

**Machine Learning Approach to Detect & Annotate Eye Diseases using
Retinal Images**

2023-162

Status Document

Perera H. A. N. S.

IT20166106

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

Department of Information Technology

**Sri Lanka Institute of Information Technology
Sri Lanka**

May 2023

Table of Contents

1.	Project progress.....	3
1.1	Dataset	3
1.2	Model Implementation of Diabetic Retinopathy Detection.....	4
2.	Project View.....	6
3.	Gantt chart.....	8
4.	Screenshots of Conversations and Calls - Microsoft Teams.....	9

1. Project progress

1.1 Dataset

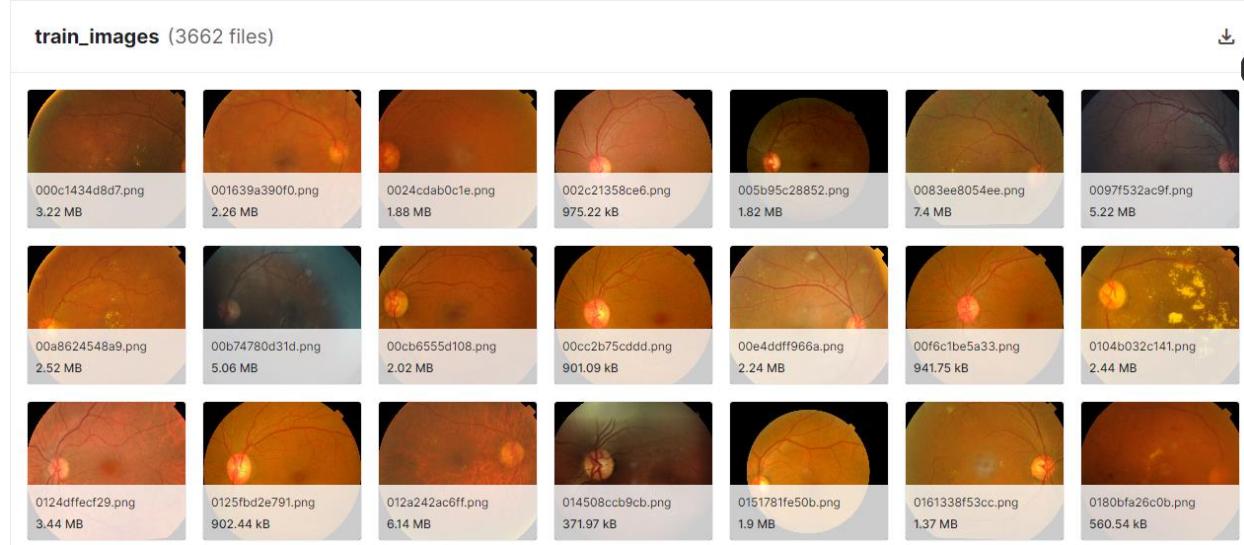


Figure 1 - APTOS 2019 Blindness Detection Dataset

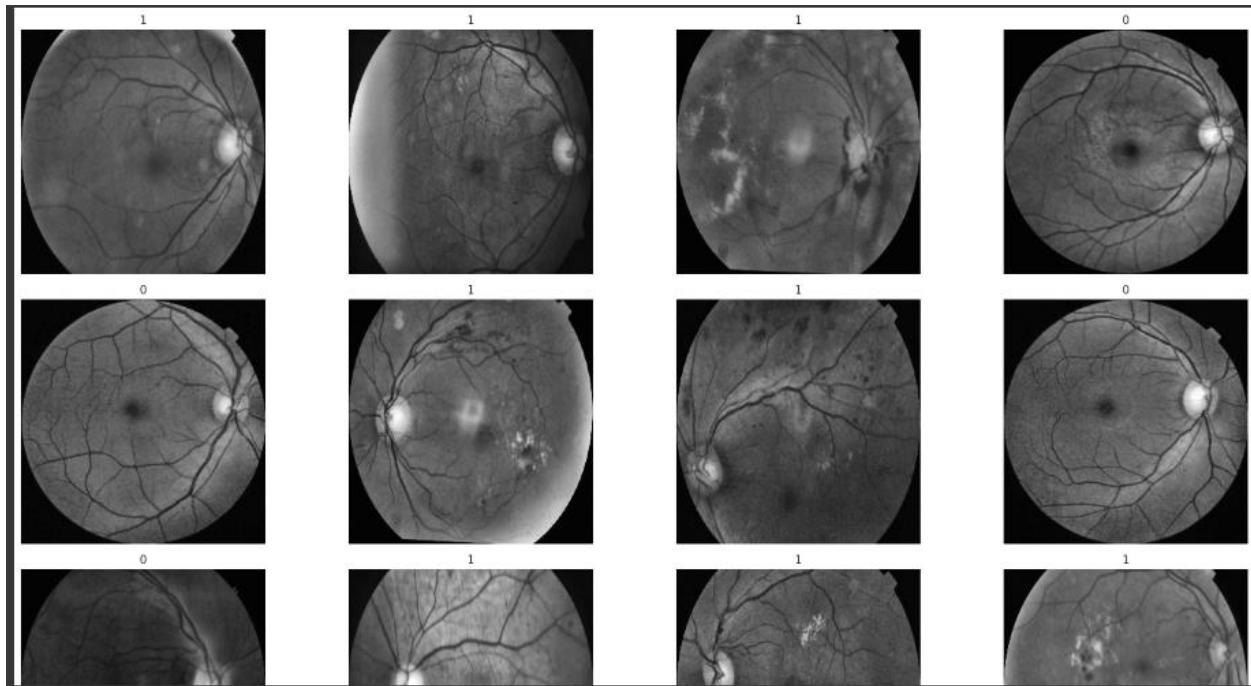


Figure 2- Preprocessed Retinal Fundus Images

1.2 Model Implementation of Diabetic Retinopathy Detection

The algorithm uses TensorFlow and Keras frameworks with the EfficientNetB3 deep learning model architecture for image analysis. EfficientNetB3, a pre-trained Convolutional Neural Network, efficiently extracts high-level features from images. The system enhances the model's capacity by adding extra layers, including Batchnormalization for normalization, a bottleneck layer for feature extraction, and regularization techniques to prevent overfitting. A Dropout layer is used to further combat overfitting. The output layer consists of a Dense layer with a softmax activation function for detecting Diabetic Retinopathy. The use of EfficientNetB3 makes the model suitable for mobile applications due to its performance and computational efficiency.

```
batch_size = 32
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)

ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1) if ts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size

def scalar(img):
    return img

tr_gen = ImageDataGenerator(preprocessing_function= scalar)
ts_gen = ImageDataGenerator(preprocessing_function= scalar)

train_gen = tr_gen.flow_from_dataframe( train_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                         color_mode= 'rgb', shuffle= True, batch_size= batch_size)

valid_gen = ts_gen.flow_from_dataframe( valid_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                         color_mode= 'rgb', shuffle= True, batch_size= batch_size)

test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                         color_mode= 'rgb', shuffle= False, batch_size= test_batch_size)
```

Figure 3 - Model Implementation

```
[ ] class MyCallback(keras.callbacks.Callback):
    def __init__(self, model, patience, stop_patience, threshold, factor, batches, epochs, ask_epoch):
        super(MyCallback, self).__init__()
        self.model = model
        self.patience = patience
        self.stop_patience = stop_patience
        self.threshold = threshold
        self.factor = factor
        self.batches = batches
        self.epochs = epochs
        self.ask_epoch = ask_epoch
        self.ask_epoch_initial = ask_epoch

        self.count = 0
        self.stop_count = 0
        self.best_epoch = 1
        self.initial_lr = float(tf.keras.backend.get_value(model.optimizer.lr))
        self.highest_tracc = 0.0
        self.lowest_vloss = np.inf
        self.best_weights = self.model.get_weights()
        self.initial_weights = self.model.get_weights()

    def on_train_begin(self, logs= None):
        msg = 'Do you want model asks you to halt the training [y/n] ?'
        print(msg)
```

Figure 4 - Model Implementation

```
[ ] img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_gen.class_indices.keys()))

base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top= False, weights= "imagenet", input_shape= img_shape, pooling= 'max')

model = Sequential([
    base_model,
    BatchNormalization(axis= -1, momentum= 0.99, epsilon= 0.001),
    Dense(256, kernel_regularizer= regularizers.l2(l= 0.016), activity_regularizer= regularizers.l1(0.006),
          bias_regularizer= regularizers.l1(0.006), activation= 'relu'),
    Dropout(rate= 0.45, seed= 123),
    Dense(class_count, activation= 'softmax')
])

model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy', metrics= ['accuracy'])

model.summary()
```

Figure 5 - Model Implementation

2. Project View

The screenshot shows the Microsoft Planner interface in 'List' view. On the left, there's a sidebar with icons for Activity, Chat, Teams, Calendar, Calls, Files, Apps, and Help. The main area has a header with 'General' selected, followed by 'Posts', 'Files', 'Planner', and a '+' button. Below the header are buttons for 'Group by Bucket', 'Filter', 'List', 'Board', 'Charts', and 'Schedule'. The main content area is divided into three columns: 'Nilaksha', 'Praveen', and 'Rasanga'. Each column has a 'Add task' button. Under each name, there's a list of tasks with due dates and initials (P, KI) next to them. The tasks are:

Column	Task	Due Date	Initials
Nilaksha	Testing the Application	07/23	P
	Integrate the Trained Model	07/22	P
	Implement Functionalities of Application using React Native	07/14	P
	Implement UI using React Native	07/10	P
	Design Mobile Application Interfaces		
Praveen	Testing the Application	07/23	KI
	Integrate the Trained Model	07/22	KI
	Implement Functionalities of Application using React Native		
	Implement UI using React Native	07/14	KI
	Design Mobile Application Interfaces		
Rasanga	Testing the Application	07/23	
	Integrate the Trained Model	07/22	
	Implement Functionalities of Application using React Native	07/14	
	Implement UI using React Native	07/10	
	Design Mobile Application Interfaces		

Figure 6 – Planner - Task List View

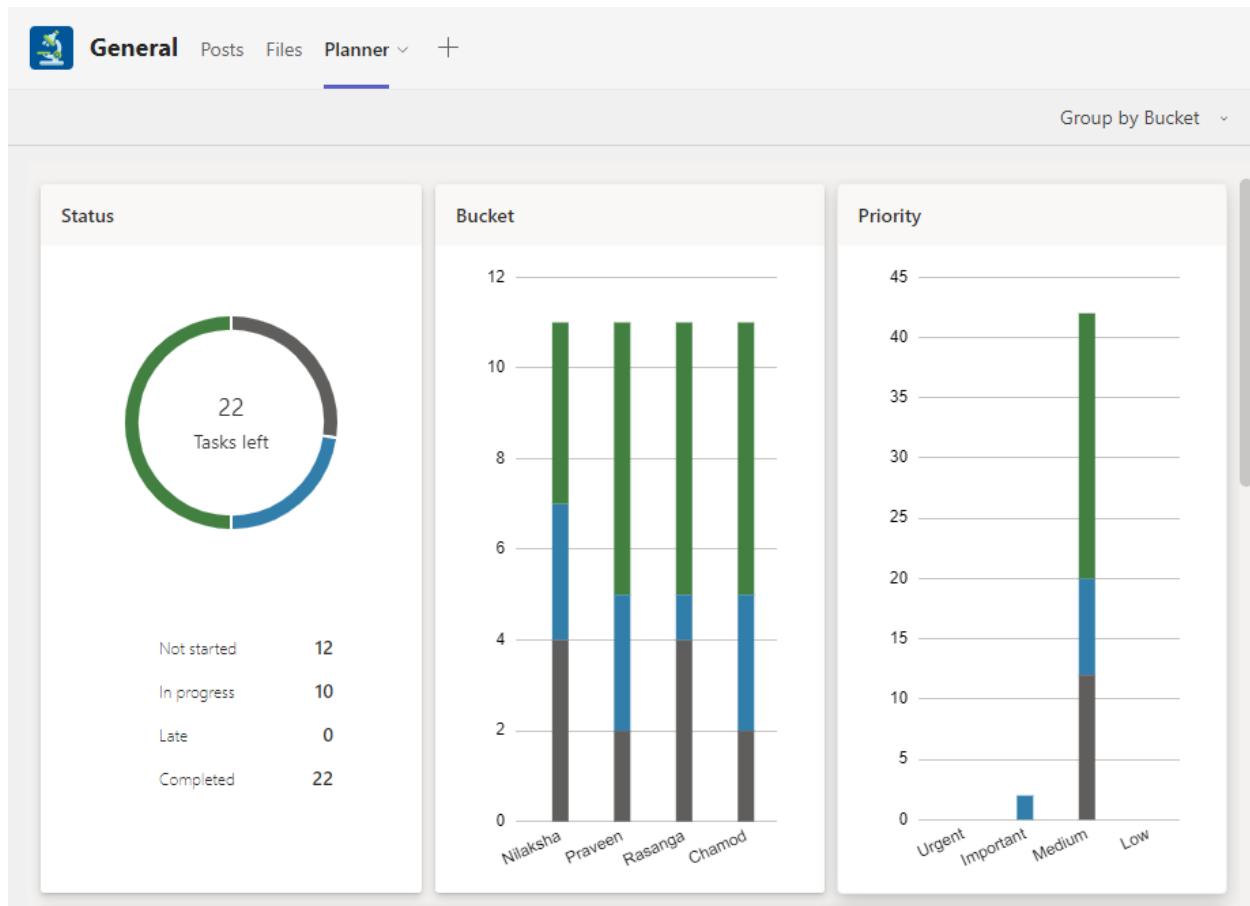


Figure 7 - Planner - Chart View

The screenshot shows the Microsoft Planner application interface. On the left, there's a vertical sidebar with icons for Activity, Chat, Teams, Assignments, Calendar, Calls, Files, Apps, and Help. The main area is titled "General" and shows a "Planner" tab selected. The view is set to "Month" mode for July 2023. The calendar grid displays various tasks:

- Week of July 9:** Tasks include "Design ...", "Implement ...", and "Implement ...".
- Week of July 16:** Tasks include "Design a...", "Train the ...", and "Integrate th...".
- Week of July 23:** Task is "Testing the ...".

To the right of the calendar, there's a sidebar titled "Unscheduled tasks" with a button to "Add task". Below the calendar, names are listed next to their respective tasks:

- Nilaksha (near July 14)
- Praveen (near July 17)
- Rasanga (near July 18)
- Chamod (near July 22)
- Completed (near July 29)

Figure 8 - Planner - Schedule View

3. Gantt chart

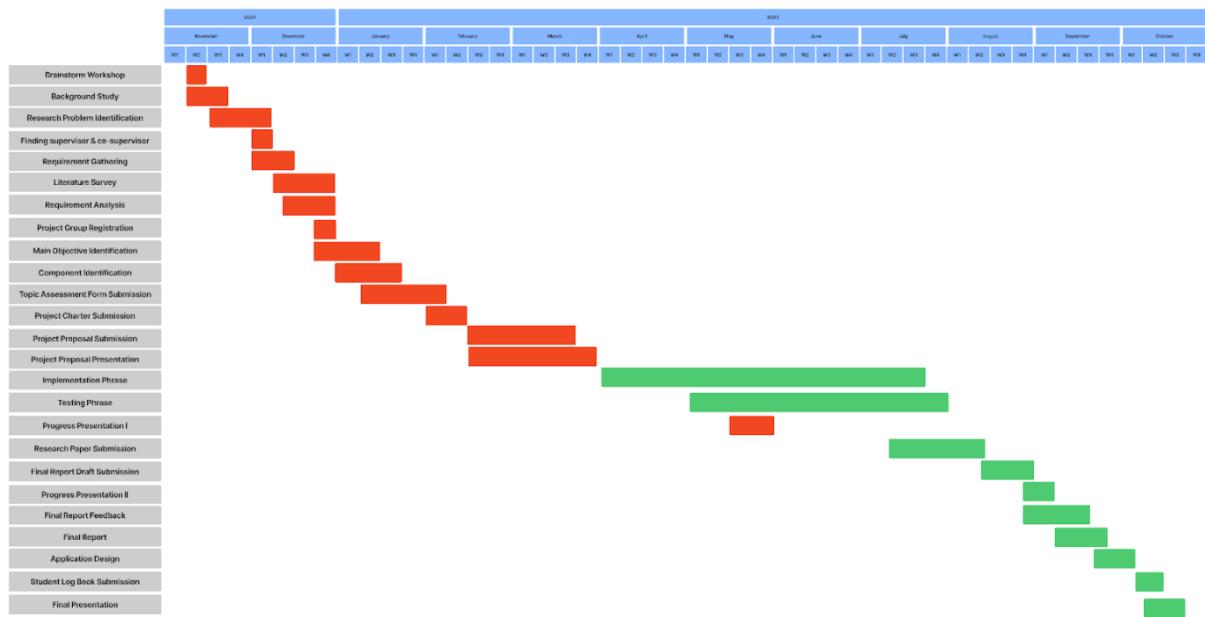


Figure 9 - Gantt Chart

4. Screenshots of Conversations and Calls - Microsoft Teams

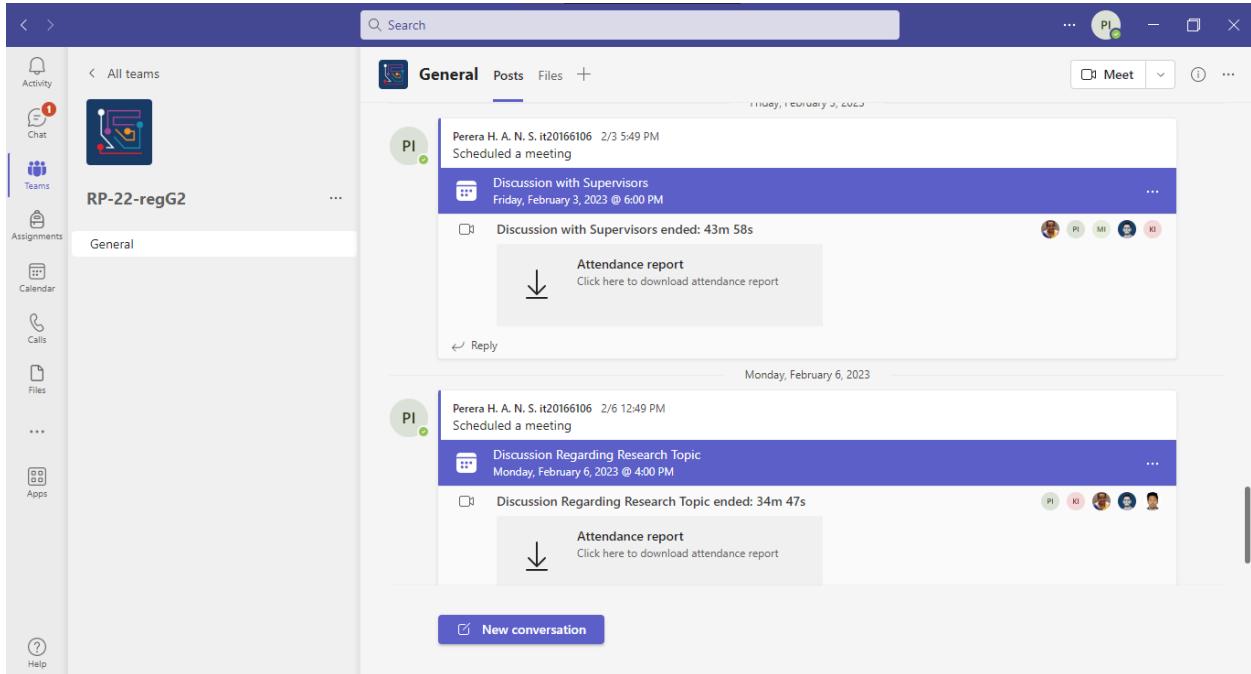


Figure 10 - MS Teams Channel

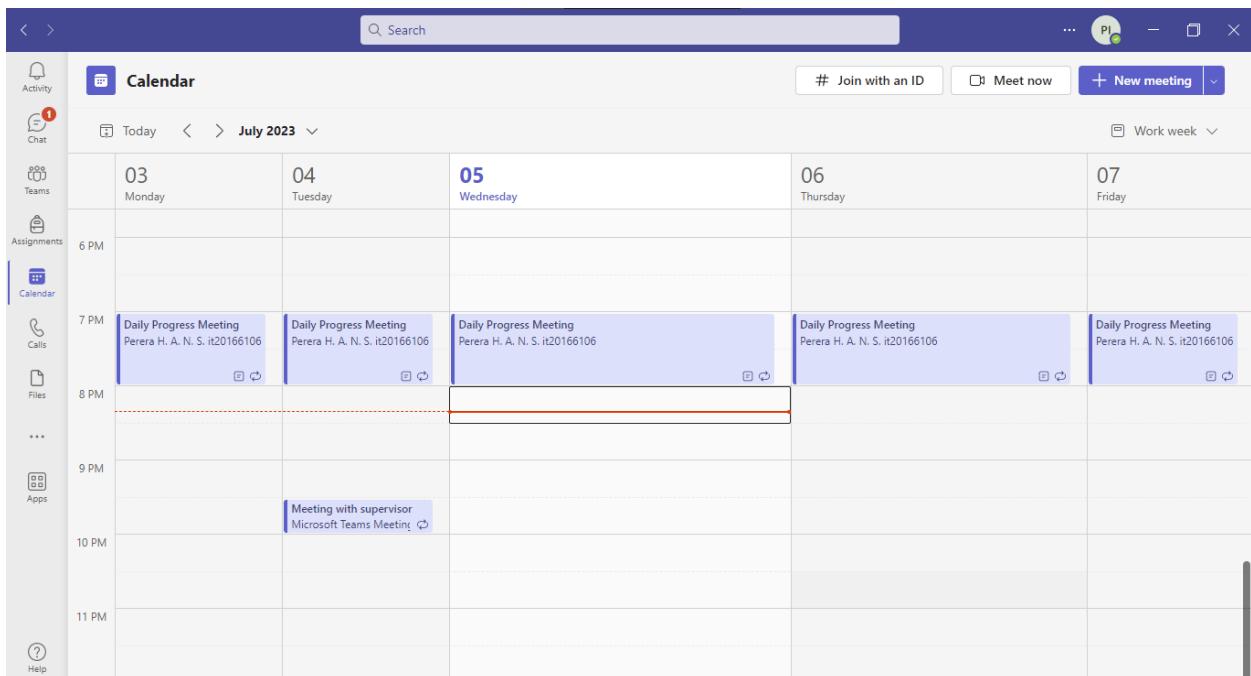


Figure 11 - Scheduled Meetings

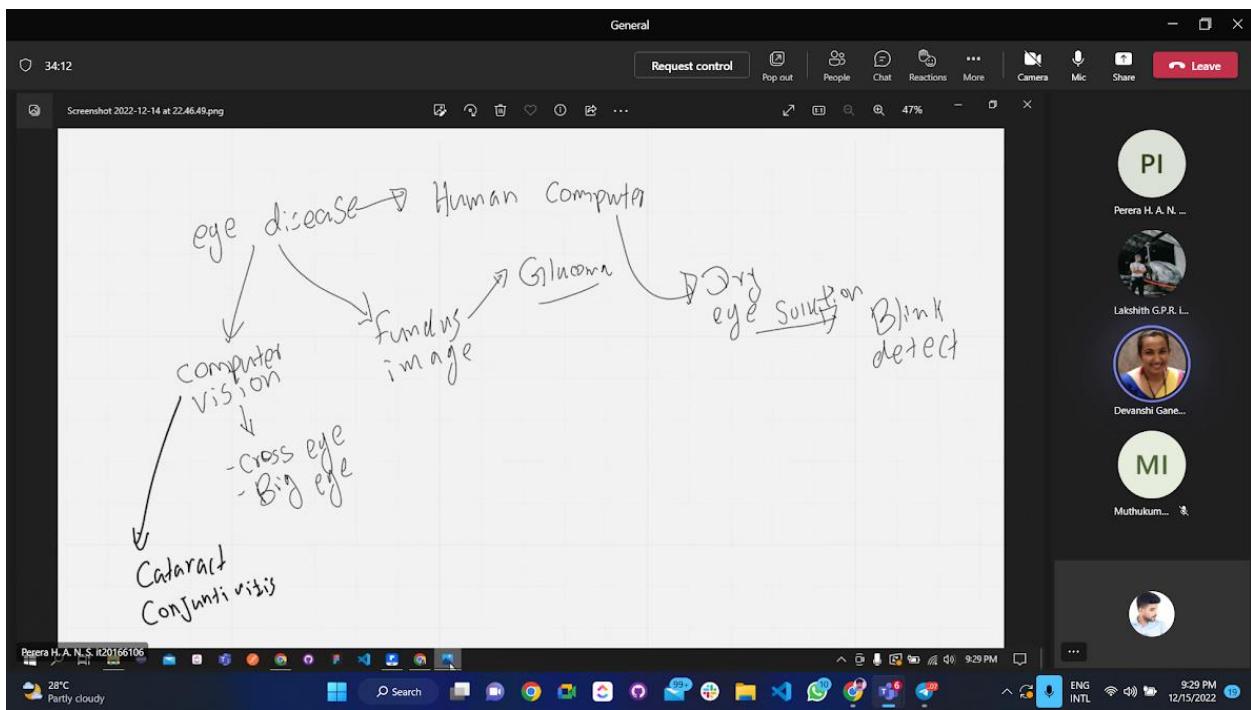


Figure 12 -Meetings with Supervisors