**LOW LEVEL DOCUMENTATION**

**COMMENT LINE REPORT**

BHAVYA JS

1.creating DB connection

client = MongoClient("localhost", port=27017)  
dbname = config.dbname  
collectionname=config.commentreportcn

2. Function to get all files.

def getallfiles(filespath, extensions): # function to return the file paths in the given directory with given extention  
 filelist = []  
 for root, dirs, files in os.walk(filespath):  
 for file in files:  
 if file.lower().endswith(tuple([item.lower() for item in extensions])):  
 filelist.append(os.path.join(root, file))  
 return filelist

3.Function to get application name of the file.

def fetch\_application(file):  
 colname=config.componenttypecn  
 col=client[dbname][colname]  
 application=list(col.find({"component\_name":file.split('\\')[-1]},{'id':0}))  
 application=application[0]['application']  
 return application

4. Function to get type of file .jsp, or .js or .java or .css.

def getExtensionType(file):  
 *"""  
 this function is to find extension type for files* ***:param*** *filename: passing filename to fetch extension type* ***:return****: we are returning extension type based on file type  
 """* f = open(file, 'r')  
 extension\_type = ""  
 # extension\_type = ''  
 if file.endswith(".js"):  
 extension\_type = "JAVA\_SCRIPT"  
 if file.endswith(".jsp"):  
 extension\_type = "JAVA\_SERVER\_PAGE"  
 if file.endswith(".css"):  
 extension\_type = "STYLE\_SHEET"  
  
 if file.endswith(".java"):  
  
 for line in f.readlines():  
 if line.strip().startswith("public class"):  
 if line.\_\_contains\_\_("HttpServlet"):  
 # print(line)  
  
 extension\_type = "SERVLET"  
 break  
 elif line.strip().\_\_contains\_\_("public interface"):  
 extension\_type = "INTERFACE"  
 break  
 else:  
 extension\_type = "JAVA\_CLASS"  
 if file.\_\_contains\_\_("DAO"):  
 extension\_type = "DATA\_ACCESS\_OBJECT"  
  
 return extension\_type

5. Function to find all comment lines and adding <br> tag after each comment line and adding it to separate json.

def getallreports(filespath):  
 *""" thi function collects all comment lines form a file and  
adds it into seperate json"""* METADATA = []  
 output1=''  
 comment\_lines=[]  
 comment\_flag=False  
 files = getallfiles(filespath, extentions)  
 for fi in files:  
 f = open(fi, "r")  
 #print(f)  
 for line in f.readlines():  
  
 #print(line)  
 if line.strip() == "":  
 continue  
  
 if fi.endswith(".js"):  
 #print(fi)  
 if (line.strip().\_\_contains\_\_('\*/') and line.strip().\_\_contains\_\_('/\*')):  
 comment\_lines.append(line)  
 comment\_flag = False  
 if line.strip().endswith("\*/"):  
 comment\_lines.append(line)  
 comment\_flag = False  
  
 if comment\_flag:  
 comment\_lines.append(line)  
 continue  
 if line.strip().startswith("/\*"):  
 comment\_lines.append(line)  
 comment\_flag = True  
 if line.strip().startswith("//"):  
 comment\_lines.append(line)  
 comment\_flag = False  
  
  
  
 if fi.endswith(".css"):  
 if line.strip().\_\_contains\_\_("\*/"):  
 comment\_lines.append(line)  
 comment\_flag= False  
 if comment\_flag:  
 comment\_lines.append(line)  
 continue  
 if line.strip().\_\_contains\_\_("/\*"):  
 comment\_lines.append(line)  
 comment\_flag = True  
  
  
 if fi.endswith(".java"):  
 if line.strip().endswith("\*/") :  
 comment\_lines.append(line)  
 comment\_flag = False  
  
 if comment\_flag:  
 comment\_lines.append(line)  
 continue  
 if line.strip().startswith("/\*"):  
 comment\_lines.append(line)  
 comment\_flag = True  
 if line.strip().startswith("//"):  
 comment\_lines.append(line)  
 comment\_flag = False  
  
 if fi.endswith('.html') or fi.endswith(".jsp"):  
  
 if line.strip().\_\_contains\_\_("<!--") and line.\_\_contains\_\_("-->"):  
 comment\_lines.append(line)  
 comment\_flag=False  
  
 output1="<br>".join(comment\_lines)  
 output1={  
 "application" : fetch\_application(fi),  
 "component\_type" : getExtensionType(fi),  
 "component\_name" : fi.split("\\")[-1].split(".")[0],  
 "codeString" :output1 }  
 METADATA.append(copy.deepcopy(output1))  
  
 #print(json.dumps(METADATA, indent=4))  
 return METADATA

6.Function to insert the json to DB.

def dbinsertfunction(filespath, dbname, collectionname):  
 *"""  
 this function is to update database by calling show code and getfiles functions* ***:param*** *dbname: database name from config file* ***:param*** *collectionname: collectionname from config file  
 """* col = client[dbname][collectionname]  
 output = getallreports(filespath)  
 if output != []:  
 if col.count\_documents({}) != 0:  
 col.drop()  
 print("Deleted the old", dbname, collectionname, "collection")  
  
 col.insert\_one({"type": "metadata",  
 "headers": ["component\_name", "component\_type", "application"]})  
 col.insert\_many(output)  
 print("Inserted the list of jsons of", dbname, collectionname)  
 else:  
 print("There are no jsons in the output to insert in the DB", dbname, collectionname)

7. calling function and writing output to excel file.

if \_\_name\_\_ == '\_\_main\_\_':  
 output = getallreports(filespath)  
 if not os.path.exists("outputs//"):  
 os.makedirs("outputs//")  
 json.dump(output , open('outputs\\cobol\_report.json', 'w'), indent=4)  
 pd.DataFrame(output).to\_excel("outputs\\cobol\_report.xlsx", index=False)  
 dbinsertfunction(filespath, dbname, collectionname)