Progress Report
Product-based Capstone Project

- **Please make a copy of this document (do not ask for edit permissions)**
- Discuss with your team prior to filling up this doc so that the content of this progress is the same across all team members.
- Replace the highlighted part of this proposal with your team's answers.
- See the limits of each answer below. You may exceed one page if necessary. Maximum 2 pages.
- Submit this document individually to this form

Project Name : Online Ambulance Booking Application to Improve Emergency Medical Response
Team ID : CH2-PS284
Team Member :

1. (ML) M227BSY1252 – Anggito Karta Wijaya – Universitas Jember - [Active]
2. (ML) M227BSY0317 – Dewanata Hammada – Universitas Jember - [Active]
3. (ML) M429BSX0777 – Sasa Kirana Wulandari – Universitas Dinamika Bangsa [Active]
4. (CC) C134BSX3657 – Naila Ardelia – Politeknik Negeri Sriwijaya - [Active]
5. (CC) C134BSX3720 – Clara Oktariani – Politeknik Negeri Sriwijaya - [Active]
6. (MD) A227BSY2735 – Hans Havilah Fadhlan – Universitas Jember - [Active]
7. (MD) A491BSX2165 – Chintia Agustin – Universitas Muhammadiyah Bandung - [Active]

Progress Report

Product-based Capstone Project

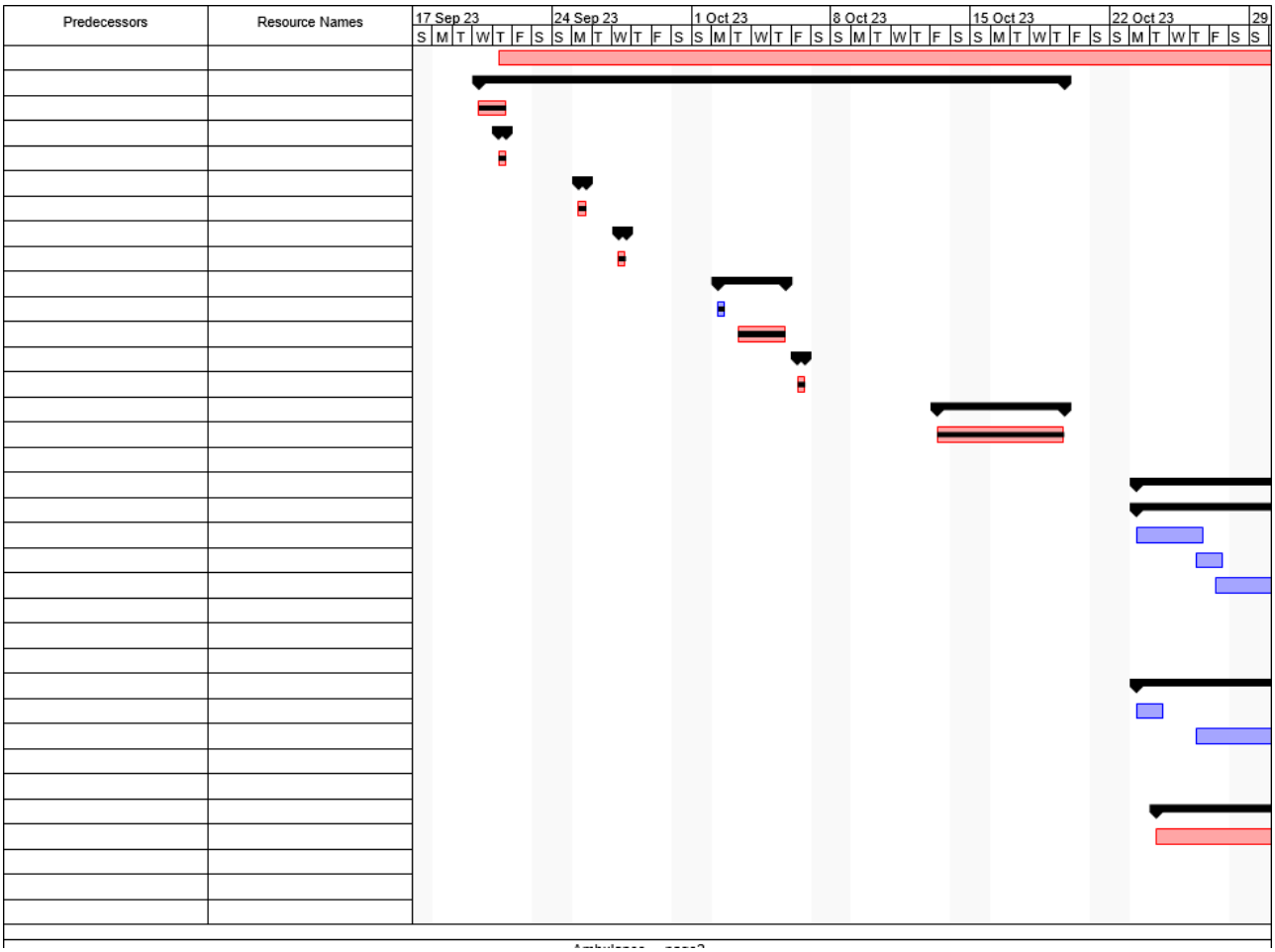
Project Schedule (based on project plan):

		Name	Duration	Start	Finish
1		AMBULANCE	66 days?	9/21/23 8:00 AM	12/21/23 5:00 PM
2	✔	**Preparing Sprint**	**22 days?**	**9/20/23 8:00 AM**	**10/19/23 5:00 PM**
3	✔	Reqruitment team and valid team	2 days?	9/20/23 8:00 AM	9/21/23 5:00 PM
4	✔	**The First Meeting our Team**	**1 day?**	**9/21/23 8:00 AM**	**9/21/23 5:00 PM**
5	✔	Chit-Chat and Gathering	1 day?	9/21/23 8:00 AM	9/21/23 5:00 PM
6	✔	**The Second Meeting out Team**	**1 day?**	**9/25/23 8:00 AM**	**9/25/23 5:00 PM**
7	✔	Discussing project ideas from our group	1 day?	9/25/23 8:00 AM	9/25/23 5:00 PM
8	✔	**The Third Meeting our Team**	**1 day?**	**9/27/23 8:00 AM**	**9/27/23 5:00 PM**
9	✔	Discussing creating Trello, LinkTree and UI/UX	1 day?	9/27/23 8:00 AM	9/27/23 5:00 PM
10	✔	**The Fourth Meeting our Team**	**4 days?**	**10/2/23 8:00 AM**	**10/5/23 5:00 PM**
11	✔	Discussing the Fix idea from our group	1 day?	10/2/23 8:00 AM	10/2/23 5:00 PM
12	✔	Creating trello, LinkTree, Drive and UI/UX from our group's F..	3 days?	10/3/23 8:00 AM	10/5/23 5:00 PM
13	✔	**The Fifth Meeting our Team**	**1 day?**	**10/6/23 8:00 AM**	**10/6/23 5:00 PM**
14	✔	Discussed our group's project plan	1 day?	10/6/23 8:00 AM	10/6/23 5:00 PM
15	✔	**The Sixth Meeting our Team**	**5 days?**	**10/13/23 8:00 AM**	**10/19/23 5:00 PM**
16	✔	Working on the Project Plan	5 days?	10/13/23 8:00 AM	10/19/23 5:00 PM
17		**Sprint 1**	**23 days?**	**10/23/23 8:00 AM**	**11/22/23 5:00 PM**
18		**Machine Learning**	**10 days?**	**10/23/23 8:00 AM**	**11/3/23 5:00 PM**
19		Collecting and Researching the dataset	4 days?	10/23/23 8:00 AM	10/26/23 5:00 PM
20		Preprocessing the datasets	2 days?	10/26/23 8:00 AM	10/27/23 5:00 PM
21		Making Model Architecture	2 days?	10/27/23 8:00 AM	10/30/23 5:00 PM
22		Training and Testing Dataset	2 days?	10/30/23 8:00 AM	10/31/23 5:00 PM
23		Deploy to Built Postman	3 days?	11/1/23 8:00 AM	11/3/23 5:00 PM
24		Finishing and Check Revision	1 day?	11/3/23 8:00 AM	11/3/23 5:00 PM
25		**Cloud Computing**	**23 days?**	**10/23/23 8:00 AM**	**11/22/23 5:00 PM**
26		Create design microservices for application and API	2 days?	10/23/23 8:00 AM	10/24/23 5:00 PM
27		Configuring Server Firebase(Google Cloud)	5 days?	10/26/23 8:00 AM	11/1/23 5:00 PM
28		Create Microservices and API	11 days?	11/1/23 8:00 AM	11/15/23 5:00 PM
29		Testing API with Machine Learning Model	5 days?	11/15/23 8:00 AM	11/21/23 5:00 PM
30		**Mobile Development**	**22 days?**	**10/24/23 8:00 AM**	**11/22/23 5:00 PM**
31		Designing Mobile Application	22 days?	10/24/23 8:00 AM	11/22/23 5:00 PM
32		**Sprint 2**	**21 days?**	**11/22/23 8:00 AM**	**12/20/23 5:00 PM**
33		**Cloud Computing**	**13 days?**	**11/22/23 8:00 AM**	**12/8/23 5:00 PM**

		Name	Duration	Start	Finish
34		Testing and Tracking Geo Location API	13 days?	11/22/23 8:00 AM	12/8/23 5:00 PM
35		**Mobile Development**	**21 days?**	**11/22/23 8:00 AM**	**12/20/23 5:00 PM**
36		Integration with API	19 days?	11/22/23 8:00 AM	12/18/23 5:00 PM
37		Testing	3 days?	12/18/23 8:00 AM	12/20/23 5:00 PM

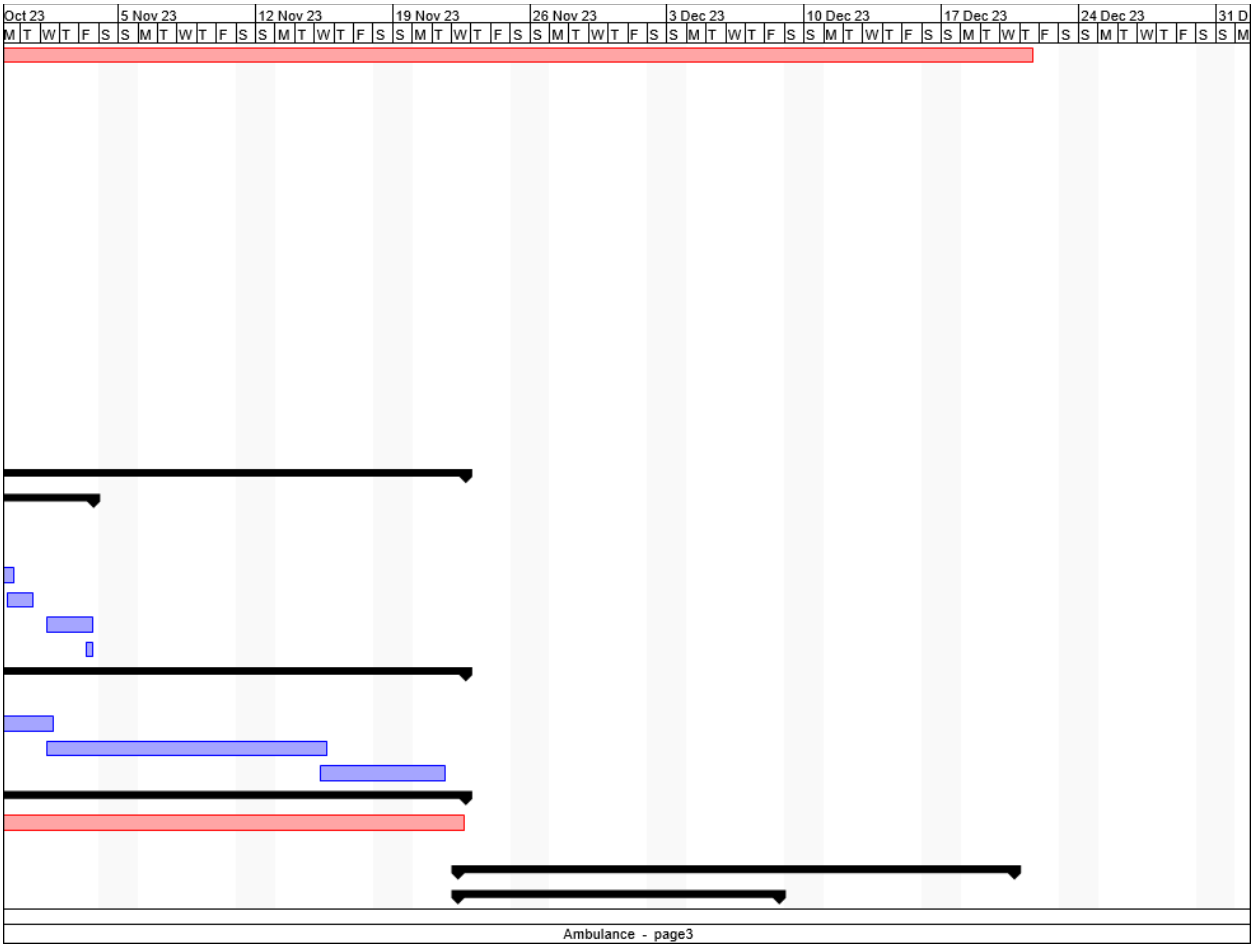
Progress Report

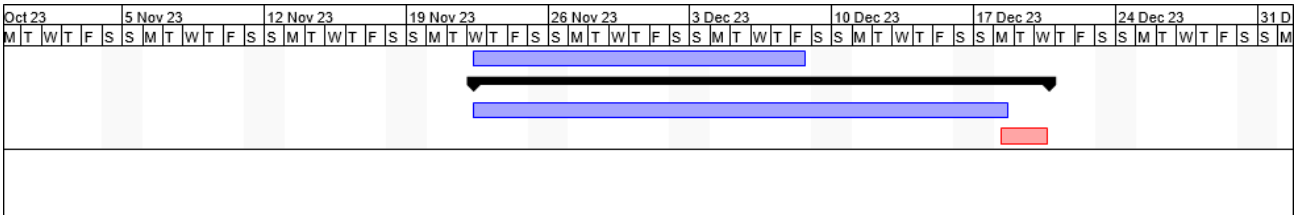
Product-based Capstone Project



Progress Report

Product-based Capstone Project





Progress Report
Product-based Capstone Project

Project Progress Description:

Machine Learning
Completed

- Developing Face Feature Extraction Model with Facenet.
- Creating Augmentation Generator with ImageDataGenerator.
- Implementing Face Detection with dlib.
- Building Model for Face Verification with SKlearn Isolation Forest.
- Successful Creation of Verification Model with Augmented Training Data.
- Creating API for Face Verification Model with Flask.
- Implementing Authentication for API.
- Applying MySQL as a Database in Face Verification API.
- Successful Deployment of Face Verification API to Google Cloud.

On Progress

- Create a model registration flask
- Integrating with Sql gcloud with flask API
- Integrating gcloud storage with flask API

Not Started yet

- Perform stress test on fire flask
- Create API documentation

Cloud Computing
Completed

- Deploy the apps - We have developed a robust API to facilitate and optimize inference from ML models for better performance, as well as introduced version control so that models can be managed easily.

On Progress

- Create database on the cloud

Not Started yet

- Geolocation service - we are currently refining this feature, implementing a geolocation model to track ambulances that will be used in the app.

Mobile Development
Completed

- UI and Prototipe

On Progress

- Shift to Application Logic
- Machine Learning Integration

Progress Report
Product-based Capstone Project

Not Started yet

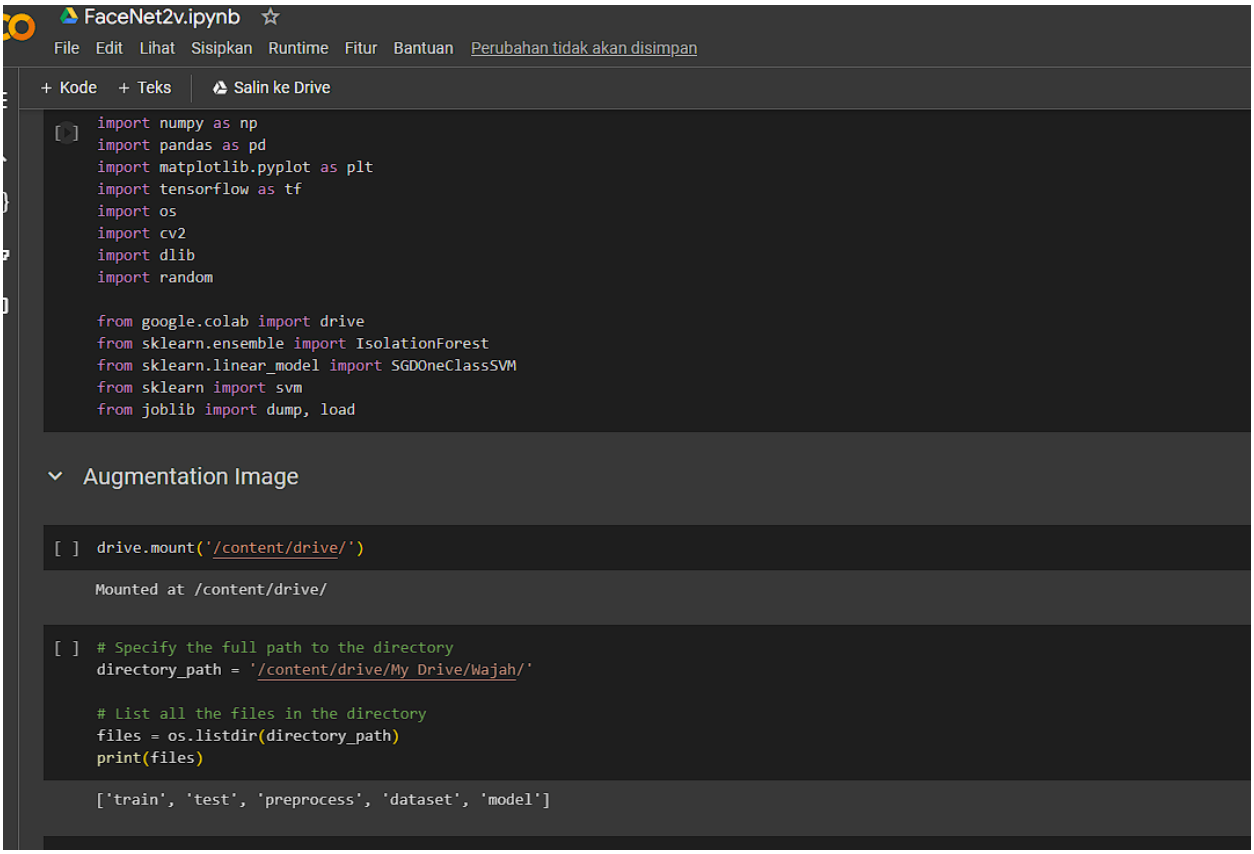
- Cloud API Integration
- Testing

If the complete capstone is 100%, tell us how many percent of your group's project has been completed. Please describe the reason for your percentage.
At the time of reporting, project progress should have reached at least 50% of the total project scope

Our capstone project made 75% progress, focusing on face model development with Facenet, augmentation generator generation, and face detection with dlib. face verification model using SKlearn Isolation Forest and API was built with Flask, integrated with MySQL, and successfully implemented on Google Cloud. We developed application with a robust API, supporting inference from ML models and version control. Work is underway model registration, integration of Google Cloud SQL and gcloud storage with the Flask API, and moving into application logic and machine learning integration. Not yet completed Flask robustness testing, API documentation, geolocation service enhancements, cloud API integration, and testing.

Progress Report
Product-based Capstone Project

Please attach your supporting evidence.
Machine Learning



bangkit

Google goto traveloka

Kampus Merdeka INDONESIA JAYA

Progress Report

Product-based Capstone Project

```
[ ]  def convert_and_trim_bb(image, rect):
       startX = 0
       startY = 0
       w = 0
       h = 0
       if len(rect) > 0:
         rect = rect[0]
         # extract the starting and ending (x, y)-coordinates of the
         # bounding box
         startX = rect.left()
         startY = rect.top()
         endX = rect.right()
         endY = rect.bottom()
         # ensure the bounding box coordinates fall within the spatial
         # dimensions of the image
         startX = max(0, startX)
         startY = max(0, startY)
         endX = min(endX, image.shape[1])
         endY = min(endY, image.shape[0])
         # compute the width and height of the bounding box
         w = endX - startX
         h = endY - startY
         # return our bounding box coordinates

       return (startX, startY, w, h)
```

Progress Report

Product-based Capstone Project

Load Dataset

```
[ ]  def image_face_processing(path, path_output):
       detector = dlib.get_frontal_face_detector()
       kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
       clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
       # detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
       # detector = cv2.CascadeClassifier(args["cascade"])
       files = os.listdir(path)

       if os.path.exists(path_output + "preprocess/") == False:
         os.mkdir(path_output + "preprocess/")

       for i in range(len(files)):
           # List of files in the sub-folder
           if os.path.exists(path_output + "preprocess/" + files[i] + "/") == False:
             os.mkdir(path_output + "preprocess/" + files[i] + "/")

           file_sub = os.listdir(os.path.join(directory_path + "train/", files[i]))  # Use os.path.join to construct the correct path

           for z in range(len(file_sub)):
             try:
               data = cv2.imread(os.path.join(path, files[i], file_sub[z]))
               faces = detector(data)
               crop = convert_and_trim_bb(data, faces)

               if sum(crop) > 1 :
                 faces_crop = data[crop[1]:crop[1]+crop[3], crop[0]:crop[0]+crop[2]]
                 photo = cv2.cvtColor(photo, cv2.COLOR_BGR2GRAY)
                 photo = cv2.equalizeHist(photo)
                 photo = clahe.apply(photo)
                 photo = cv2.filter2D(photo, -1, kernel)
                 cv2.imwrite(path_output + "preprocess/" + files[i] + "/" + file_sub[z], photo)

             except:
               pass
```

Progress Report
Product-based Capstone Project

```
image_face_processing(directory_path+"train/", directory_path)
```

```
[ ] ImgDataGenerator = tf.keras.preprocessing.image.ImageDataGenerator
```

```
def train_val_generators(TRAINING_DIR, VALIDATION_DIR):

    train_datagen = ImgDataGenerator(
                    rescale=1./255,
                    featurewise_center=True,
                    featurewise_std_normalization=True,
                    width_shift_range=0.2,
                    height_shift_range=0.2,
                    brightness_range=[0.25, 1.3],
                    rotation_range=20,
                    horizontal_flip=True,
                    fill_mode="nearest",
                    )

    train_generator = train_datagen.flow_from_directory(directory=TRAINING_DIR,
                                                        batch_size=150,
                                                        class_mode="binary",
                                                        target_size=(160, 160))

    validation_datagen = ImgDataGenerator( rescale=1./255,)

    validation_generator = validation_datagen.flow_from_directory(directory=VALIDATION_DIR,
                                                                  batch_size=100,
                                                                  class_mode="binary",
                                                                  target_size=(160, 160))
    return train_generator, validation_generator
```

```
[ ] train_generator, test_generator = train_val_generators(directory_path + "dataset/", directory_path + "preprocess/")
```

```
Found 4 images belonging to 1 classes.
Found 91 images belonging to 11 classes.
```

Progress Report

Product-based Capstone Project

```
[ ]  num_variations = 38
     X_train, y_train = [], []

     for i in range(num_variations):
         # Generate a batch of one image
         val = random.randint(0, 3)
         a, b = train_generator.next()
         X_train.append(a[0])
         X_train.append(a[2])
         X_train.append(a[1])
         X_train.append(a[3])

         y_train.append(b[0])
         y_train.append(b[2])
         y_train.append(b[1])
         y_train.append(b[3])

     X_train = np.array(X_train)
     y_train = np.array(y_train)
     X_test, y_test = test_generator.next()
```

```
[ ]  print(X_train.shape)
     for n in range(len(y_test)):
       if y_test[n] != 0:
         y_test[n] = 1
     print(y_test)
```

```
(152, 160, 160, 3)
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

Progress Report

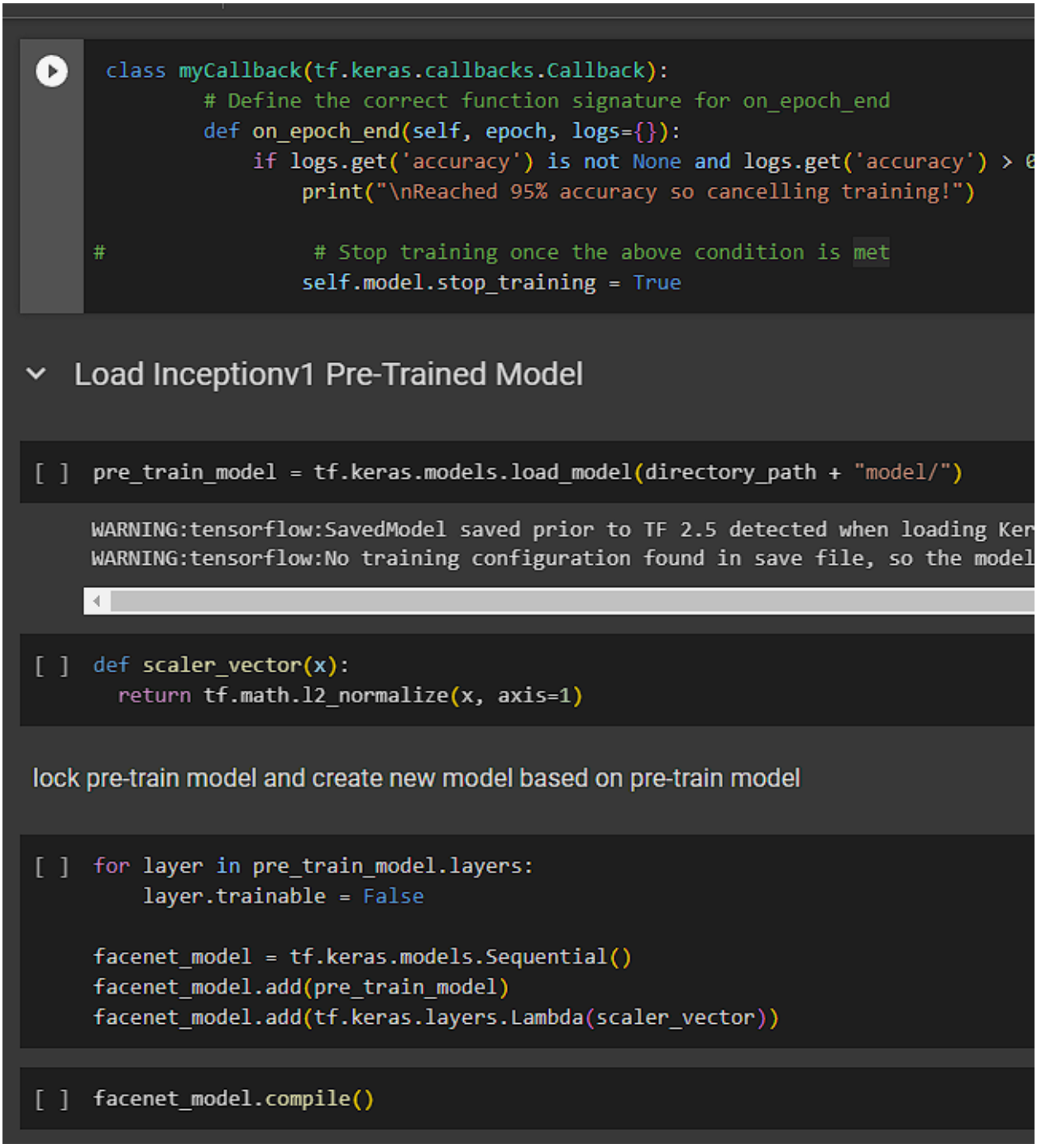
Product-based Capstone Project

```
fig = plt.figure(figsize=(10, 7))


for m in range(25):
  fig.add_subplot(5, 5, m+1)
  plt.imshow(X_train[m])
```

Progress Report

Product-based Capstone Project



```
class myCallback(tf.keras.callbacks.Callback):
        # Define the correct function signature for on_epoch_end
        def on_epoch_end(self, epoch, logs={}):
            if logs.get('accuracy') is not None and logs.get('accuracy') > 0
                print("\nReached 95% accuracy so cancelling training!")

#                # Stop training once the above condition is met
                self.model.stop_training = True
```

Load Inceptionv1 Pre-Trained Model

```
[ ] pre_train_model = tf.keras.models.load_model(directory_path + "model/")
```

```
WARNING:tensorflow:SavedModel saved prior to TF 2.5 detected when loading Ker
WARNING:tensorflow:No training configuration found in save file, so the model
```

```
[ ] def scaler_vector(x):
      return tf.math.l2_normalize(x, axis=1)
```

lock pre-train model and create new model based on pre-train model

```
[ ] for layer in pre_train_model.layers:
        layer.trainable = False

    facenet_model = tf.keras.models.Sequential()
    facenet_model.add(pre_train_model)
    facenet_model.add(tf.keras.layers.Lambda(scaler_vector))
```

```
[ ] facenet_model.compile()
```

Progress Report
Product-based Capstone Project

```
tf.keras.models.save_model(facenet_model,'')
# Now convert best model to tensorflow lite format
converter = tf.lite.TFLiteConverter.from_keras_model(facenet_model)
tflite_model = converter.convert()

# #Now save model
with open("embedding.tflite","wb") as f:
  f.write(tflite_model)
```

```
facenet_model.summary()
```

Prepare Data for Embedding

```
[ ]  def img_to_encoding(train_data, model):
        img = np.array(train_data) / 255.
        x_train = np.expand_dims(img, axis=0) # add a dimension of 1 as first dimension
        embedding = model.predict(x_train)
        print(np.linalg.norm(embedding))
        return embedding
        # return embedding / np.linalg.norm(embedding, ord=2) # scale embedding result so the magnitude of vector value is 1
```

Calculate the distance between 2 vector. smaler is better, smaller is more similiar

```
[ ]  def image_load(path):
        img = tf.keras.preprocessing.image.load_img(path, target_size=(160, 160))
        return img
```

```
[ ]  def verify(image1, image2):
        dist = (np.linalg.norm(tf.subtract(image1, image2), ord=2))
        if dist < 0.98:
            print("It's same, welcome in!")
        else:
            print("It's not same, please go away")
        return dist
```

Progress Report

Product-based Capstone Project

```
# data1 = facenet_model.predict(np.expand_dims(image_load(directory_path + "preprocess/dewanata/WhatsApp Image 2023-11-14 at 15.56.13.jpeg"), axis=0))
# data2 = facenet_model.predict(np.expand_dims(image_load(directory_path + "preprocess/dewanata/WhatsApp Image 2023-11-14 at 15.56.05.jpeg"), axis=0))
data1 = img_to_encoding(image_load(directory_path + "preprocess/dewanata/WhatsApp Image 2023-11-14 at 15.56.13.jpeg"),facenet_model)
data2 = img_to_encoding(image_load(directory_path + "preprocess/dewanata/WhatsApp Image 2023-11-14 at 15.56.05.jpeg"),facenet_model)
# print(data1.shape)
# embed1 = data1 / np.linalg.norm(data1, ord=2)
# embed2 = data2 / np.linalg.norm(data2, ord=2)
verify(data1, data2)
```

```
1/1 [==============================] - 0s 109ms/step
0.99999994
1/1 [==============================] - 0s 91ms/step
0.99999994
It's same, welcome in!
0.93208617
```

Embedding train and test data to get the vector from image data

```
train = facenet_model.predict(X_train)
test = facenet_model.predict(X_test)
```

```
5/5 [==============================] - 10s 2s/step
3/3 [==============================] - 4s 1s/step
```

```
new_model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, input_shape=(128,)),
    tf.keras.layers.Dense(1, activation='softplus')
])

new_model.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.Adam(0.001), metrics=['accuracy'])
new_model.fit(train, y_train, epochs=5, validation_data=(test, y_test))
```

```
Epoch 1/5
3/3 [==============================] - 1s 107ms/step - loss: 0.8667 - accuracy: 0.0750 - val_loss: 0.7780 - val_accuracy: 0.1373
Epoch 2/5
3/3 [==============================] - 0s 29ms/step - loss: 0.5451 - accuracy: 0.9875 - val_loss: 1.0488 - val_accuracy: 0.0000e+00
Epoch 3/5
3/3 [==============================] - 0s 20ms/step - loss: 0.3652 - accuracy: 1.0000 - val_loss: 1.3111 - val_accuracy: 0.0000e+00
Epoch 4/5
3/3 [==============================] - 0s 21ms/step - loss: 0.2542 - accuracy: 1.0000 - val_loss: 1.5609 - val_accuracy: 0.0000e+00
Epoch 5/5
3/3 [==============================] - 0s 20ms/step - loss: 0.1836 - accuracy: 1.0000 - val_loss: 1.7955 - val_accuracy: 0.0000e+00
```

Progress Report

Product-based Capstone Project

```
predict_data = img_to_encoding(image_load(directory_path+"preprocess/dewanata/WhatsApp Image 2023-11-14 at 15.56.13.jpeg"),facenet_model)
predict_data2 = img_to_encoding(image_load(directory_path+"preprocess/dewi/1.jpg"),facenet_model)
```

```
1/1 [==============================] - 0s 171ms/step
0.99999994
1/1 [==============================] - 0s 156ms/step
1.0
```

```
new_model.predict(predict_data)
```

```
1/1 [==============================] - 0s 26ms/step
array([[0.11945447]], dtype=float32)
```

Prepare vector by scaling

```
[ ] testing = []
    for i in test:
        testing.append( i / np.linalg.norm(i, ord=2) )

    test = np.array(testing)
    test_label = y_test
    print(test.shape)
    print(y_test.shape)

    training = []
    for i in train:
        training.append( i / np.linalg.norm(i, ord=2))

    train = np.array(training)
    train_label = y_train

    print(train.shape)
    print(train_label.shape)
```

```
(47, 128)
(47, 1)
(100, 128)
(100,)
```

Progress Report
Product-based Capstone Project

Single image for model to predict

Using IsolationForest or OneClassSVM for 1 class Classification Model

```
[ ]  oc_svm = svm.OneClassSVM(gamma=0.08, kernel='rbf', nu=0.01)  # Obtained using grid search
     if_clf = IsolationForest(contamination=0.025, max_features=1.0, max_samples=1.0, random_state = 3, n_estimators=200)  # Obtained using grid search
```

```
[ ]  oc_svm.fit(train,y_train)
     if_clf.fit(train,y_train)
```

```
IsolationForest
IsolationForest(contamination=0.025, max_samples=1.0, n_estimators=200,
                random_state=3)
```

Compare the result

```
[ ]  from sklearn.metrics import precision_score, recall_score, f1_score
     pred = if_clf.predict(test)
     pred[pred == 1] = 0
     pred[pred == -1] = 1

     precision = precision_score(y_test, pred)
     recall = recall_score(y_test, pred)
     f1 = f1_score(y_test, pred)

     print(f'Precision: {precision:.2f}')
     print(f'Recall: {recall:.2f}')
     print(f'F1 Score: {f1:.2f}')
```

Progress Report
Product-based Capstone Project

```
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')

print("________")

pred2 = oc_svm.predict(test)
pred2[pred2 == 1] = 0
pred2[pred2 == -1] = 1
precision = precision_score(y_test, pred2)
recall = recall_score(y_test, pred2)
f1 = f1_score(y_test, pred2)

print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
```

```
Precision: 0.92
Recall: 0.56
F1 Score: 0.70

________
Precision: 0.95
Recall: 0.93
F1 Score: 0.94
```

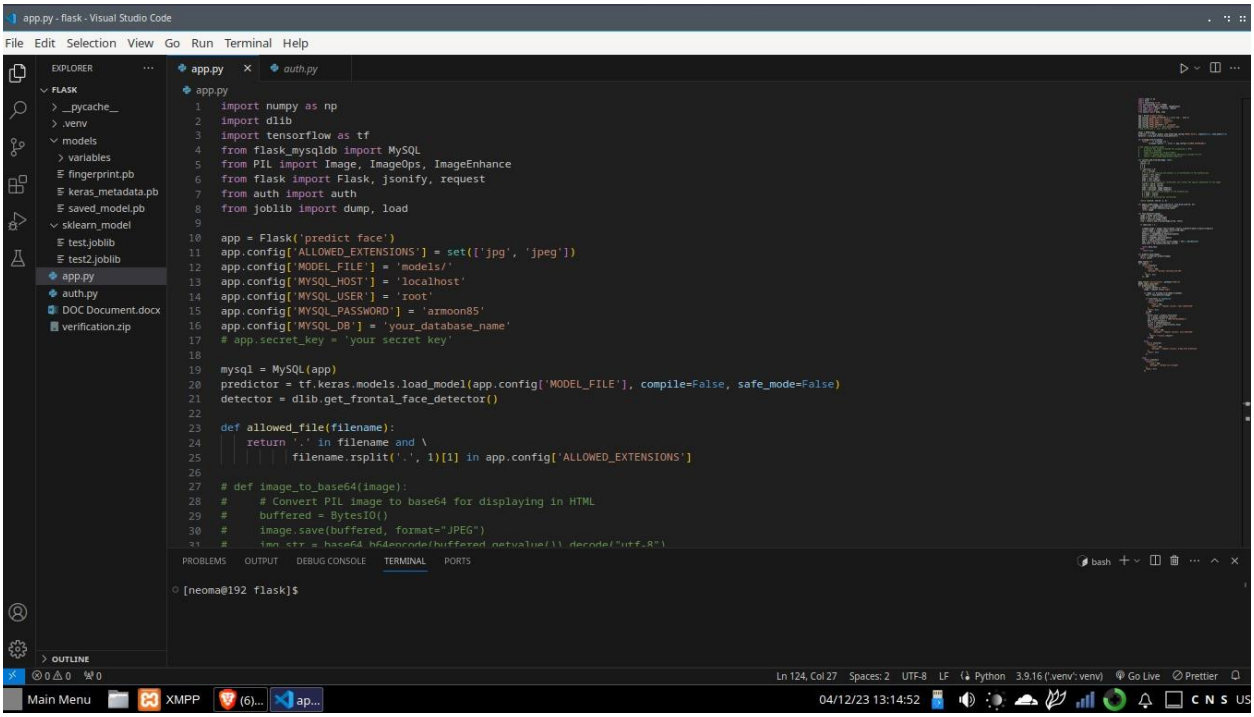
```
dump(if_clf, 'test.joblib')
```

```
['test.joblib']
```

```
if_test = load('test.joblib')
```

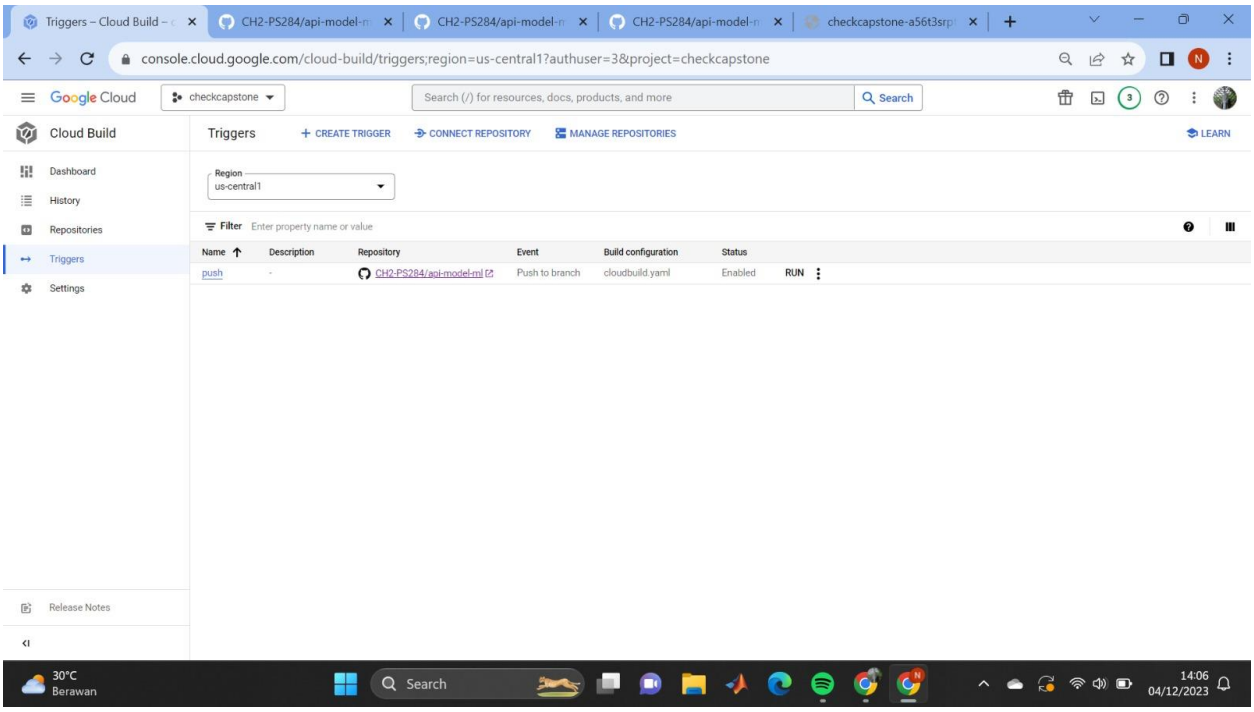
Progress Report

Product-based Capstone Project

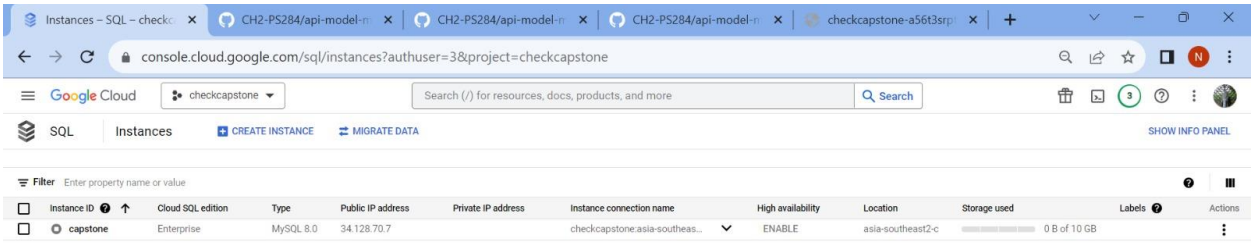


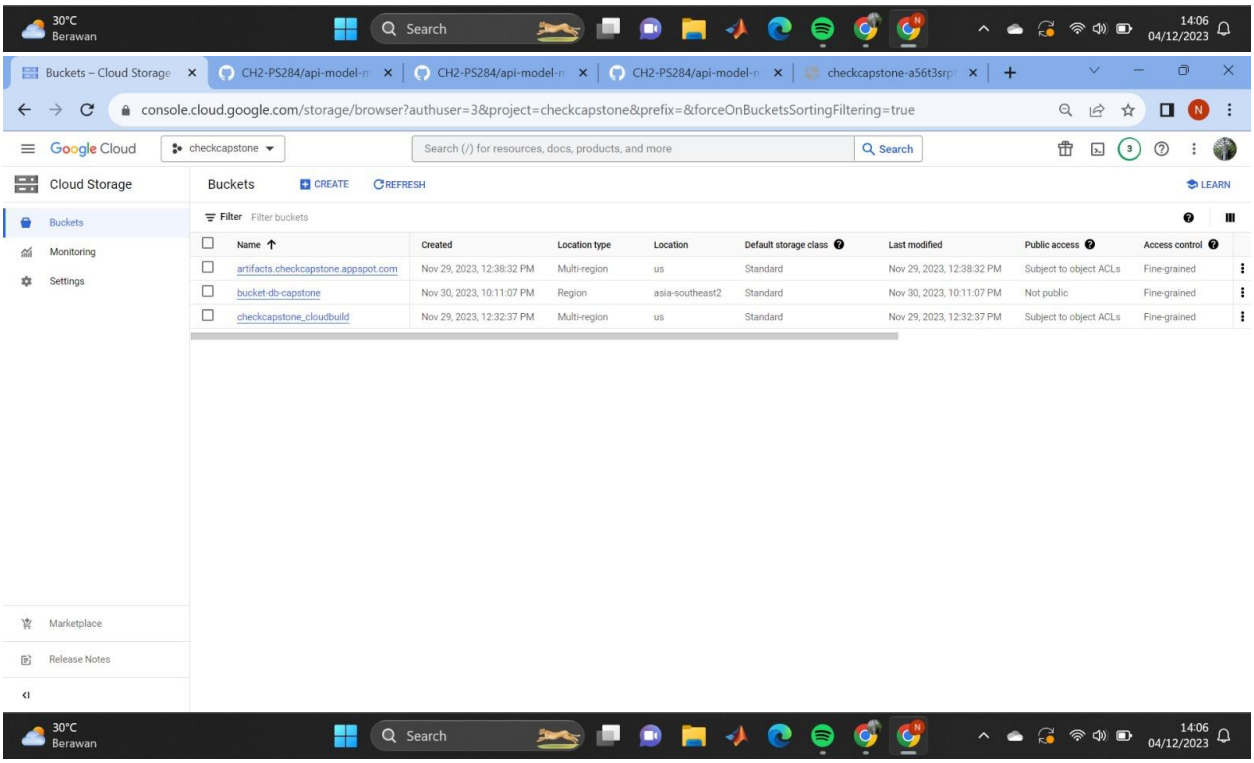
Progress Report
Product-based Capstone Project

Cloud Computing

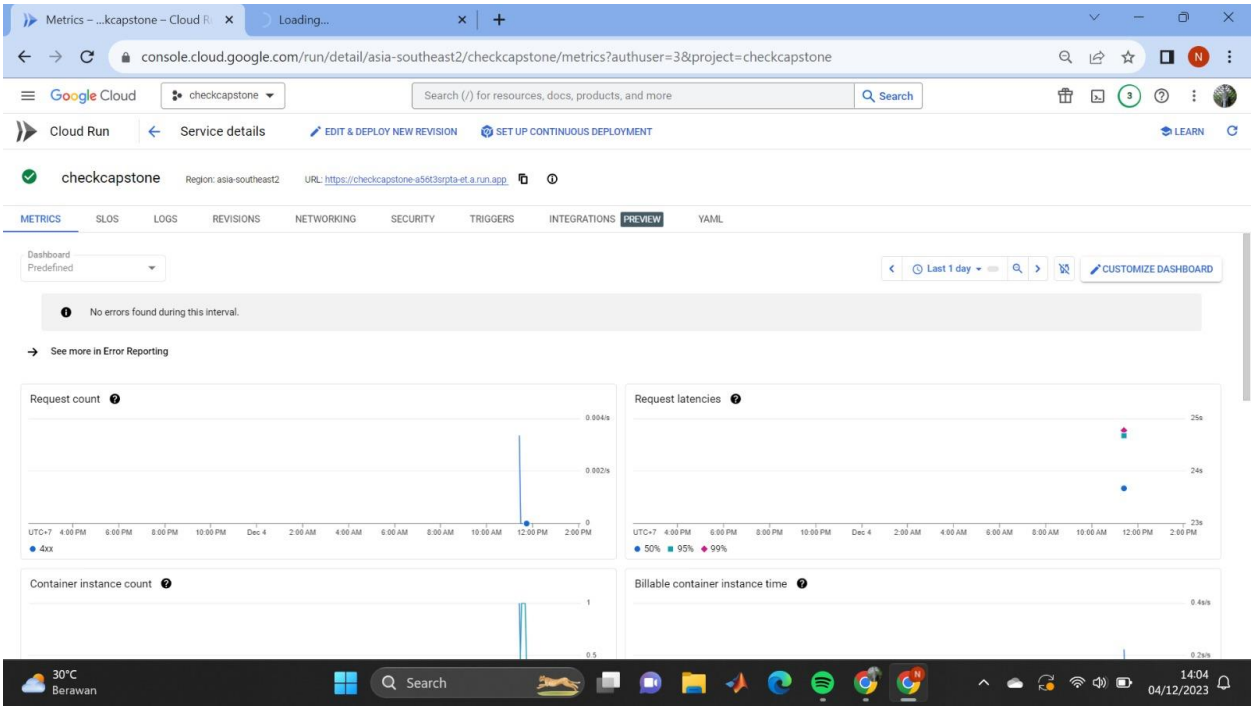


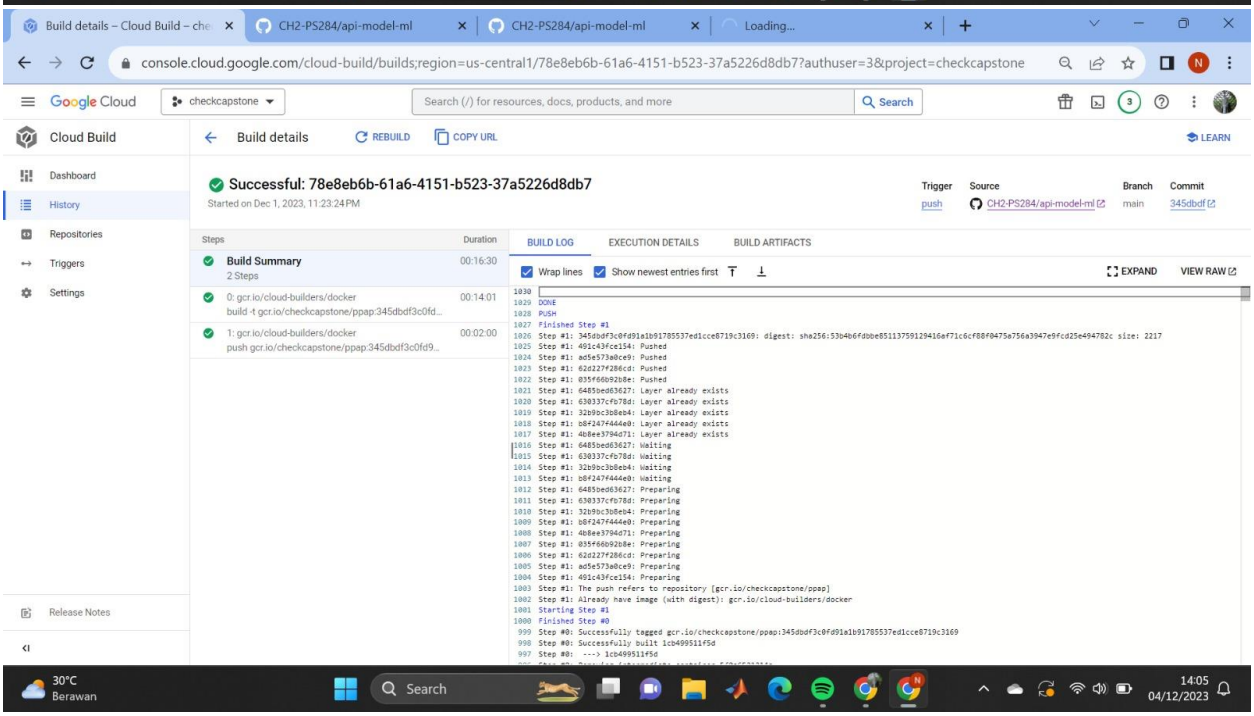
Progress Report
Product-based Capstone Project



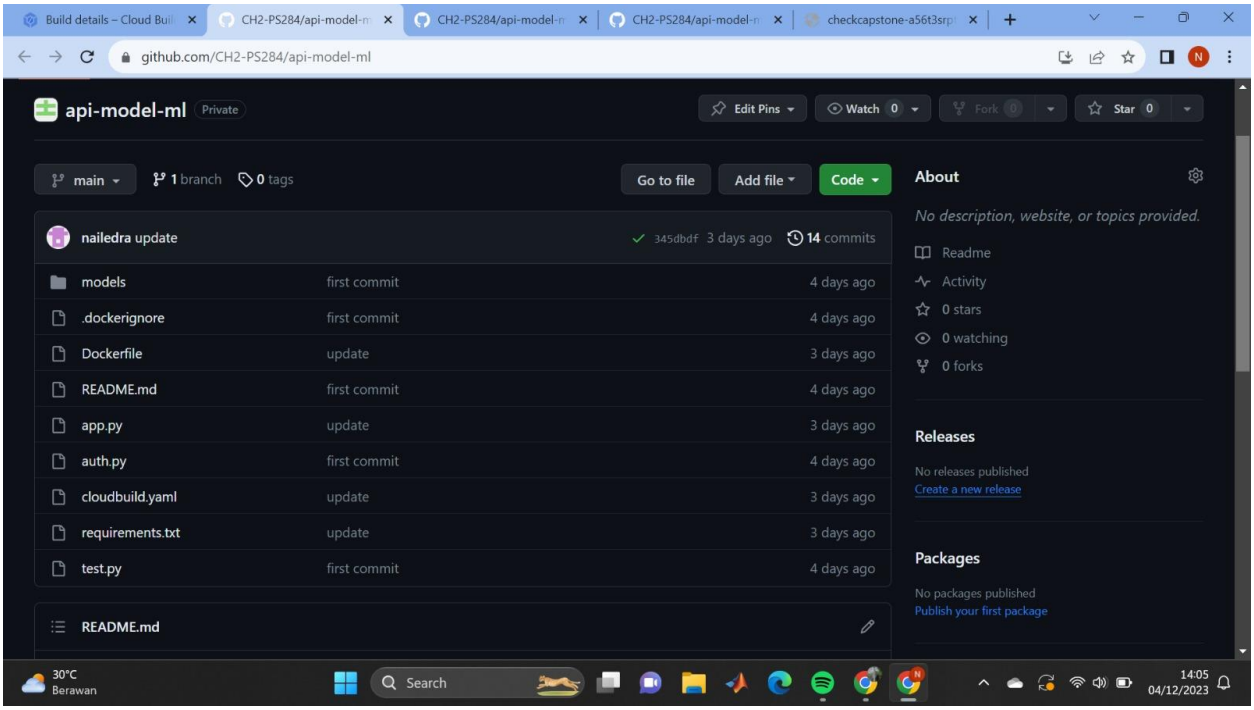


Progress Report
Product-based Capstone Project



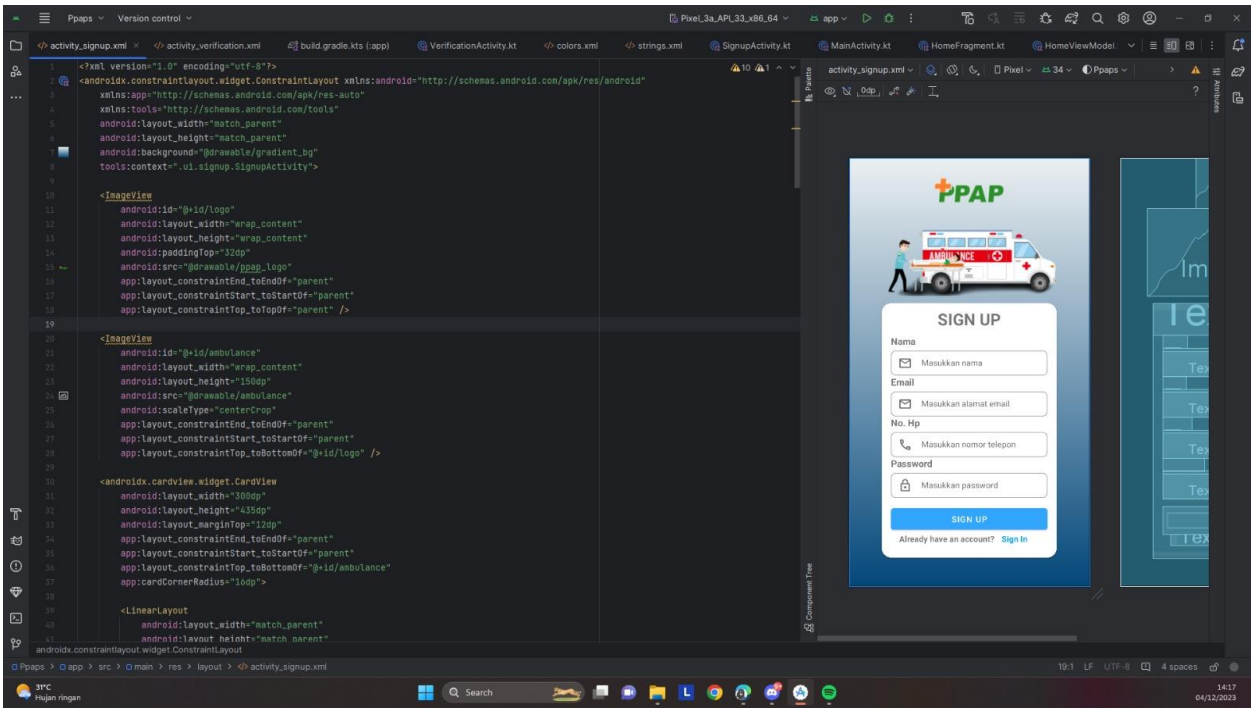


Progress Report
Product-based Capstone Project



Progress Report
Product-based Capstone Project

Mobile Development



Progress Report
Product-based Capstone Project