**LOW LEVEL DOCUMENTATION**

**CRUD REPORT**

BHAVYA JS

1. To create database connection.

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

1. Function to create dictionary for DAO files with key as method name and value as list of lines in that method.

def createFunToDic(file,extentions):  
  
 *"""This function is used to create a dictionary for each methods in DAO files  
 with key as function name and values as list of method lines."""* output = {}  
 function\_name = ""  
 functions = []  
 flag = False  
 flag1 = False  
 count = 0  
 f = open(file, "r")  
 fi = file.split("\\")[-1].split(".")[0]  
 if fi.endswith("DAO"):  
  
 for line in f.readlines():  
  
 if (line.strip().startswith("protected") or line.strip().startswith("public") or line.strip().startswith("private")) and line.strip().\_\_contains\_\_("(") and line.strip().\_\_contains\_\_(")"):  
  
 function\_name = line.split("(")[0].split()[-1]  
 #print(function\_name)  
 flag=True  
 if flag and line.strip().\_\_contains\_\_("{"):  
 count += 1  
 flag1 = True  
 if flag and line.strip().\_\_contains\_\_("}"):  
  
 count -= 1  
  
 if flag and flag1 and count == 0:  
 flag=False  
 flag1=False  
 output[function\_name]=copy.deepcopy(functions)  
  
 functions.clear()  
 if flag:  
 functions.append(line)  
 #print(json.dumps(output,indent=4))  
 return output

3.Function to create list of Jsons for DAO files.

def dao\_files(filespath,extentions):  
  
 *"""This function captures the sql query,crud,tablename and function name  
 and creates a list of json for DAO files. """* files=getallfiles(filespath,extentions)  
 METADATA = []  
 for file in files:  
 items=createFunToDic(file,extentions)  
 sql=""  
 for item in items:  
 for line in items[item]:  
 if line.strip().\_\_contains\_\_("prepareStatement") or line.strip().\_\_contains\_\_("executeQuery"):  
 variable = line.split(")")[0].split("(")[-1]  
 if variable.startswith('"'):  
 sql=variable  
 else:  
 i = items[item]  
 sql = identifysql(i,variable)  
 break  
  
 if sql!="":  
 output=({"component\_name":file.split("\\")[-1].split(".")[0],"component\_type":getcomponenttype(file),  
 "function\_name":item,  
 "SQL":sql,  
 "CRUD":identifyCRUD(sql),"Table\_name":tablename(sql)})  
 METADATA.append(copy.deepcopy(output))  
 output.clear()  
 sql=""  
 return METADATA

4.Function to fetch SQL statement.

def identifysql(i,variable):  
  
 *"""This function captures the sql query from the methods"""* sql=""  
 for line in i:  
  
 if (line.strip().\_\_contains\_\_(variable) and line.strip().\_\_contains\_\_("=")and line.strip().\_\_contains\_\_('"') )or(line.strip().\_\_contains\_\_(variable) and line.strip().\_\_contains\_\_("=")and line.strip().\_\_contains\_\_('"')and line.strip().\_\_contains\_\_('+')):  
 sql=sql+line.split(variable)[1].split(";")[0]  
 sql=sql.replace("=","",1)  
 return sql

5. Function to fetch CRUD operation in the SQL query.

def identifyCRUD(sql):  
  
 *"""This function finds the CRUD operation in the found sql query and returns "READ" if  
 select is found in query,similarly "UPDATE" for update query,"CREATE" for insert query,  
 "DELETE" for delete query."""* crud=""  
 sql=sql.split('"')[1]  
  
 if sql.strip().lower().startswith("select"):  
 crud="READ"  
 elif sql.strip().lower().startswith("update"):  
 crud="UPDATE"  
 elif sql.strip().lower().startswith("insert"):  
 crud="CREATE"  
 else:  
 crud="DELETE"  
 return crud

6.Function to fetch table name from SQL query.

def tablename(sql):  
  
 *"""This function captures the table name from the given sql query  
 i.e. if select is found in sql query then the word next to from will be the table name,  
 similarly for update the word next to update will be the table name."""* table\_name=""  
 sql = sql.split('"')[1]  
  
 if sql.strip().lower().startswith("select") or sql.strip().lower().startswith("delete"):  
 table\_name=sql.lower().split("from")[-1].split()[0]  
 if table\_name.\_\_contains\_\_("("):  
 table\_name=table\_name.split("(")[0]  
 elif sql.strip().lower().startswith("insert"):  
 table\_name=sql.lower().split("into")[-1].split()[0]  
 if table\_name.\_\_contains\_\_("("):  
 table\_name=table\_name.split("(")[0]  
 else:  
 table\_name = sql.lower().split("update")[-1].split()[0]  
 return table\_name

7. Function to create crud collection from the servlet files.

def getallreports(filespath,extentions):  
  
 *"""This function reads all the servlet files and finds  
 the DAO files triggering point and creates a database for all the files. """* metadata=[]  
 crud=dao\_files(filespath,extentions)  
 files=getallfiles(filespath,extentions)  
 name=""  
 for file in files:  
 f=open(file,"r")  
 fi = file.split("\\")[-1].split(".")[0]  
 if fi.endswith("DAO"):  
 continue  
 else:  
 for line in f.readlines():  
 for name in crud:  
 funName = name["function\_name"]  
 fName = name["component\_name"]  
 fileName = fName.split(".")[0]  
 if line.strip().\_\_contains\_\_(fileName)and line.strip().\_\_contains\_\_("new"):  
 names=line.split("=")[0].split()[-1]  
 if line.strip().\_\_contains\_\_(names+"."+funName+"(") and line.strip().\_\_contains\_\_(")") and line.strip().\_\_contains\_\_(";"):  
 output1=({"component\_name":file.split("\\")[-1],"component\_type":getcomponenttype(file),  
 "CRUD":name["CRUD"],"SQL":name["SQL"],"Table\_name":name["Table\_name"]})  
  
 if output1 not in metadata:  
 metadata.append(copy.deepcopy(output1))  
 output1.clear()  
 break  
 #print(json.dumps(metadata,indent=4))  
 return metadata

8. Function to insert the collection to database.

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,extentions)  
 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)

9 .Function call and code to create excel for output.

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