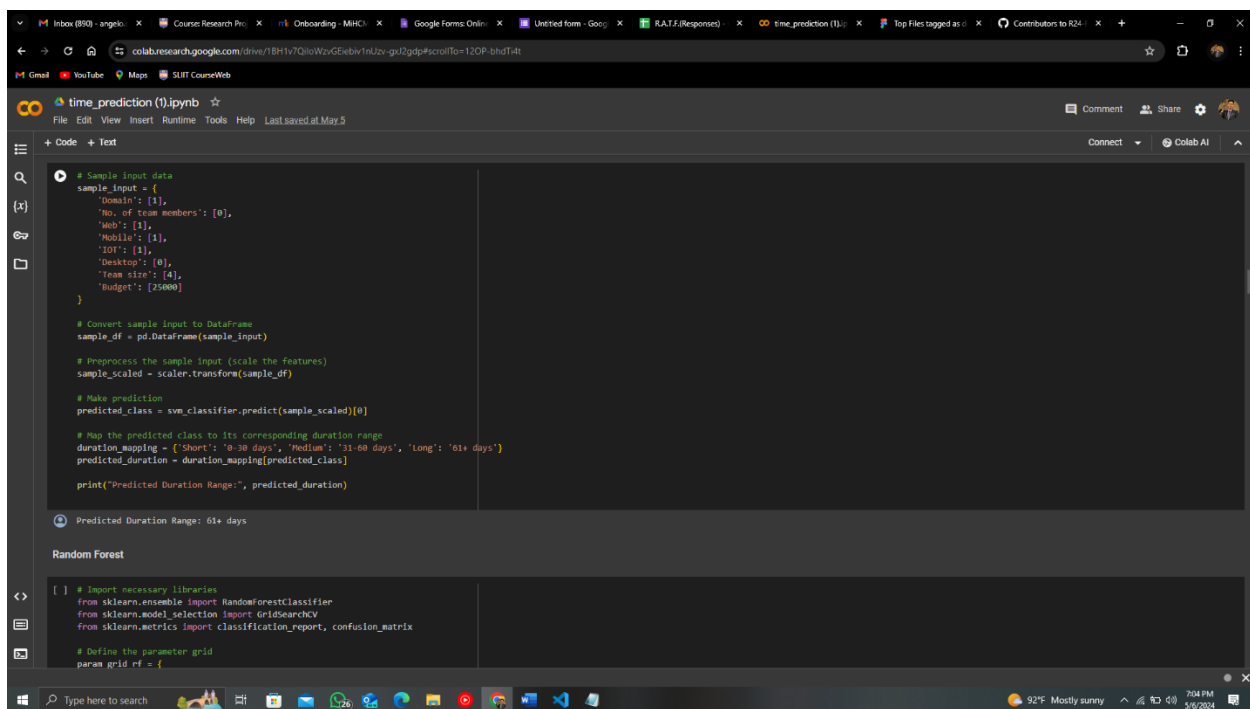


Figure 4- ML Model

```

# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix

# Define the parameter grid
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Initialize GridSearchCV
grid_search_rf = GridSearchCV(estimator=rf_classifier, param_grid=param_grid_rf, cv=5, scoring='accuracy')

# Fit the model
grid_search_rf.fit(X_train, y_train)

# Get the best parameters
best_params_rf = grid_search_rf.best_params_

# Train the model with the best parameters
best_rf_classifier = RandomForestClassifier(**best_params_rf, random_state=42)
best_rf_classifier.fit(X_train, y_train)

# Make predictions
y_pred_rf = best_rf_classifier.predict(X_test)

# Evaluate the model
print("Best Parameters:", best_params_rf)
print(classification_report(y_test, y_pred_rf))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))

```

Figure 5-Random Forest model

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

# Load the CSV file
data = pd.read_csv('/content/sample_data/Project_Dataset (1).csv') # Replace 'your_data.csv' with the actual file path

# Separate features (X) and target variable (y)
X = data.drop(['No. of days', 'No. of team members'], axis=1) # Dropping 'No. of team members' column
y = data['No. of days']

# 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)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define the parameter grid for SVR
param_grid_svr = {
    'C': [0.1, 1, 10, 100],
    'gamma': [0.01, 0.1, 1],
    'epsilon': [0.01, 0.1, 1]
}

# Initialize SVR
svr = SVR()

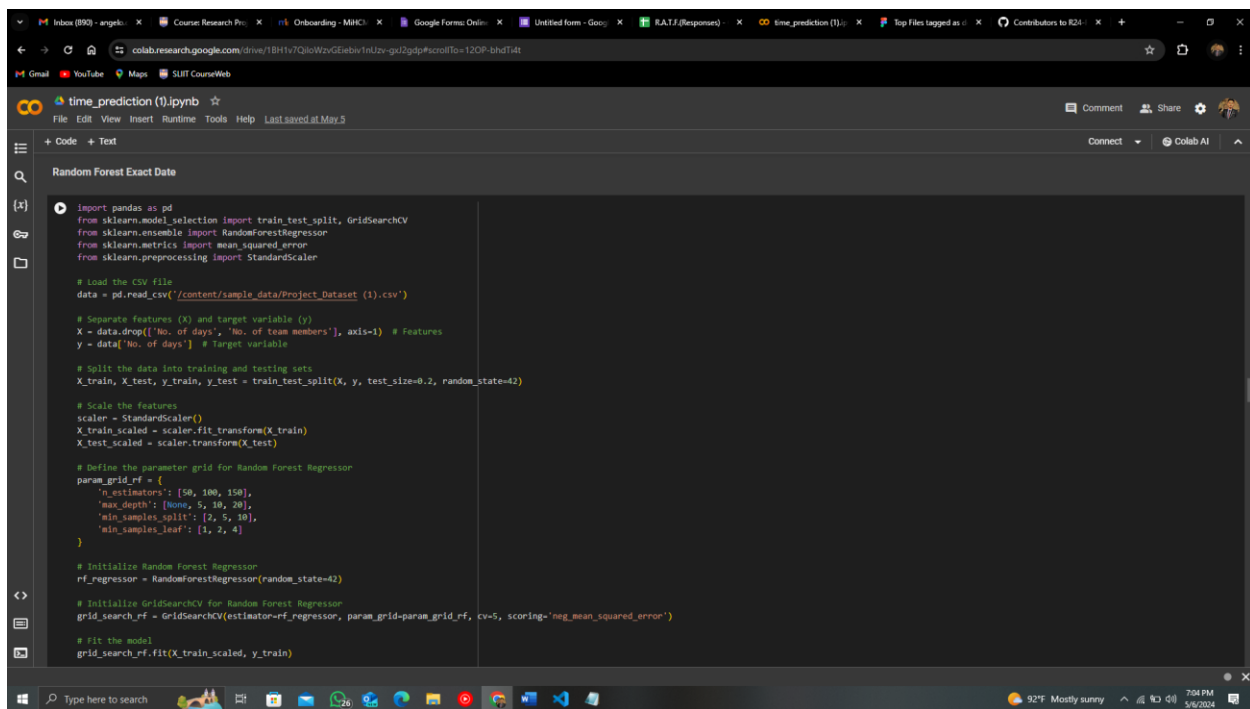
# Initialize GridSearchCV for SVR
grid_search_svr = GridSearchCV(estimator=svr, param_grid=param_grid_svr, cv=5, scoring='neg_mean_squared_error')

# Fit the model
grid_search_svr.fit(X_train_scaled, y_train)

# Get the best parameters
best_params_svr = grid_search_svr.best_params_

```

Figure 6-SVR exact date model



```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# Load the CSV file
data = pd.read_csv('/content/sample_data/Project_Dataset (1).csv')

# Separate features (X) and target variable (y)
X = data.drop(['No. of days', 'No. of team members'], axis=1) # Features
y = data['No. of days'] # Target variable

# 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)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

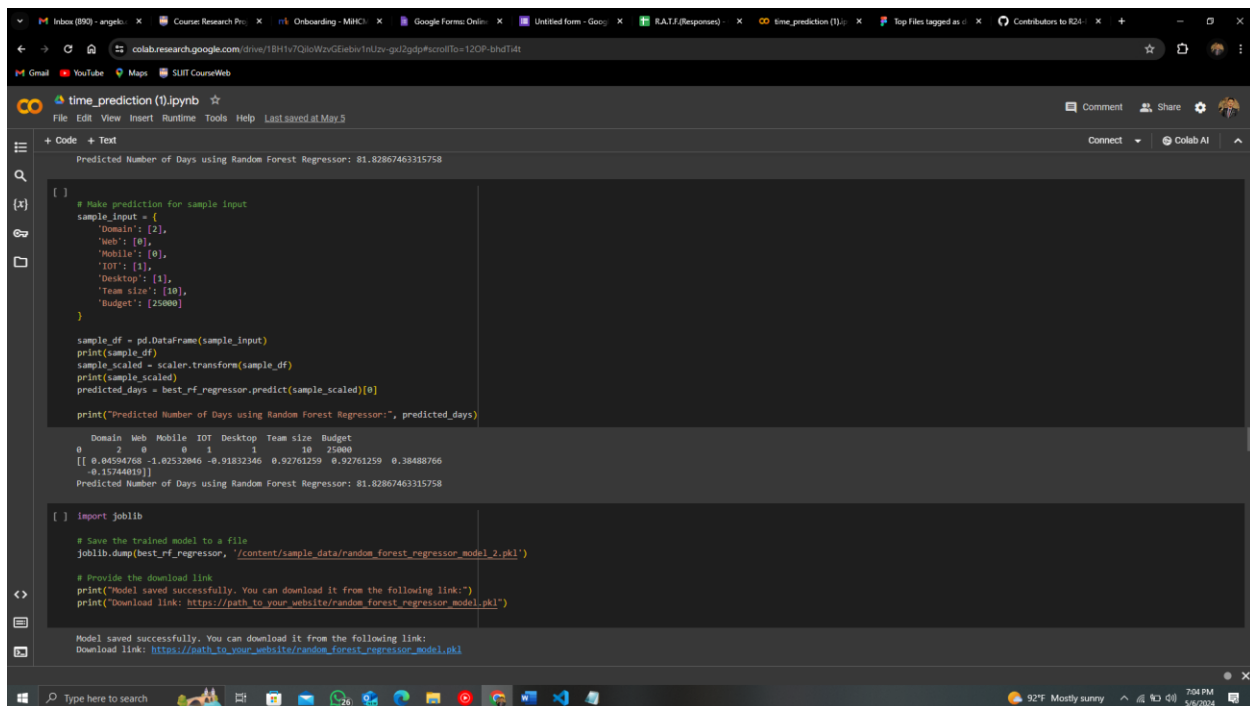
# Define the parameter grid for Random Forest Regressor
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Initialize GridSearchCV for Random Forest Regressor
grid_search_rf = GridSearchCV(estimator=rf_regressor, param_grid=param_grid_rf, cv=5, scoring='neg_mean_squared_error')

# Fit the model
grid_search_rf.fit(X_train_scaled, y_train)
```

Figure 7-Random Forest exact date



```
Predicted Number of Days using Random Forest Regressor: 81.82867463315758

[ ]
# Make prediction for sample input
sample_input = {
    'Domain': [2],
    'Web': [0],
    'Mobile': [0],
    'IoT': [1],
    'Desktop': [1],
    'Team size': [10],
    'Budget': [25000]
}

sample_df = pd.DataFrame(sample_input)
print(sample_df)
sample_scaled = scaler.transform(sample_df)
print(sample_scaled)
predicted_days = best_rf_regressor.predict(sample_scaled)[0]

print("Predicted Number of Days using Random Forest Regressor:", predicted_days)

Domain Web Mobile IoT Desktop Team size Budget
0 2 0 0 1 1 10 25000
[[ 0.04594708 -1.02532046 -0.91832346  0.92761259  0.92761259  0.38488766
  -0.15744039]]
Predicted Number of Days using Random Forest Regressor: 81.82867463315758

[ ] import joblib

# Save the trained model to a file
joblib.dump(best_rf_regressor, '/content/sample_data/random_forest_regressor_model.pkl')

# Provide the download link
print("Model saved successfully. You can download it from the following link:")
print("Download link: https://path\_to\_your\_website/random\_forest\_regressor\_model.pkl")

Model saved successfully. You can download it from the following link:
Download link: https://path\_to\_your\_website/random\_forest\_regressor\_model.pkl
```

Figure 8- RF implementation

```
[ ] import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# Load the CSV File
data = pd.read_csv('/content/sample_data/Project_Dataset (1).csv')

# Separate features (X) and target variable (y)
X = data.drop(['No. of days', 'No. of team members'], axis=1) # Features
y = data['No. of days'] # Target variable

# 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)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define the parameter grid for Random Forest Regressor
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Initialize GridSearchCV for Random Forest Regressor
grid_search_rf = GridSearchCV(estimator=rf_regressor, param_grid=param_grid_rf, cv=5, scoring='neg_mean_squared_error')

# Fit the model
grid_search_rf.fit(X_train_scaled, y_train)
```

Figure 9-Random Forest changed

```
[ ] import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler

# Load the CSV file
data = pd.read_csv('/content/sample_data/Project_Dataset (1).csv')

# Separate features (X) and target variable (y)
X = data.drop(['No. of days', 'No. of team members'], axis=1) # Features
y = data['No. of days'] # Target variable

# 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)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define the parameter grid for Random Forest Regressor
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Initialize GridSearchCV for Random Forest Regressor
grid_search_rf = GridSearchCV(estimator=rf_regressor, param_grid=param_grid_rf, cv=5, scoring='neg_mean_squared_error')

# Fit the model
grid_search_rf.fit(X_train_scaled, y_train)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Load the CSV file
data = pd.read_csv('/content/sample_data/Project_Dataset (1).csv')

# Separate features (X) and target variable (y)
X = data.drop(['No. of days', 'No. of team members'], axis=1) # Features
y = data['No. of days'] # Target variable

# 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)

# Define the parameter distributions for Random Forest Regressor
param_dist_rf = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [None, 5, 10, 20, 30],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['auto', 'sqrt', 'log2'],
    'bootstrap': [True, False]
}

# Initialize Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Initialize RandomizedSearchCV for Random Forest Regressor
random_search_rf = RandomizedSearchCV(estimator=rf_regressor, param_distributions=param_dist_rf, n_iter=100, cv=5, scoring='neg_mean_squared_error', random_state=42)

# Fit the model
random_search_rf.fit(X_train, y_train)

# Get the best parameters
best_params_rf = random_search_rf.best_params_
```

Figure 10-RF without scale

```
[ ] sample_input = [[2, 1, 1, 1, 1, 20, 23000]]
predicted_days = best_rf_regressor.predict(sample_input)[0]

print("Predicted Number of Days using Random Forest Regressor:", predicted_days)

Predicted Number of Days using Random Forest Regressor: 181.19177117927282
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(

[ ] import joblib

# Define the file path to save the model
model_file_path = '/content/sample_data/random_forest_regressor_model_final.pkl'

# Save the trained model to a file
joblib.dump(best_rf_regressor, model_file_path)

print("Model saved successfully.")

Model saved successfully.

[ ] import joblib

# Define the file path where the model is saved
model_file_path = '/content/sample_data/random_forest_regressor_model_final.pkl'

# Load the saved model from the file
loaded_rf_regressor = joblib.load(model_file_path)

# Make prediction for sample input
sample_input = [[2, 1, 1, 1, 0, 6, 23000]]
predicted_days = loaded_rf_regressor.predict(sample_input)[0]

print("Predicted Number of Days using Loaded Random Forest Regressor:", predicted_days)

Predicted Number of Days using Loaded Random Forest Regressor: 103.0875562650835
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

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. Regular team conversations were arranged to discuss, share knowledge, and plan.

The crew also used WhatsApp as an additional tool to stay in constant communication with their supervisors. This enabled timely updates and cooperation between the supervisor and co-supervisor, ensuring that everyone was informed and on the same page throughout the project.

Teams Channel

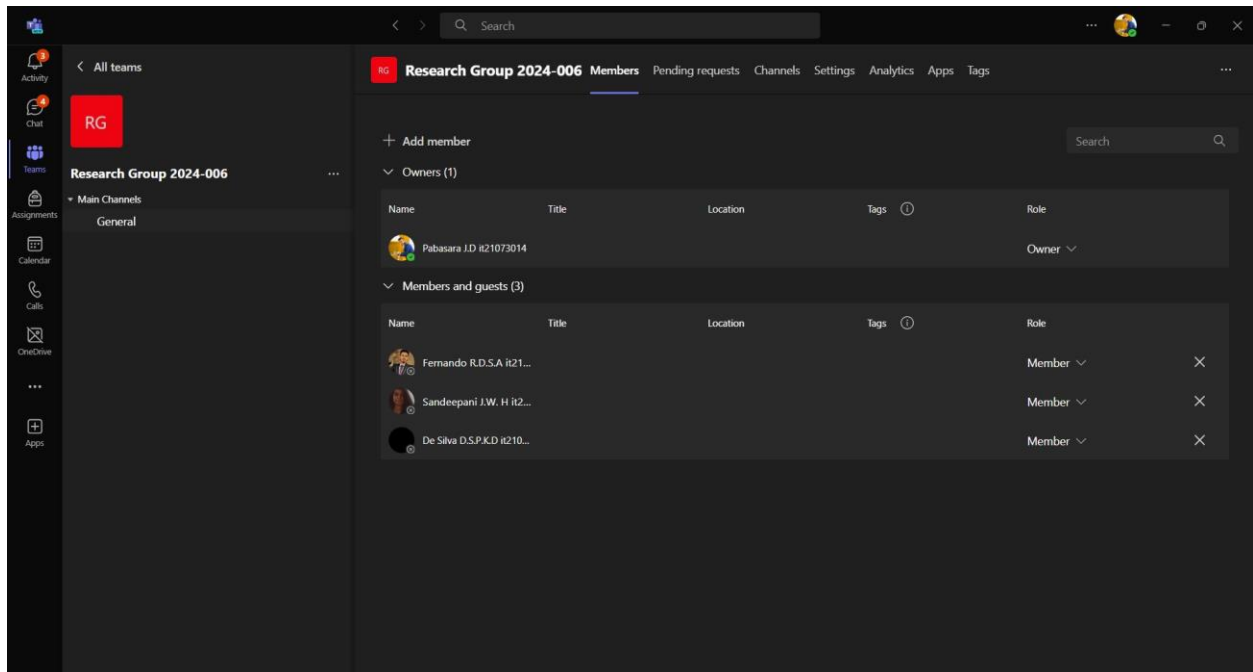


Figure 11 - Teams Channel Creation

Teams Calls with the Research Team

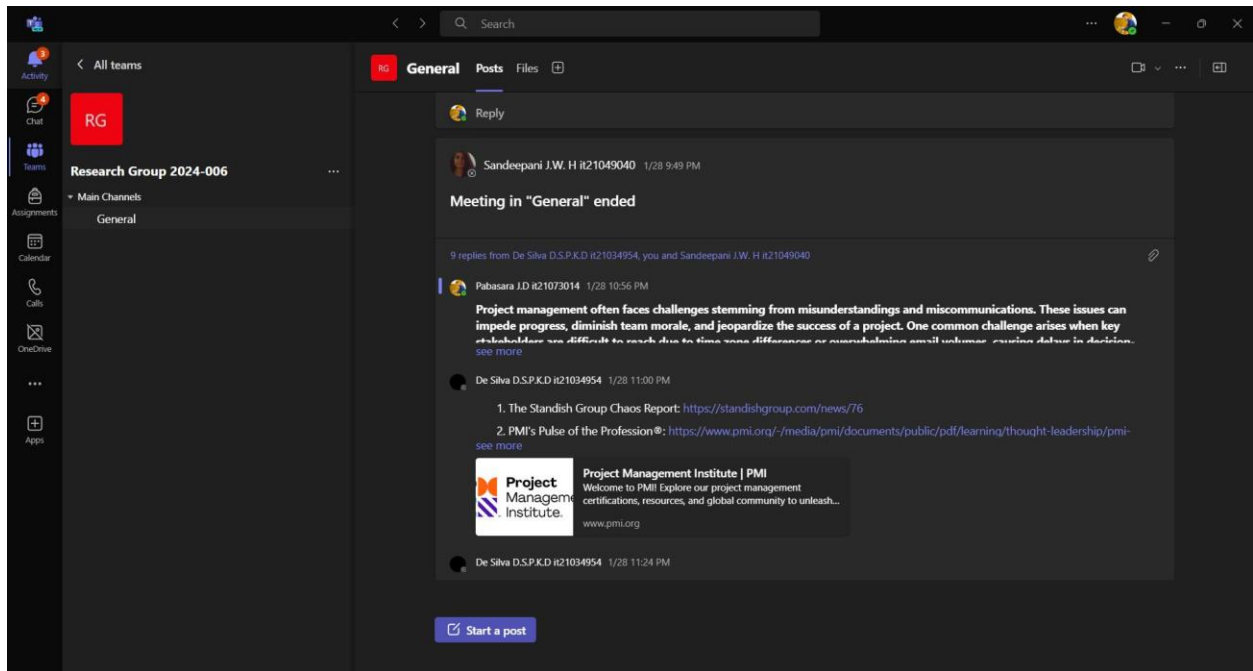


Figure 12 - Overview of Team Calls and Communication

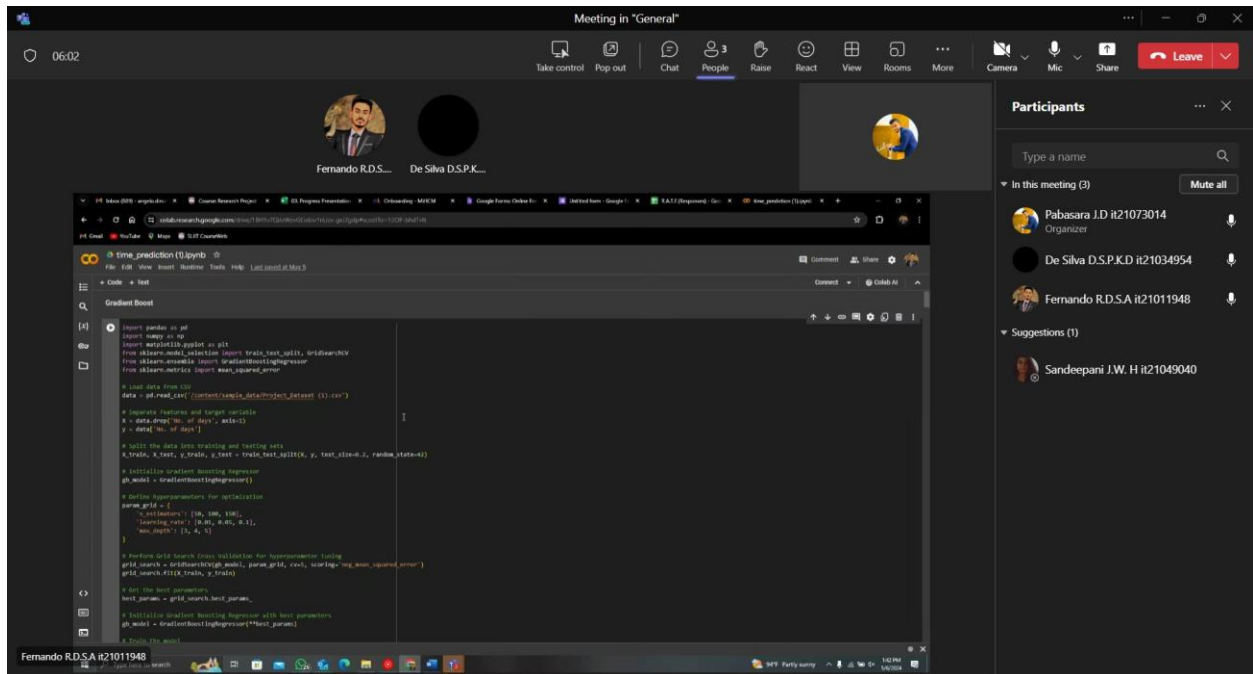


Figure 13 - Team Member Teams Calls (Example 01)

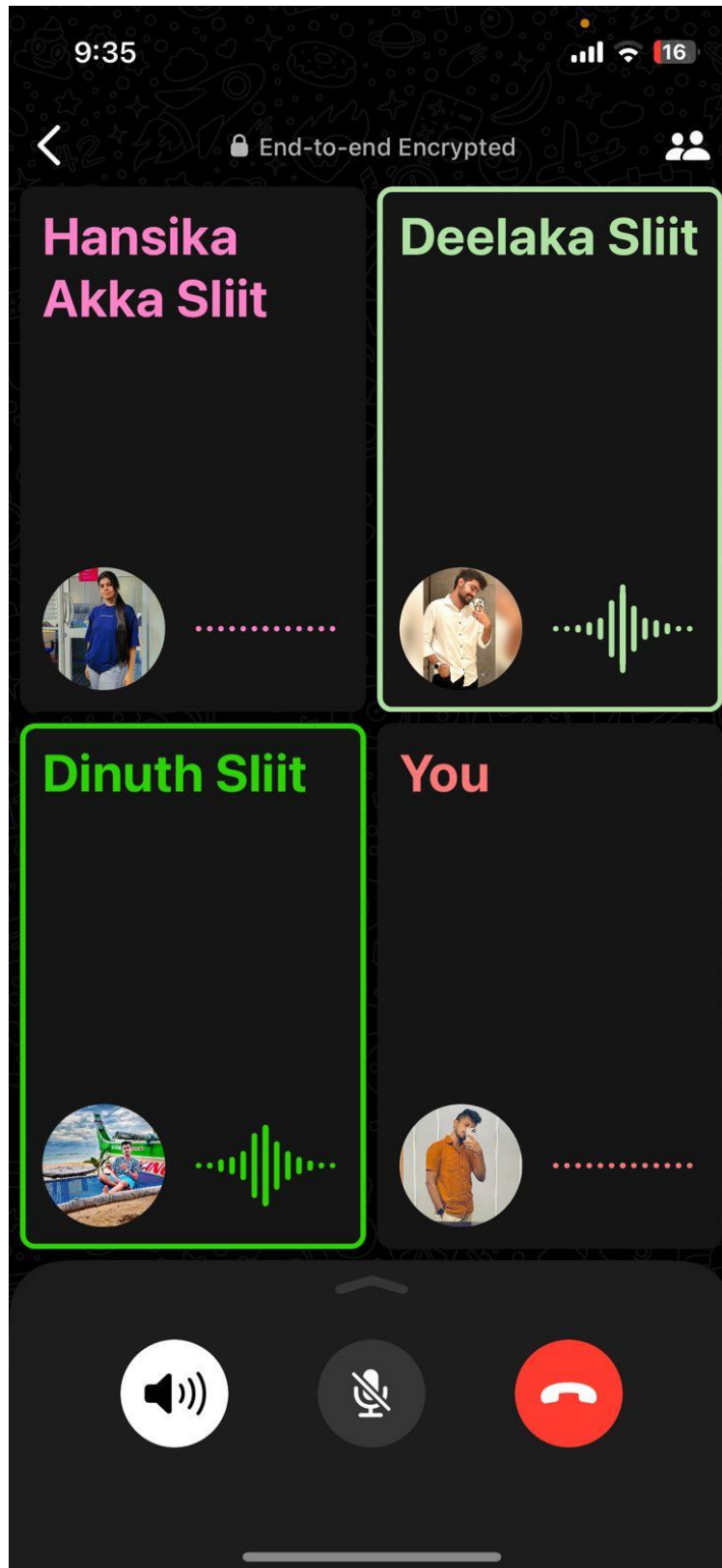


Figure 14 - Team Member WhatsApp Calls (Example 02)

Online Calls with Supervisors

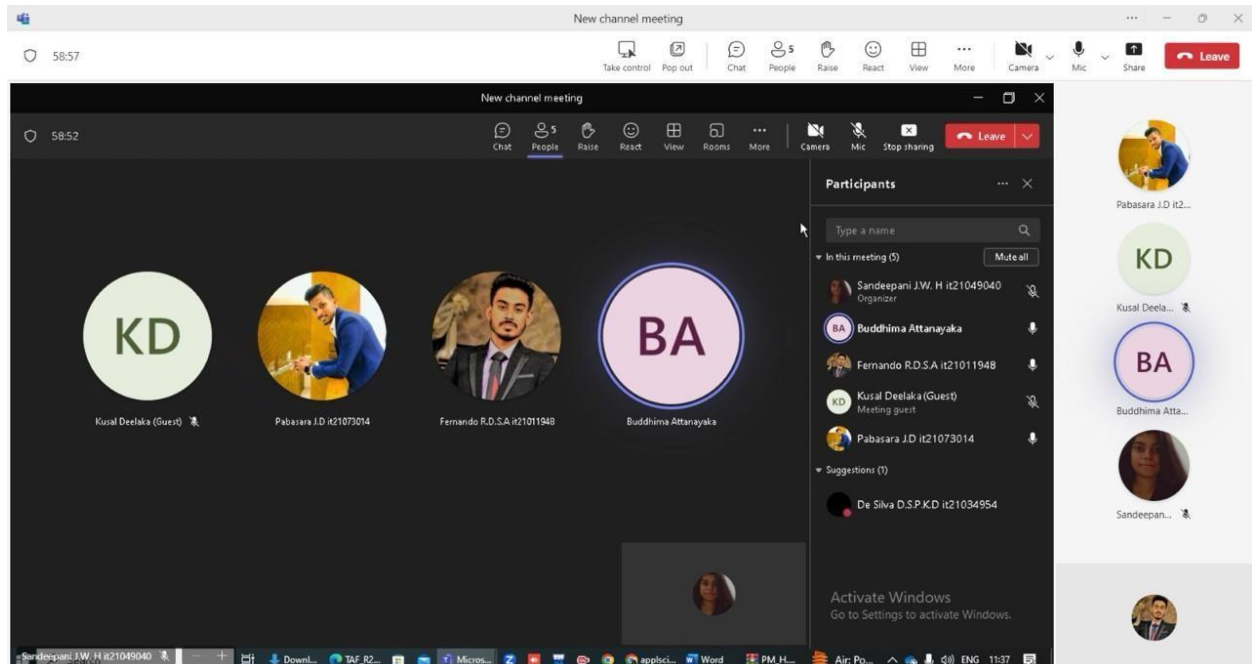


Figure 15 - Teams Meeting with the Co-Supervisor (Example 01)

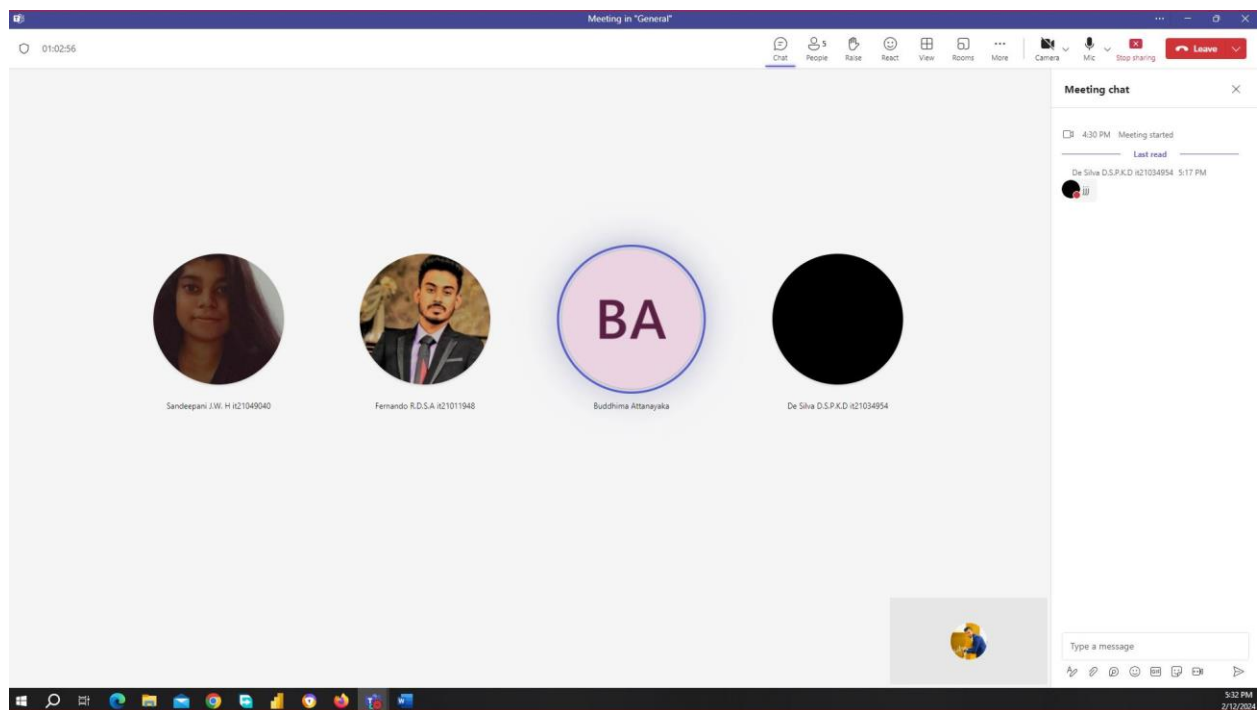


Figure 16 - Teams Meeting with the Co-supervisor (Example 02)

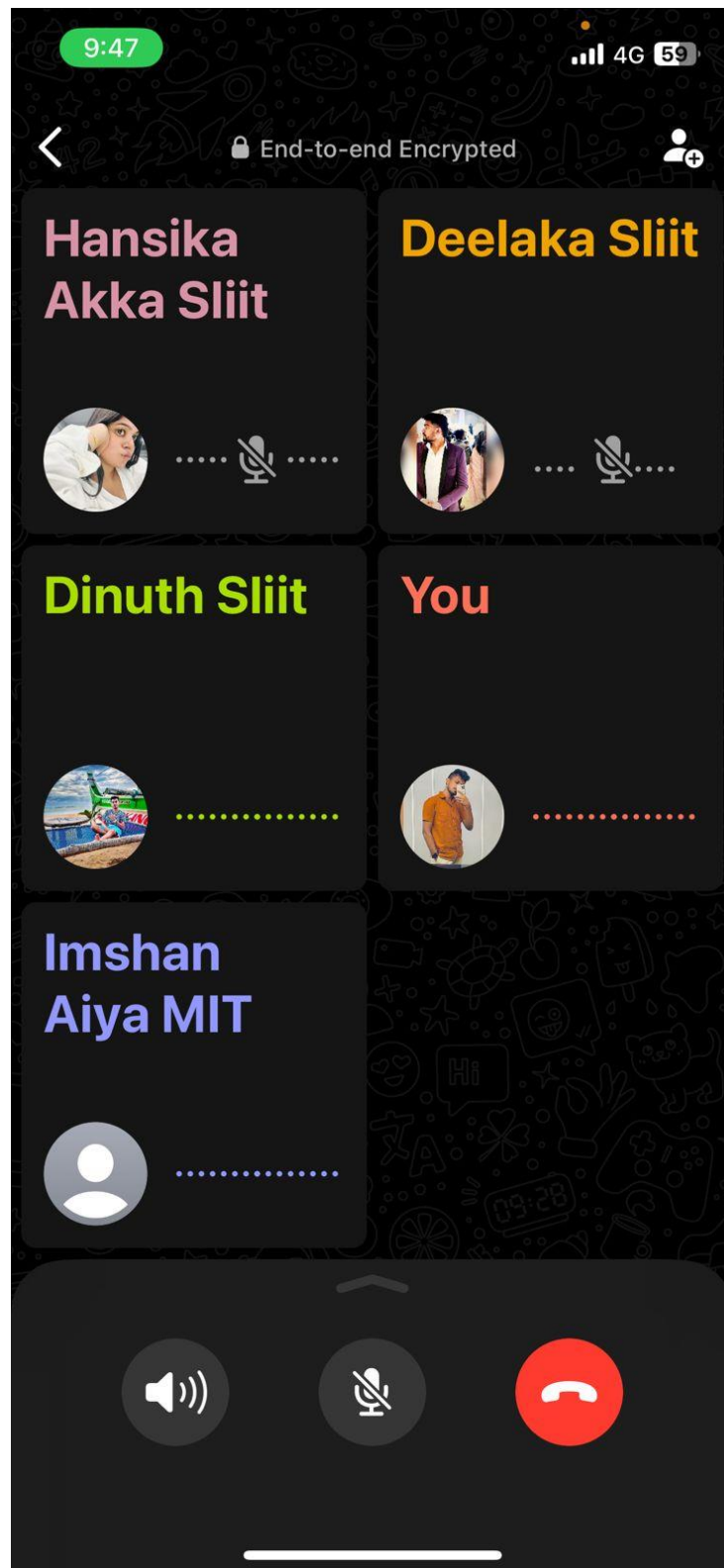


Figure 17 - WhatsApp Call with External Supervisor (LSEG)

Physical Meetings with Group Members



Figure 18 - Physical Meetings with team members.

WhatsApp Group Creation

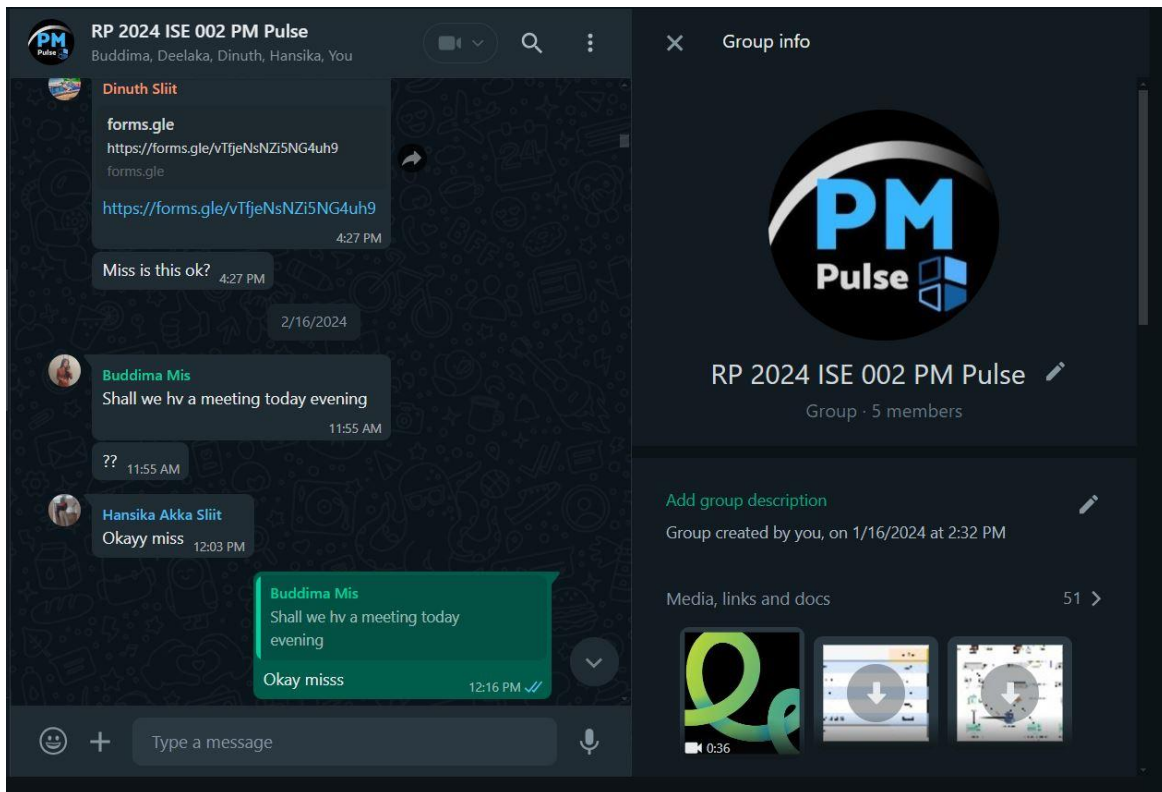
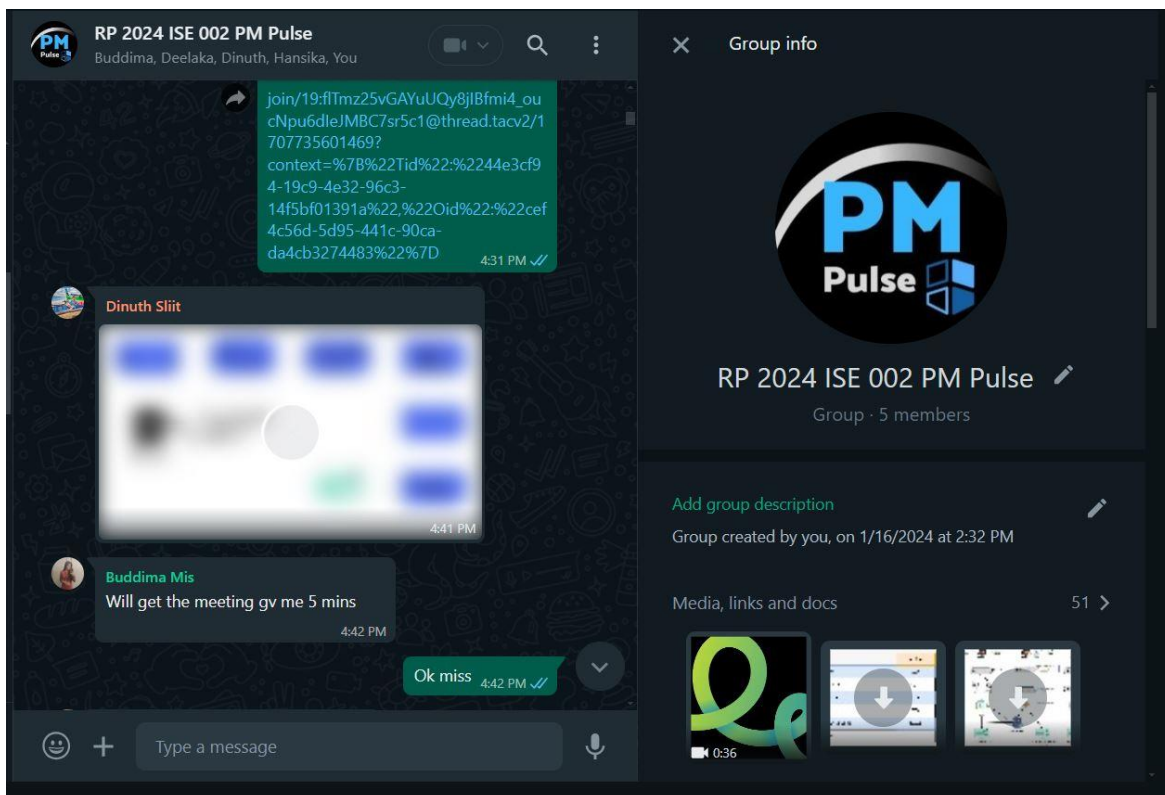


Figure 19 - WhatsApp Group Creation with Co-Supervisor



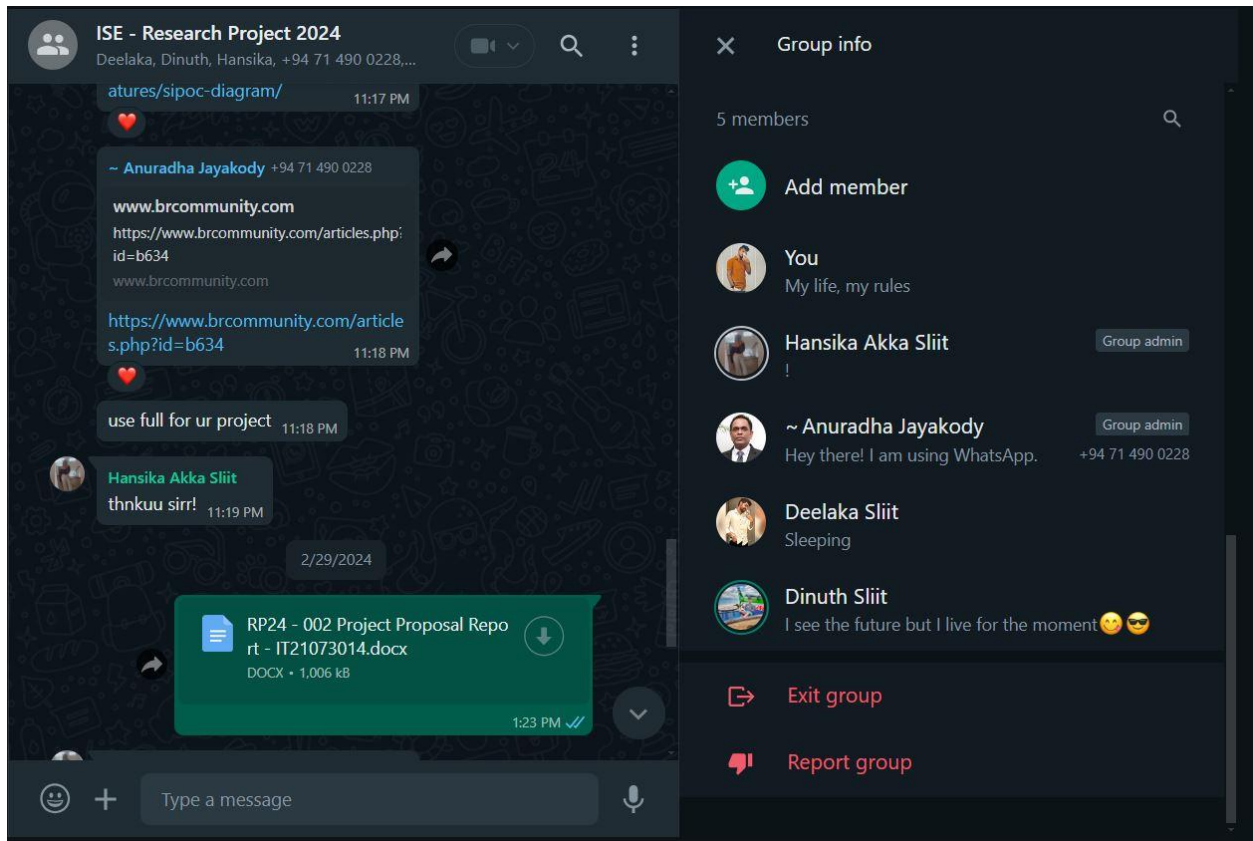


Figure 20 - WhatsApp Group Creation with Supervisor

Project Timeline

We put together a Gantt chart that outlines the entire timeline of our project, encompassing all the deadlines for our deliverables. This visual representation provides a comprehensive overview of our project's progression from its inception to its completion.

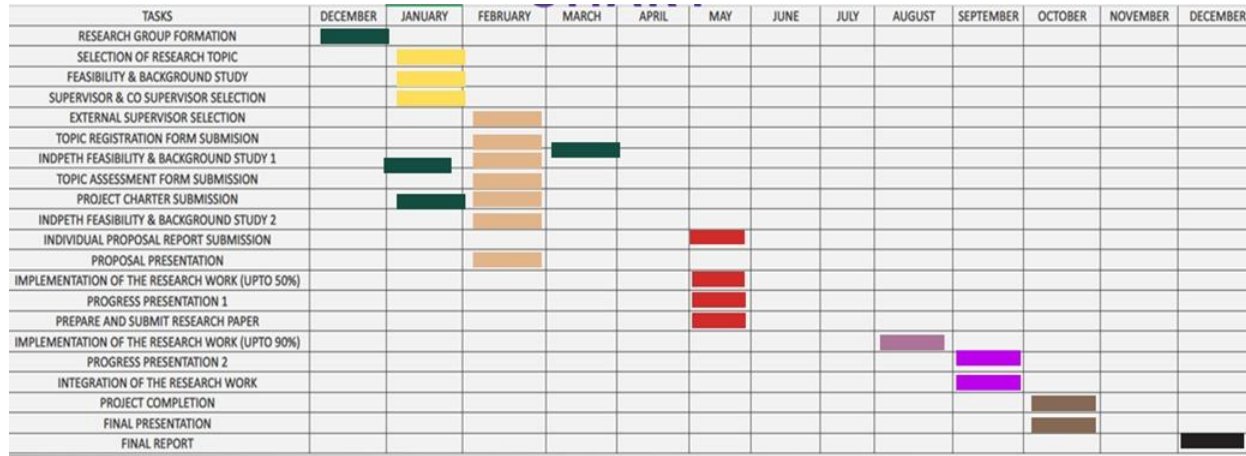


Figure 21 - Gantt Chart (Project Timeline)

Work Break-Down

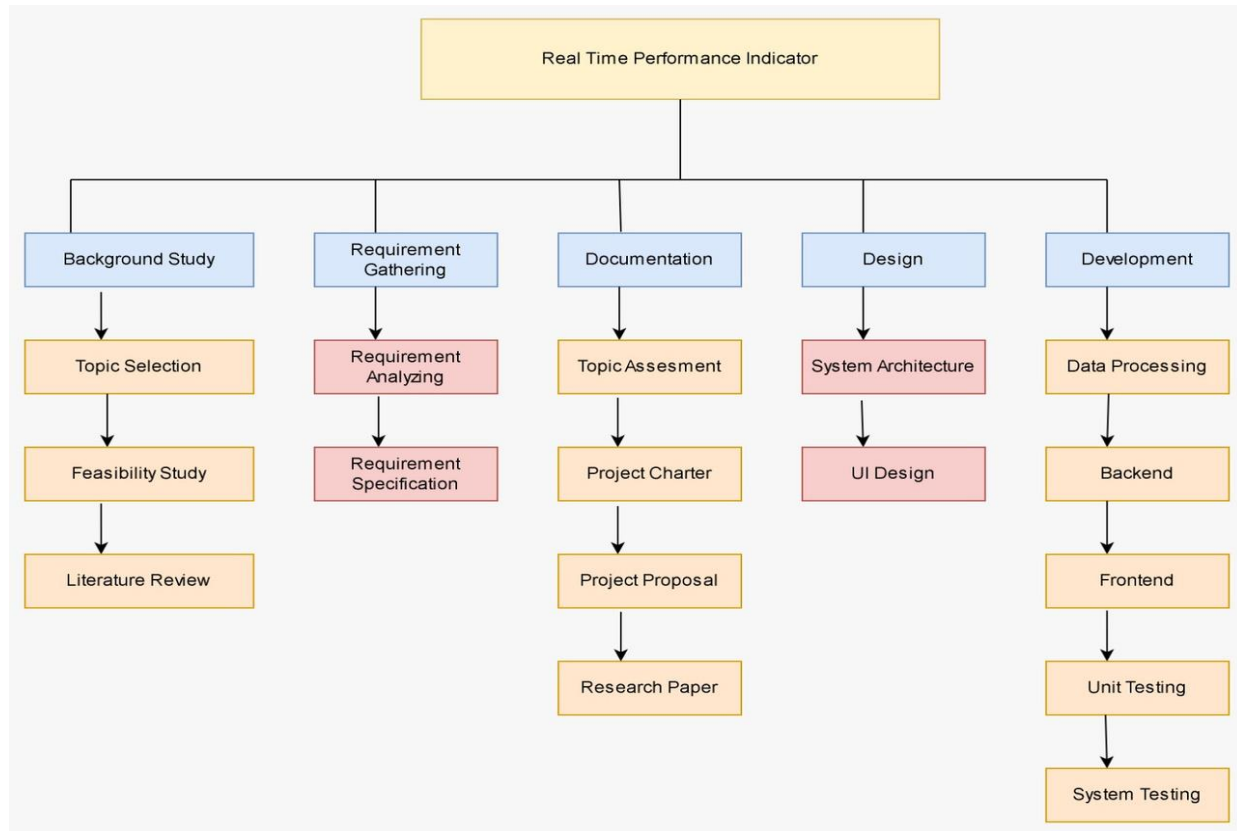


Figure 22 - Work Break-Down Structure

Version Controlling

The team implemented GitHub as their version control system, creating a repository that included all four group members. Each member was responsible for committing their code changes and progress to the repository.

To ensure transparency and collaboration, all code changes were incrementally committed and pushed to individual branches. These branches were later merged into the main branch during weekly meetings. This approach allowed for effective version control and streamlined collaboration among team members.

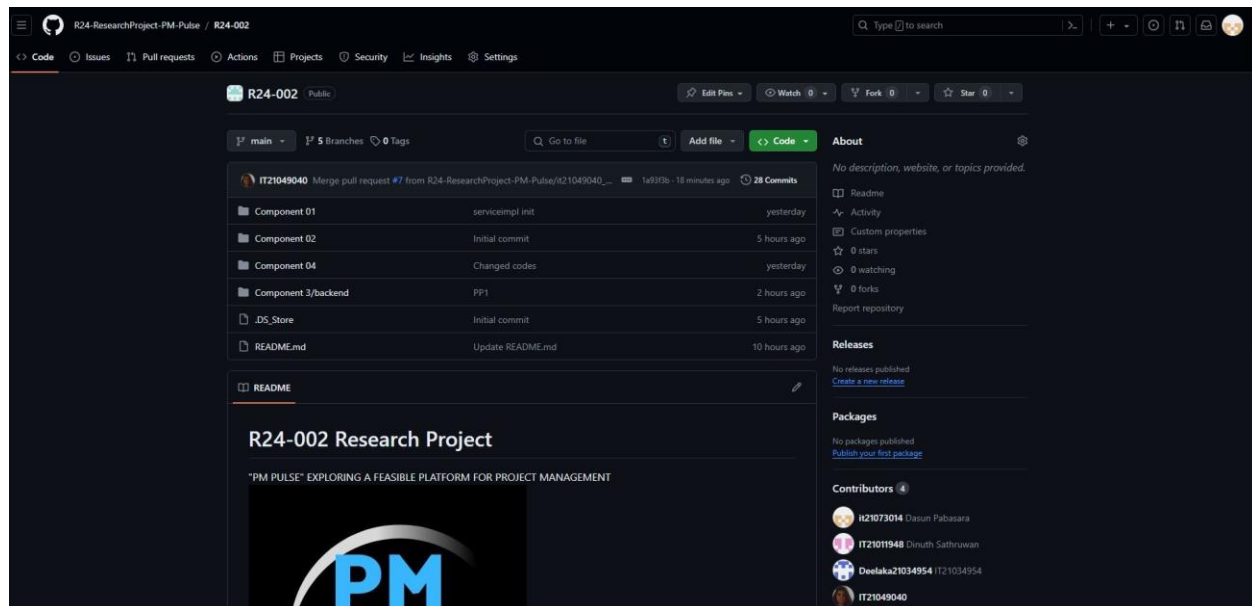


Figure 23 - GitHub Repository

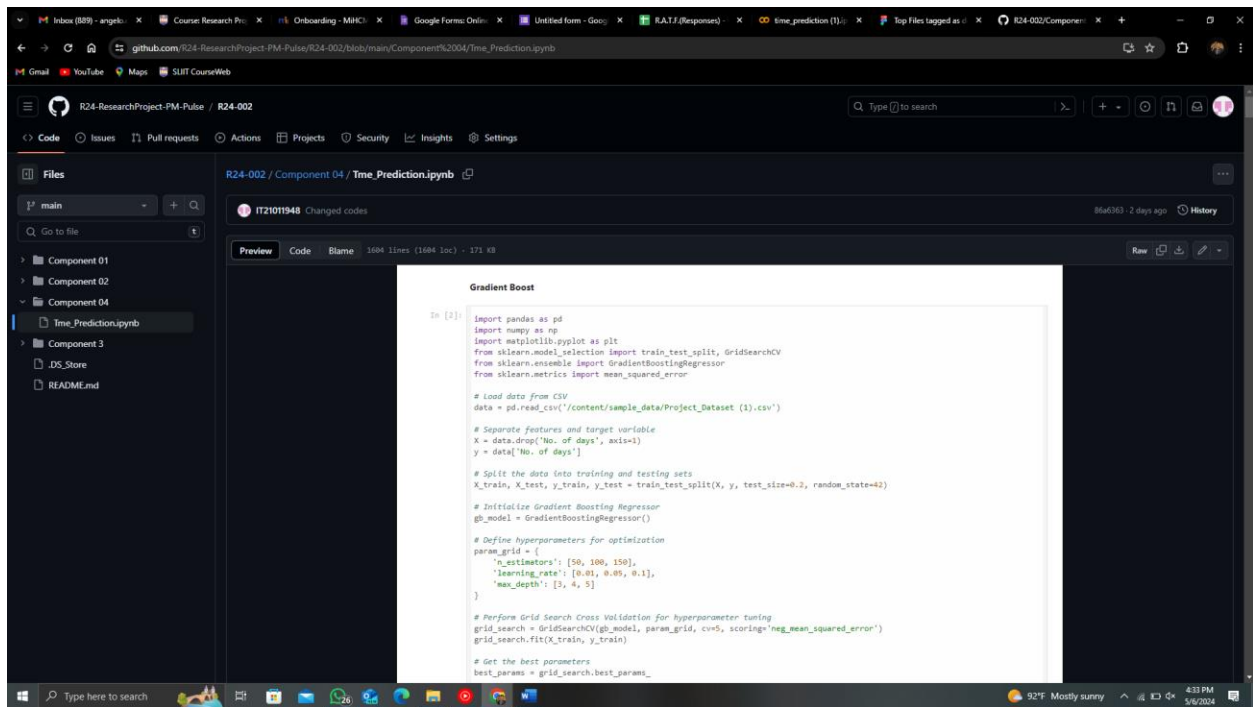
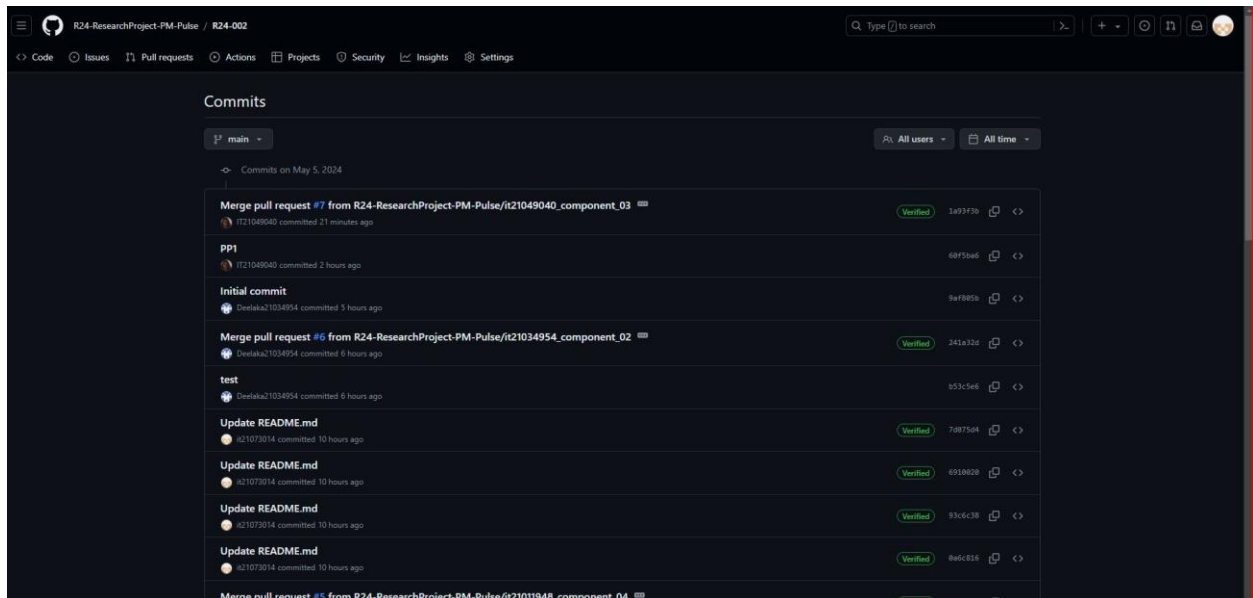


Figure 24 - GitHub Repository (Component 04 Branch)

GitHub Commits



GitHub Graph of Commits

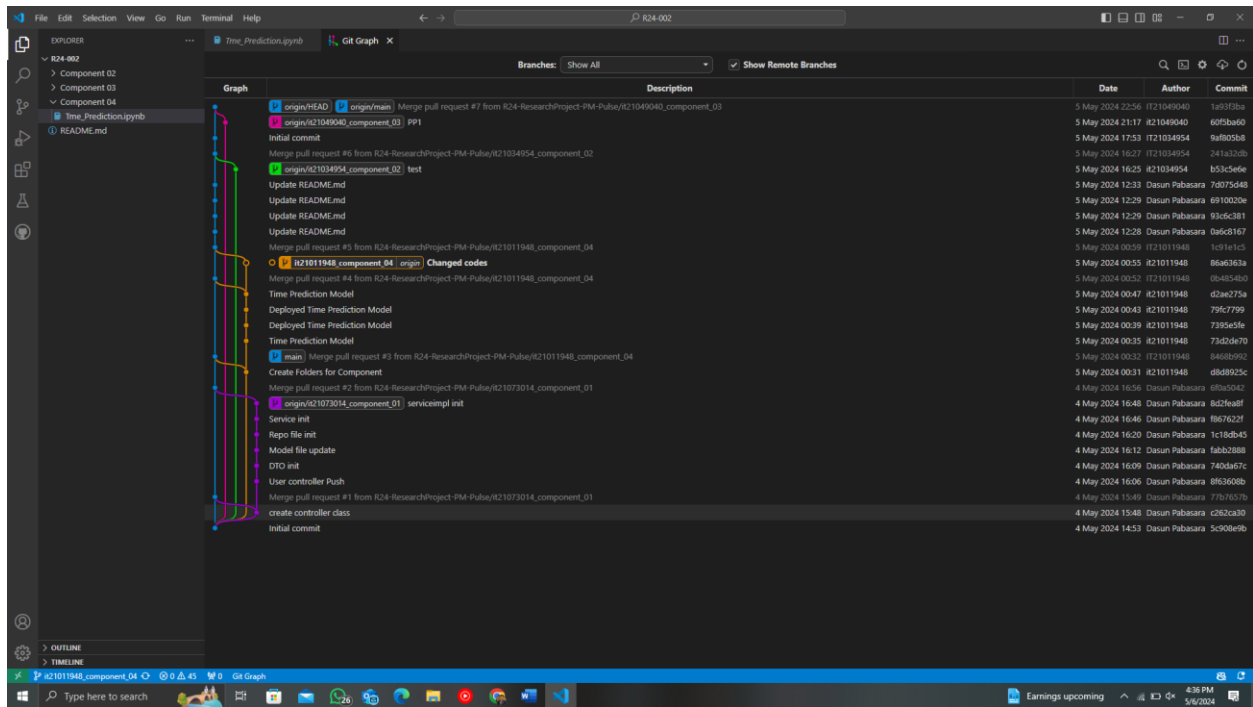
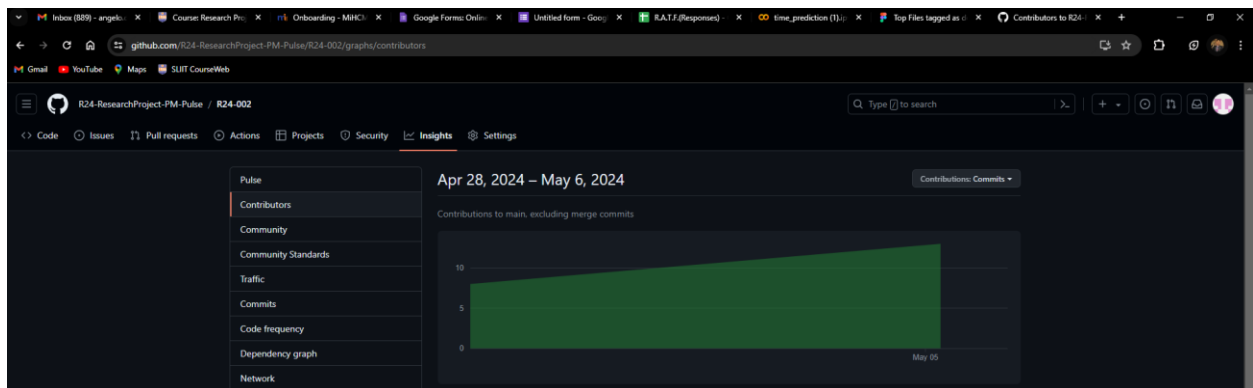


Figure 26 - Git Graph

GitHub Contributors





Contribution Charts

Figure 27 - GitHub Contributors

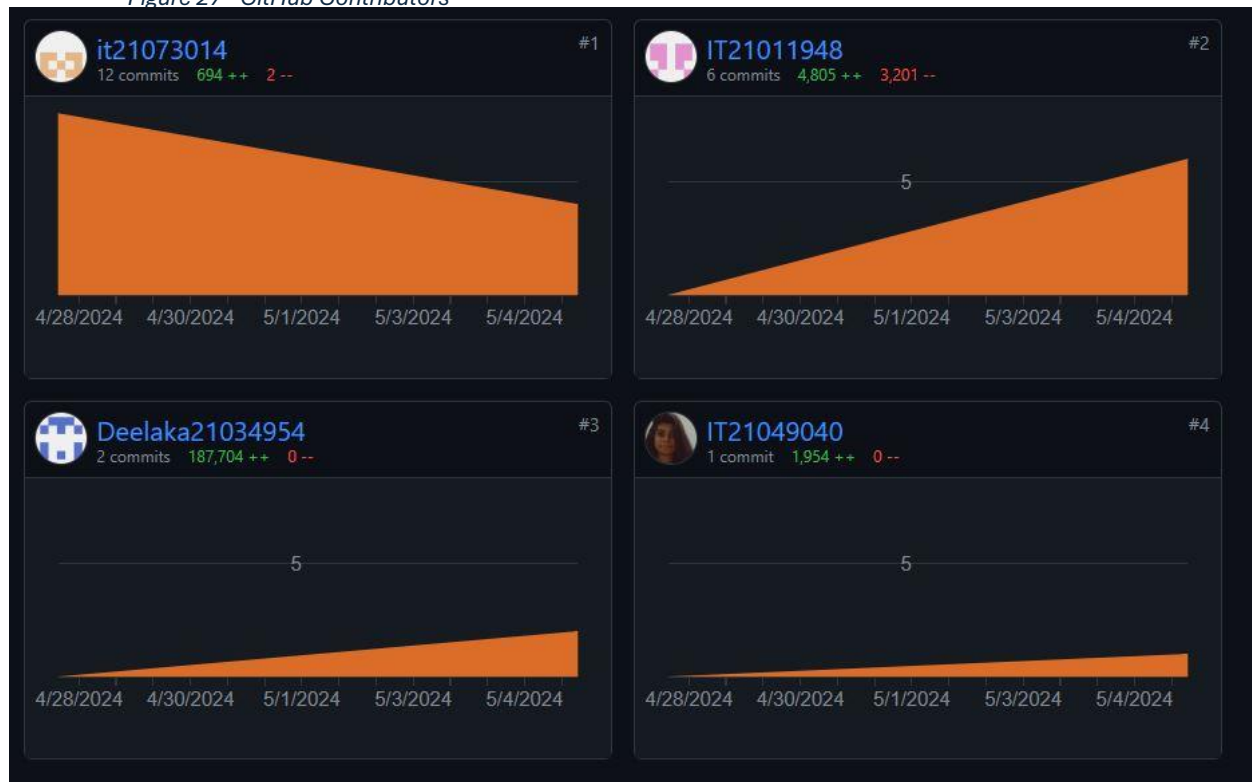


Figure 28 - Contributors Charts