1.

client = MongoClient('localhost'**, 27017**)  
dbname = config.dbname  
collectionname = config.masterinventorycn  
colname\_app =config.componenttypecn  
  
filespath = config.filespath  
extentions = [".jsp"**,** ".java"**,** ".css"**,** ".js"]

Connection to MongoDB and file path, extensions, database name, collection from config file.

2.

def get\_files(filespath**,** extentions): # function to return the file paths in the given directory with given extention  
 filelist = []  
 for r**,** d**,** f in walk(filespath):  
 #print(r, d, f)  
 for file in f:  
 for extention in extentions:  
 if file.endswith(extention) and file.count(".") == extention.count("."):  
 filepath = join(r**,** file)  
 filelist.append(filepath)  
 #print(filepath)  
 #dead\_para\_count(filepath)  
 #print(filelist)  
 return filelist

This function is to fetch all required files from folders using given extensions.

3.

def fetch\_application(file): ###function to fetch application name from database  
  
 col = client[dbname][colname\_app]  
 #file=file.split('\\')[-1]  
 application=list(col.find({"component\_name" : file.split('\\')[-**1**]}**,**{'\_id':**0**}))  
 application=application[**0**]['application']  
 return application

This function is to fetch application type from database for each file.

4.

def getExtensionType(file): ##this function is to fetch component type from database  
  
 col = client[dbname][colname\_app]  
 #file=file.split('\\')[-1]  
 component\_type=list(col.find({"component\_name" : file.split('\\')[-**1**]}**,**{'\_id':**0**}))  
 component\_type=component\_type[**0**]['component\_type']  
 return component\_type

This function is to fetch extension type from database for each file.

5.

def master\_inventory(filespath):  
 *"""  
 1.this function is to fetch blank lines, comment lines, executable lines, total lines from all java, css, and jsp files* ***:return****: returns list of jsons for each file.  
 """* files = get\_files(filespath**,** extentions)  
 file = ''  
 METADATA = []  
  
 for file in files:  
 comment\_lines = **0** blank\_lines = **0** comment\_flag = False  
 Sloc = **0** Loc = **0** total\_lines = []  
 block\_comments = **0** html\_comment = **0** html\_comment\_flag = False  
 total\_comment\_lines = **0** f = open(file**,** 'r')  
 for line in f.readlines():  
  
 total\_lines.append(line)  
 if line.strip() == '':  
 blank\_lines = blank\_lines + **1** if file.strip().endswith('.java'):  
  
 if line.strip().startswith("//"):  
 comment\_lines = comment\_lines + **1** if line.strip().\_\_contains\_\_('/\*'):  
 comment\_flag = True  
 if line.strip().\_\_contains\_\_('\*/'):  
 block\_comments=block\_comments+**1** comment\_flag = False  
 if comment\_flag:  
 block\_comments = block\_comments + **1** if line.\_\_contains\_\_('/\*') and line.\_\_contains\_\_('\*/'):  
 block\_comments=block\_comments+**1** if file.strip().endswith('.css') or file.strip().endswith('.jsp'):  
  
 if line.strip().\_\_contains\_\_("<!--"):  
 html\_comment\_flag = True  
 if html\_comment\_flag and line.strip().\_\_contains\_\_("-->"):  
 html\_comment\_flag = False  
 if line.\_\_contains\_\_('<!--') and line.\_\_contains\_\_('-->'):  
 html\_comment=html\_comment+**1** if html\_comment\_flag:  
 html\_comment = html\_comment + **1** Loc = len(total\_lines)  
 total\_comment\_lines = block\_comments + html\_comment + comment\_lines  
 Sloc = Loc - total\_comment\_lines - blank\_lines  
  
 output = {"component\_name": file.split('\\')[-**1**]**,** "component\_type": getExtensionType(file)**,** "Loc": Loc**,** "commented\_lines": total\_comment\_lines**,** "blank\_lines": blank\_lines**,** "Sloc": Sloc**,** "Path": ""**,** "application":fetch\_application(file)**,** "orphan": ""**,** "Active": ""**,** "execution\_details": ""**,** "no\_of\_variables": ""**,** "no\_of\_dead\_lines":  
 ""**,** "cyclomatic\_complexity": ""**,** "FP": ""**,** "dead\_para\_count": ""**,** "dead\_para\_list":  
 ""**,** "total\_para\_count": total\_para\_count(file)**,** "comments": ""}  
 METADATA.append(copy.deepcopy(output))  
 total\_lines.clear()  
 # print(json.dumps(METADATA,indent=4))  
 return METADATA

This function is to fetch empty lines, comment lines, Sloc, Loc, application, extension type for each file.

6.

def dbinsertfunction(dbname**,** collectionname**,**filespath):  
 *"""  
 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  
 """* output = master\_inventory(filespath)  
  
 #total\_files = get\_files(filespath,extentions)  
 # print(output)  
 col = client[dbname][collectionname]  
 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"**,** "codeString"  
  
 ]})  
 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)

this to insert output into database, only if output is not empty.