

MACHINE LEARNING APPROACH TO DETECT & ANNOTATE DISEASES USING RETINAL IMAGGES

2023-162

Status Document II

Lakshith G. P. R.

IT20165666

B.Sc. (Hons) Degree in Information Technology
Specializing in Software Engineering

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

September 2023

Table of Contents

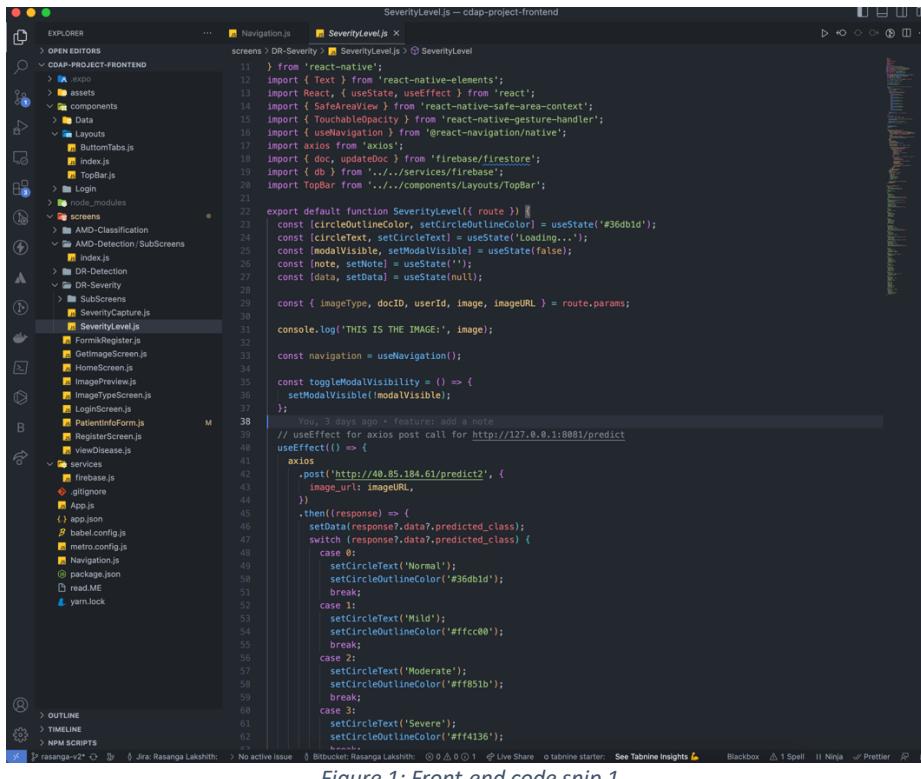
1.	PROJECT PROGRESS	3
1.1.	FRONT-END IMPLEMENTATION.....	3
1.2.	BACK-END IMPLEMENTATION	4
1.3.	MOBILE APP UI DESIGNS	6
2.	PROJECT VIEW	7
3.	GANTT CHART	8
4.	SCREENSHOTS OF CONVERSATIONS AND CALLS – MS TEAMS.....	8

Table of Figures

Figure 1: Front-end code snip 1	3
Figure 2: Front-end code snip 2	3
Figure 3: Back-end Flask code.....	4
Figure 4: Back-end Service code	4
Figure 5: Custom made lightweight model.....	5
Figure 6: Preprocessed images	5
Figure 7: Planner - board view.....	7
Figure 8: Planner - chart view.....	7
Figure 9: Planner - schedule view.....	7
Figure 10: Gantt chart.....	8

1. Project progress

1.1. Front-end Implementation



```
SeverityLevel.js - cdap-project-frontend
Navigation.js SeverityLevel.js X
screens > DR-Severity > SeverityLevel.js > SeverityLevel
11 } from 'react-native';
12 import { Text } from 'react-native-elements';
13 import React, { useState, useEffect } from 'react';
14 import { SafeAreaView } from 'react-native-safe-area-context';
15 import { Touchableopacity } from 'react-native-gesture-handler';
16 import { useNavigation } from '@react-navigation/native';
17 import axios from 'axios';
18 import { doc, updateDoc } from 'firebase/firestore';
19 import { db } from '../../../../../services/firebase';
20 import Topbar from '../../../../../components/Layouts/topBar';
21
22 export default function SeverityLevel({ route }) {
23   const [circleOutlineColor, setCircleOutlineColor] = useState('#36dbbd');
24   const [circleText, setCircleText] = useState('Loading...');

  ... (remaining code)

  // useEffect for axios post call for http://127.0.0.1:8081/predict
  useEffect(() => {
    axios
      .post(`http://40.85.184.61/predict2`, {
        image_url: imageURL
      })
      .then(response) => {
        setData(response.data.predicted_class);
        switch (response.data.predicted_class) {
          case 0:
            setCircleText('Normal');
            setCircleOutlineColor('#36dbbd');
            break;
          case 1:
            setCircleText('Mild');
            setCircleOutlineColor('#ffcc00');
            break;
          case 2:
            setCircleText('Moderate');
            setCircleOutlineColor('#ff851b');
            break;
          case 3:
            setCircleText('Severe');
            setCircleOutlineColor('#ff4136');
            break;
        }
      }
  });
}

const handleAddNote = async () => {
  try {
    const docRef = doc(db, 'Patients', docID);

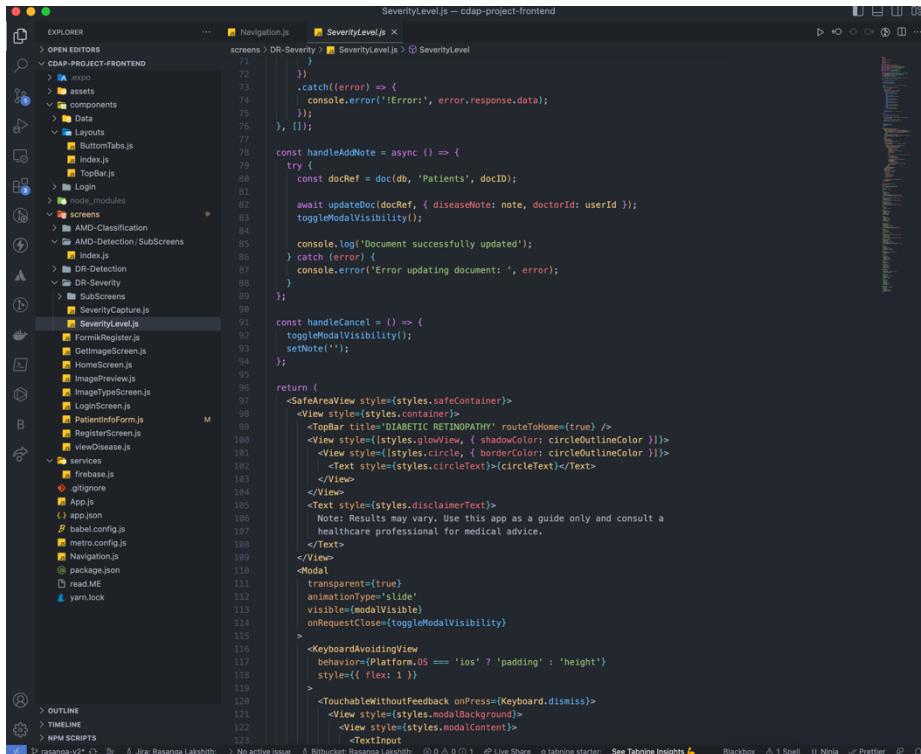
    await updateDoc(docRef, { diseaseNote: note, doctorId: userId });

    console.log('Document successfully updated');
  } catch (error) {
    console.error('Error updating document: ', error);
  }
};

const handleCancel = () => {
  toggleModalVisibility();
  setNote('');
};

return (
  <SafeAreaView style={styles.safeContainer}>
    <View style={styles.container}>
      <Topbar title='DIABETIC RETINOPATHY' routeToHome={true} />
      <View style={styles.glowView, { shadowColor: circleOutlineColor }}>
        <Text style={styles.circle, { borderColor: circleOutlineColor }}><circleText></circleText></Text>
      </View>
      <Text style={styles.disclaimerText}>
        Note: Results may vary. Use this app as a guide only and consult a healthcare professional for medical advice.
      </Text>
    </View>
    <Modal
      transparent={true}
      animationType='slide'
      visible={modalVisible}
      onRequestClose={toggleModalVisibility}
    >
      <KeyboardAvoidingView
        behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
        style={{ flex: 1 }}
      >
        <TouchableWithoutFeedback onPress={Keyboard.dismiss}>
          <View style={styles.modalBackground}>
            <View style={styles.modalContent}>
              <TextInput
                style={styles.modalInput}
              >
```

Figure 1: Front-end code snip 1



```
SeverityLevel.js - cdap-project-frontend
Navigation.js SeverityLevel.js X
screens > DR-Severity > SeverityLevel.js > SeverityLevel
11 }
12 )
13 .catch((error) => {
14   console.error('!Error', error.response.data);
15 });
16 }, []);

const handleAddNote = async () => {
  try {
    const docRef = doc(db, 'Patients', docID);

    await updateDoc(docRef, { diseaseNote: note, doctorId: userId });

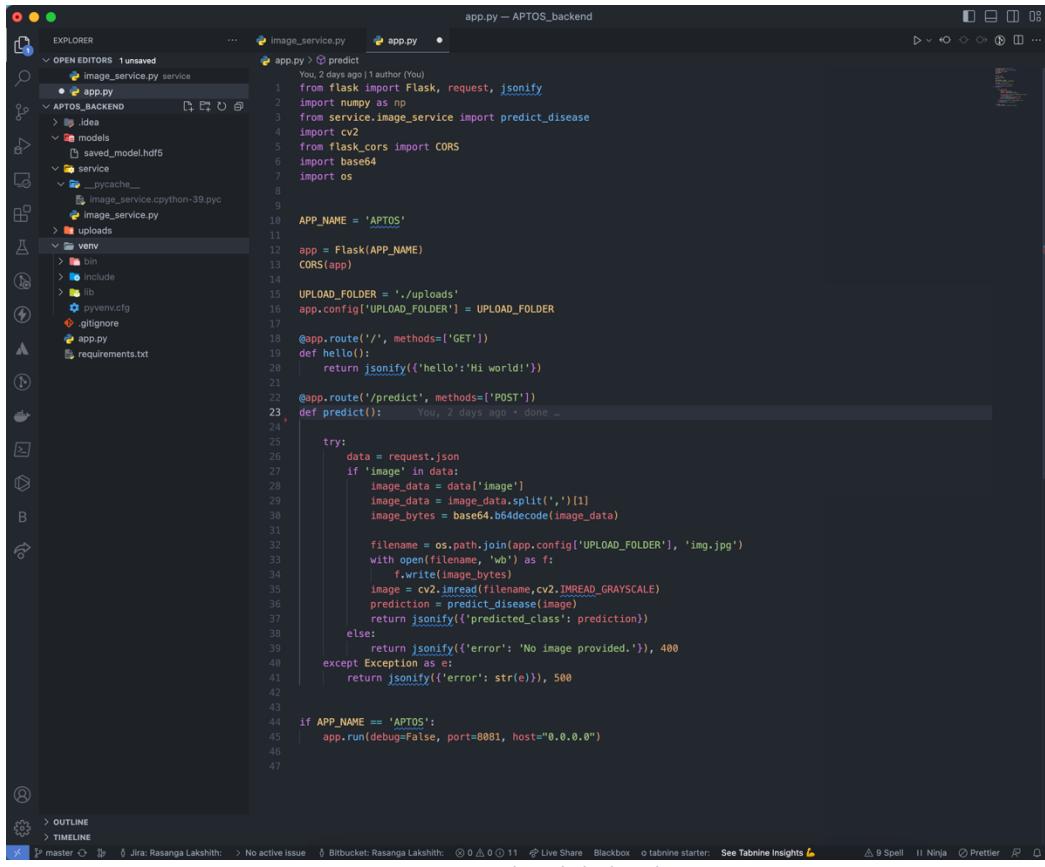
    console.log('Document successfully updated');
  } catch (error) {
    console.error('Error updating document: ', error);
  }
};

const handleCancel = () => {
  toggleModalVisibility();
  setNote('');
};

return (
  <SafeAreaView style={styles.safeContainer}>
    <View style={styles.container}>
      <Topbar title='DIABETIC RETINOPATHY' routeToHome={true} />
      <View style={styles.glowView, { shadowColor: circleOutlineColor }}>
        <Text style={styles.circle, { borderColor: circleOutlineColor }}><circleText></circleText></Text>
      </View>
      <Text style={styles.disclaimerText}>
        Note: Results may vary. Use this app as a guide only and consult a healthcare professional for medical advice.
      </Text>
    </View>
    <Modal
      transparent={true}
      animationType='slide'
      visible={modalVisible}
      onRequestClose={toggleModalVisibility}
    >
      <KeyboardAvoidingView
        behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
        style={{ flex: 1 }}
      >
        <TouchableWithoutFeedback onPress={Keyboard.dismiss}>
          <View style={styles.modalBackground}>
            <View style={styles.modalContent}>
              <TextInput
                style={styles.modalInput}
              >
```

Figure 2: Front-end code snip 2

1.2. Back-end Implementation



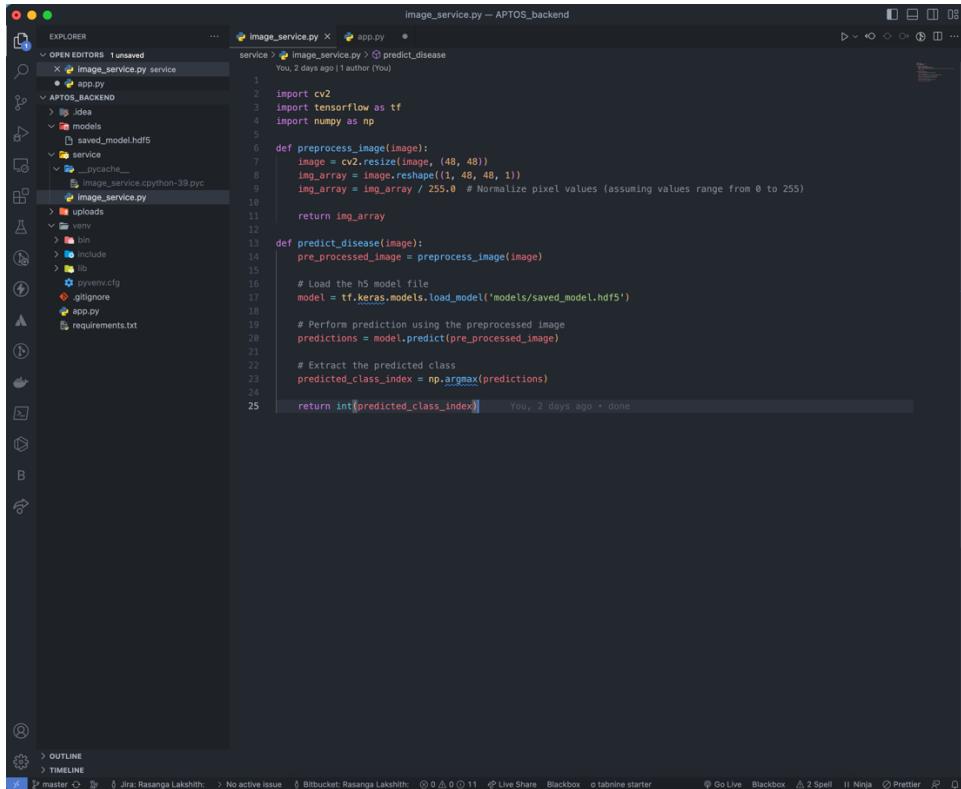
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure with files like `image_service.py`, `app.py`, and `requirements.txt`.
- Editor:** Displays the `app.py` file content, which is the main Flask application code.
- Bottom Status Bar:** Shows the current file is `app.py — APTOS_backend`, along with other status indicators like Jira, Bitbucket, and GitHub links.

```
image_service.py -- APTOS_backend
app.py -- APTOS_backend

You, 2 days ago | 1 author (You)
1 from flask import Flask, request, jsonify
2 import numpy as np
3 from service.image_service import predict_disease
4 import cv2
5 from flask_cors import CORS
6 import base64
7 import os
8
9
10 APP_NAME = 'APTO5'
11
12 app = Flask(APP_NAME)
13 CORS(app)
14
15 UPLOAD_FOLDER = './uploads'
16 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
17
18 @app.route('/', methods=['GET'])
19 def hello():
20     return jsonify({'hello': 'Hi world!'})
21
22 @app.route('/predict', methods=['POST'])
23 def predict():
24     try:
25         data = request.json
26         if 'image' in data:
27             image_data = data['image']
28             image_data = image_data.split(',')
29             image_data = image_data[1]
30             image_bytes = base64.b64decode(image_data)
31
32             filename = os.path.join(app.config['UPLOAD_FOLDER'], 'img.jpg')
33             with open(filename, 'wb') as f:
34                 f.write(image_bytes)
35             image = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
36             prediction = predict_disease(image)
37             return jsonify({'predicted_class': prediction})
38         else:
39             return jsonify({'error': 'No image provided.'}), 400
40     except Exception as e:
41         return jsonify({'error': str(e)}), 500
42
43
44 if APP_NAME == 'APTO5':
45     app.run(debug=False, port=8081, host="0.0.0.0")
46
47
```

Figure 3: Back-end Flask code



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure with files like `image_service.py`, `app.py`, and `requirements.txt`.
- Editor:** Displays the `image_service.py` file content, which contains the logic for preprocessing images and predicting diseases using a TensorFlow model.
- Bottom Status Bar:** Shows the current file is `image_service.py — APTOS_backend`, along with other status indicators like Jira, Bitbucket, and GitHub links.

```
image_service.py -- APTOS_backend
app.py -- APTOS_backend

You, 2 days ago | 1 author (You)
1
2 import cv2
3 import tensorflow as tf
4 import numpy as np
5
6 def preprocess_image(image):
7     image = cv2.resize(image, (48, 48))
8     img_array = image.reshape(1, 48, 48, 1)
9     img_array = img_array / 255.0 # Normalize pixel values (assuming values range from 0 to 255)
10
11     return img_array
12
13 def predict_disease(image):
14     pre_processed_image = preprocess_image(image)
15
16     # Load the h5 model file
17     model = tf.keras.models.load_model('models/saved_model.hdf5')
18
19     # Perform prediction using the preprocessed image
20     predictions = model.predict(pre_processed_image)
21
22     # Extract the predicted class
23     predicted_class_index = np.argmax(predictions)
24
25     return int(predicted_class_index) You, 2 days ago + done
```

Figure 4: Back-end Service code

```

# model
model = keras.Sequential()

model.add(tf.keras.layers.Conv2D(64, (3,3) , input_shape = (48,48,1) , padding="same"))
model.add(tf.keras.layers.MaxPooling2D((2,2)))

model.add(tf.keras.layers.Conv2D(64, (3,3) , padding="same"))
model.add(tf.keras.layers.MaxPooling2D((2,2)))

model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(5 , activation = 'softmax'))

model.summary()

```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 5)	46085

Total params: 83,653
Trainable params: 83,653
Non-trainable params: 0

Figure 5: Custom made lightweight model

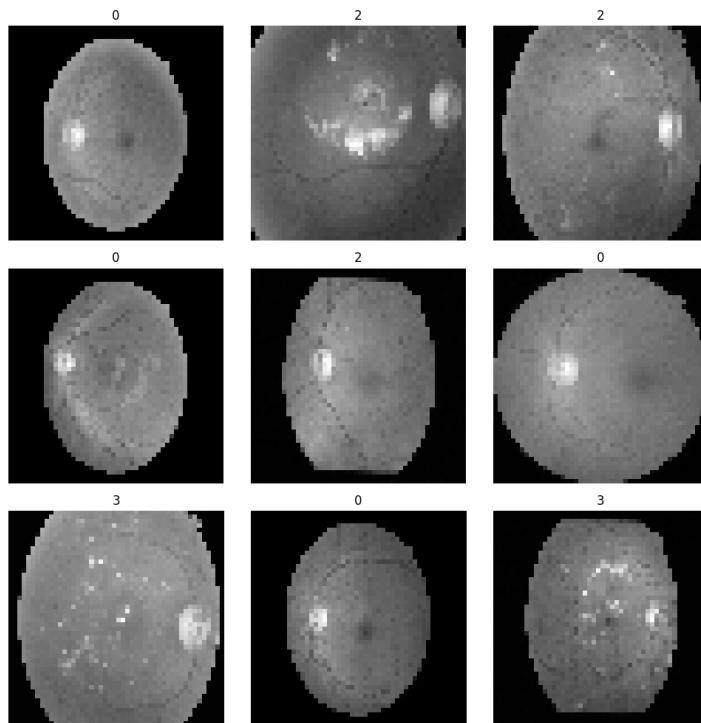
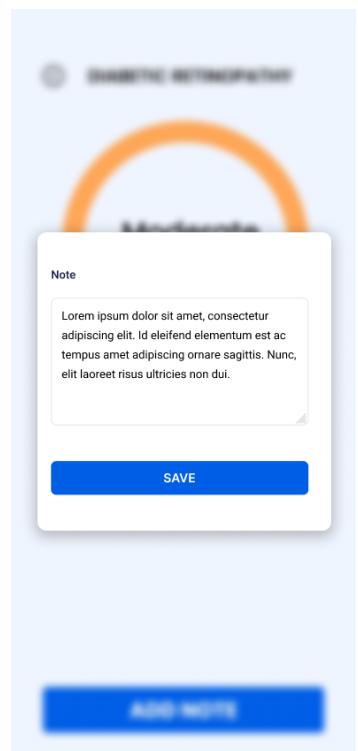
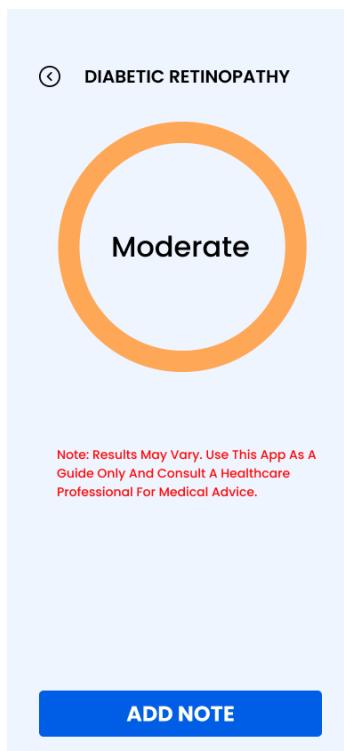
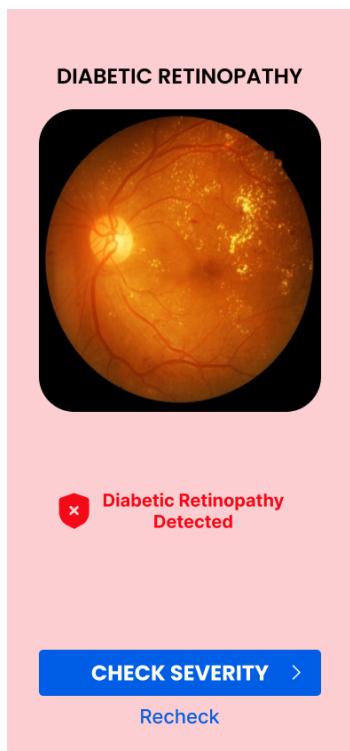


Figure 6: Preprocessed images

1.3. Mobile App UI designs



2. Project view

The screenshot shows the Microsoft Planner - Board view for a project titled "23-162_DETECT_AND_ANNOTATE...". The board is organized into four columns representing team members: Nilaksha, Praveen, Rasanga, and Chamod. Each column contains a list of tasks with their due dates and status indicators (green, red, or blue). A sidebar on the left provides navigation links for Activity, Chat, Teams, Assignments, Calendar, Calls, Files, and Apps.

Team Member	Task	Due Date	Status
Nilaksha	Testing the Application	07/23	Green
	Integrate the Trained Model	07/22	Green
	Implement Functionalities of Application using React Native	07/14	Green
	Implement UI using React Native	07/10	Green
	Design Mobile Application Interfaces	07/09	Green
	Train the Model	07/17	Green
	Design and Implement the Model	07/17	Green
	Testing the Application	07/23	Red
	Integrate the Trained Model	07/22	Red
	Implement Functionalities of Application using React Native	07/14	Blue
Praveen	Implement UI using React Native	07/10	Red
	Design Mobile Application Interfaces	07/10	Green
	Train the Model	07/17	Green
	Testing the Application	07/23	Red
	Integrate the Trained Model	07/22	Red
	Implement Functionalities of Application using React Native	07/14	Blue
	Implement UI using React Native	07/10	Red
	Design Mobile Application Interfaces	07/10	Green
	Train the Model	07/17	Green
	Testing the Application	07/23	Red
Rasanga	Integrate the Trained Model	07/23	Green
	Implement Functionalities of Application using React Native	07/14	Green
	Implement UI using React Native	07/10	Red
	Design Mobile Application Interfaces	07/10	Green
	Train the Model	07/17	Green
	Testing the Application	07/23	Red
	Integrate the Trained Model	07/22	Green
	Implement Functionalities of Application using React Native	07/14	Blue
	Implement UI using React Native	07/10	Red
	Design Mobile Application Interfaces	07/10	Green
Chamod	Implement Functionalities of Application using React Native	07/14	Blue
	Implement UI using React Native	07/12	Red
	Design Mobile Application Interfaces	07/12	Green
	Train the Model	07/10	Green
	Testing the Application	07/23	Red
	Integrate the Trained Model	07/22	Red
	Implement Functionalities of Application using React Native	07/14	Blue
	Implement UI using React Native	07/12	Red
	Design Mobile Application Interfaces	07/12	Green
	Train the Model	07/10	Green

Figure 7: Planner - board view

The screenshot shows the Microsoft Planner - Schedule view for the same project. The view displays a monthly calendar from July 2023, showing tasks assigned to team members on specific dates. Tasks are color-coded by assignee and include descriptions like "Design Mobile Application Interfaces" and "Testing the Application". A sidebar on the left provides navigation links for Activity, Chat, Teams, Assignments, Calendar, Calls, Files, and Apps.

Date	Task	Assignee
July 1	Implement Functionalities of Application using React Native	Nilaksha
July 1	Implement UI using React Native	Praveen
July 1	Integrate the Trained Model	Rasanga
July 1	Implement Functionalities of Application using React Native	Chamod
July 2	Design Mobile Application Interfaces	Nilaksha
July 2	Design Mobile Application Interfaces	Praveen
July 2	Design Mobile Application Interfaces	Rasanga
July 2	Design Mobile Application Interfaces	Chamod
July 3	Design Mobile Application Interfaces	Nilaksha
July 3	Design Mobile Application Interfaces	Praveen
July 3	Design Mobile Application Interfaces	Rasanga
July 3	Design Mobile Application Interfaces	Chamod
July 4	Design Mobile Application Interfaces	Nilaksha
July 4	Design Mobile Application Interfaces	Praveen
July 4	Design Mobile Application Interfaces	Rasanga
July 4	Design Mobile Application Interfaces	Chamod
July 5	Design Mobile Application Interfaces	Nilaksha
July 5	Design Mobile Application Interfaces	Praveen
July 5	Design Mobile Application Interfaces	Rasanga
July 5	Design Mobile Application Interfaces	Chamod
July 6	Design Mobile Application Interfaces	Nilaksha
July 6	Design Mobile Application Interfaces	Praveen
July 6	Design Mobile Application Interfaces	Rasanga
July 6	Design Mobile Application Interfaces	Chamod
July 7	Design Mobile Application Interfaces	Nilaksha
July 7	Design Mobile Application Interfaces	Praveen
July 7	Design Mobile Application Interfaces	Rasanga
July 7	Design Mobile Application Interfaces	Chamod
July 8	Design Mobile Application Interfaces	Nilaksha
July 8	Design Mobile Application Interfaces	Praveen
July 8	Design Mobile Application Interfaces	Rasanga
July 8	Design Mobile Application Interfaces	Chamod
July 9	Design Mobile Application Interfaces	Nilaksha
July 9	Design Mobile Application Interfaces	Praveen
July 9	Design Mobile Application Interfaces	Rasanga
July 9	Design Mobile Application Interfaces	Chamod
July 10	Design Mobile Application Interfaces	Nilaksha
July 10	Design Mobile Application Interfaces	Praveen
July 10	Design Mobile Application Interfaces	Rasanga
July 10	Design Mobile Application Interfaces	Chamod
July 11	Design Mobile Application Interfaces	Nilaksha
July 11	Design Mobile Application Interfaces	Praveen
July 11	Design Mobile Application Interfaces	Rasanga
July 11	Design Mobile Application Interfaces	Chamod
July 12	Implement Functionalities of Application using React Native	Nilaksha
July 12	Implement UI using React Native	Praveen
July 12	Integrate the Trained Model	Rasanga
July 12	Implement Functionalities of Application using React Native	Chamod
July 13	Implement Functionalities of Application using React Native	Nilaksha
July 13	Implement UI using React Native	Praveen
July 13	Integrate the Trained Model	Rasanga
July 13	Implement Functionalities of Application using React Native	Chamod
July 14	Implement Functionalities of Application using React Native	Nilaksha
July 14	Implement UI using React Native	Praveen
July 14	Integrate the Trained Model	Rasanga
July 14	Implement Functionalities of Application using React Native	Chamod
July 15	Implement Functionalities of Application using React Native	Nilaksha
July 15	Implement UI using React Native	Praveen
July 15	Integrate the Trained Model	Rasanga
July 15	Implement Functionalities of Application using React Native	Chamod
July 16	Design and Implement the Model	Nilaksha
July 16	Train the Model	Praveen
July 16	Testing the Application	Rasanga
July 16	Testing the Application	Chamod
July 17	Design and Implement the Model	Nilaksha
July 17	Train the Model	Praveen
July 17	Testing the Application	Rasanga
July 17	Testing the Application	Chamod
July 18	Design and Implement the Model	Nilaksha
July 18	Train the Model	Praveen
July 18	Testing the Application	Rasanga
July 18	Testing the Application	Chamod
July 19	Design and Implement the Model	Nilaksha
July 19	Train the Model	Praveen
July 19	Testing the Application	Rasanga
July 19	Testing the Application	Chamod
July 20	Design and Implement the Model	Nilaksha
July 20	Train the Model	Praveen
July 20	Testing the Application	Rasanga
July 20	Testing the Application	Chamod
July 21	Design and Implement the Model	Nilaksha
July 21	Train the Model	Praveen
July 21	Testing the Application	Rasanga
July 21	Testing the Application	Chamod
July 22	Design and Implement the Model	Nilaksha
July 22	Train the Model	Praveen
July 22	Testing the Application	Rasanga
July 22	Testing the Application	Chamod
July 23	Testing the Application	Nilaksha
July 23	Testing the Application	Praveen
July 23	Testing the Application	Rasanga
July 23	Testing the Application	Chamod
July 24	Testing the Application	Nilaksha
July 24	Testing the Application	Praveen
July 24	Testing the Application	Rasanga
July 24	Testing the Application	Chamod
July 25	Testing the Application	Nilaksha
July 25	Testing the Application	Praveen
July 25	Testing the Application	Rasanga
July 25	Testing the Application	Chamod
July 26	Testing the Application	Nilaksha
July 26	Testing the Application	Praveen
July 26	Testing the Application	Rasanga
July 26	Testing the Application	Chamod
July 27	Testing the Application	Nilaksha
July 27	Testing the Application	Praveen
July 27	Testing the Application	Rasanga
July 27	Testing the Application	Chamod
July 28	Testing the Application	Nilaksha
July 28	Testing the Application	Praveen
July 28	Testing the Application	Rasanga
July 28	Testing the Application	Chamod
July 29	Testing the Application	Nilaksha
July 29	Testing the Application	Praveen
July 29	Testing the Application	Rasanga
July 29	Testing the Application	Chamod
July 30	Testing the Application	Nilaksha
July 30	Testing the Application	Praveen
July 30	Testing the Application	Rasanga
July 30	Testing the Application	Chamod
July 31	Testing the Application	Nilaksha
July 31	Testing the Application	Praveen
July 31	Testing the Application	Rasanga
July 31	Testing the Application	Chamod

Figure 9: Planner - schedule view

3. Gantt chart

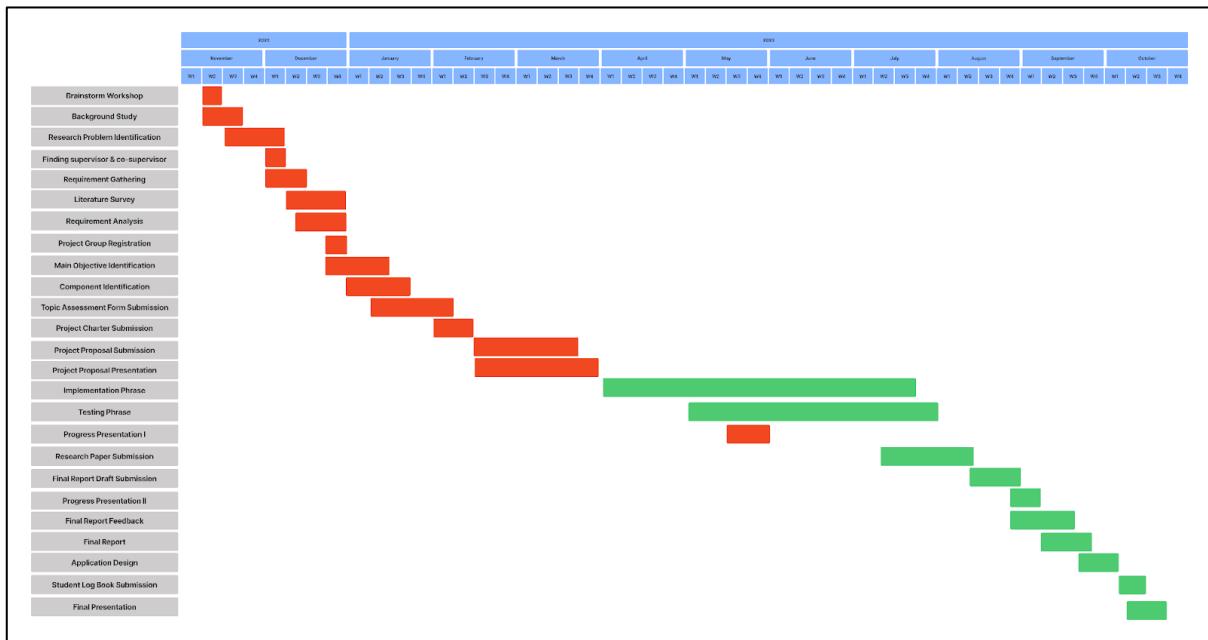
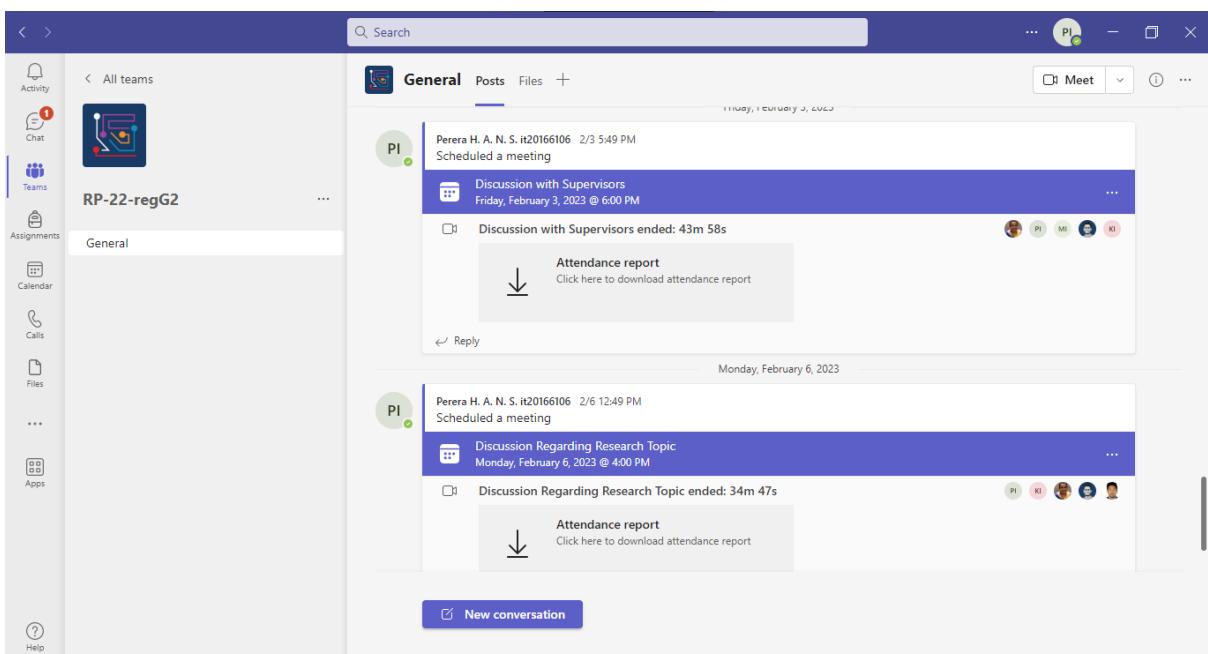


Figure 10: Gantt chart

4. Screenshots of conversations and calls – MS Teams



Calendar

Today < > July 2023 Work week

	03 Monday	04 Tuesday	05 Wednesday	06 Thursday	07 Friday
6 PM					
7 PM	Daily Progress Meeting Perera H. A. N. S. it20166106	Daily Progress Meeting Perera H. A. N. S. it20166106	Daily Progress Meeting Perera H. A. N. S. it20166106	Daily Progress Meeting Perera H. A. N. S. it20166106	Daily Progress Meeting Perera H. A. N. S. it20166106
8 PM					
9 PM		Meeting with supervisor Microsoft Teams Meeting			
10 PM					
11 PM					

