**LOW LEVEL DOCUMENTATION**

**ORPHAN REPORT**

BHAVYA JS

1.creates DB connection.

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

2.Function to get reports and stores as list.

def getallreports():  
  
 *"""This function fetches masterinventory and crossreference  
 from DB and stores as list."""* colname\_master = config.materinventorycn  
 colname\_xref=config.crossreferencecn  
 col\_master = client[dbname][colname\_master]  
 col\_xref=client[dbname][colname\_xref]  
  
 masterinventory = list(col\_master.find({'type': {"$ne": "metadata"}},{"component\_name":1,"component\_type":1, "\_id":0}))  
 crossref = list(col\_xref.find({'type': {"$ne": "metadata"}},{"called\_name":1,"called\_type":1,"\_id":0}))  
  
 return Missingreport(masterinventory,crossref)

3.Function to store the ‘called name’, ‘called type’and’called app name’ in separate set named master, similarly stores ‘component name’ ,‘component type’ and ‘application’ in a set named Xref. And compares the sets and adds it to list that are not found.

def orphanreport(masterinventory,crossref):  
  
 *"""This function compares crossreference with masterinventory  
 and finds the missing data and stores it in list. """* orphanreports=[]  
 master = set()  
 Xref=set()  
 for i in masterinventory:  
 a = i["component\_name"]  
 a = a.split(".")[0]  
 master.add(a +"," + i["component\_type"] + "," +i["application"])  
 #print(master)  
 for j in crossref:  
 Xref.add(j["called\_name"] +"," + j["called\_type"] + "," +j["called\_app\_name"])  
 #print(Xref)  
  
 for item in master:  
 #print(item)  
 if item not in Xref:  
 #print(item)  
 orphanreports.append(item)  
 #print(orphanreports)  
 #print(orphanreports)  
 return orphanreport\_json(orphanreports)

4. Function to create a json for missing reports.

def orphanreport\_json(orphanreports):  
  
 *"""This function stores all missing reports in a dictionary"""* orphanreportJson=[]  
 for item in orphanreports:  
 orphanreportDict={ }  
 component\_name=item.split(",")[0]  
 component\_type=item.split(",")[1]  
 application=item.split(",")[2]  
 orphanreportDict["called\_name"] = component\_name  
 orphanreportDict["called\_type"] = component\_type  
 orphanreportDict["called\_app\_name"] = application  
 #print(orphanreportDict)  
 orphanreportJson.append(orphanreportDict)  
 return orphanreportJson

5. Function to insert the created json 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()  
 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)