**Wafer Fault Detection Project**

**Data Validation Part**

Flask:-

trainRouteClient()

directory : "Training\_Batch\_Files"

train\_valObj = train\_validation(Training\_Batch\_Files \_path) #object initialization  
train\_valObj.train\_validation() #calling the training\_validation function

Training validation Insertion.py file

Class Train\_validation:

Def \_\_init\_\_(path):

self.raw\_data = Raw\_Data\_validation(path)  
self.file\_object =open("Training\_Logs/Training\_main\_log.txt","a+")  
self.dataTransform = dataTransform()  
self.dBOperation = dBOperation()  
self.log\_writer =logger.App\_Logger()

def train\_validation():

directory : Training Raw Data Validation

DataTypeValidation.py

class Raw data validation:

def values from schema():

return LengthOfDateStampInFile, LengthOfTimeStampInFile, column\_names, NumberofColumns

def manual regex creation():

regex = "['wafer']+['\\_'']+[\d\_]+[\d]+\.csv"  
return regex

def createDirectoryForGoodBadRawData():

def deleteExistingGoodDataTrainingFolder():

def deleteExistingBadDataTrainingFolder():

def moveBadFilesToArchiveBad():

def validationFileNameRaw(self,regex,LengthOfDateStampInFile,LengthOfTimeStampInFile):

def validateColumnLength(self,NumberofColumns):

def validateMissingValuesInWholeColumn(self):

Directory : DataTransform\_Training

DataTransformation.py

class dataTransform:

def replaceMissingWithNull():

Directory : DataTypeValidation\_Insertion\_Training

DataTypeValidation.py

Class dBOperation:

Def dataBaseConnection(dataBaseName):

Def CreateTableDb(DatabaseName,column\_names):

Def insertIntotableGoodData(Database):

Def selectingDataFramTableIntoCsv(database):

Model Training Part

Directory : Data\_Ingestion

Class trainModel():

Def trainingModel():