1.

client = MongoClient('localhost'**, 27017**)  
  
dbname = config.dbname  
collectionname = config.validationreportcn  
validation\_tags = ['required'**,** 'maxlength'**,** 'pattern']  
  
filespath = config.filespath  
extentions = [".jsp"**,** ".js"]

connection to MongoDB and files path and extensions from config to get all required files

2.

def getallfiles(filespath**,** extentions): # 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 extentions])):  
 filelist.append(os.path.join(root**,** file))  
 return filelist

This function is to iterate over all required folders and fetch files with given extensions.

3.

def get\_screenfield\_data():  
 *"""  
 1.In this function we are getting screenfield data from database  
 2.If screenfield is not empty then we are creating json for validation report* ***:return****: returns validation report json's for non empty screen fields  
 """* col = client[dbname]['screenfields']  
 validation\_dict = {}  
 output = []  
 screenfields\_list = []  
 screenfileds\_data = list(col.find({'type': {"$ne": "metadata"}}**,** {'\_id': **0,** 'application': **1,** 'filename': **1,** 'screenfield': **1,** 'required': **1,** 'maxlength': **1,** 'pattern': **1**}))  
 for data in screenfileds\_data:  
 if data['screenfield'] != '':  
 # print(data['screenfield'])  
 screenfields\_list.append(data)  
 validation\_dict['filename'] = data['filename']  
 validation\_dict['application'] = data['application']  
 validation\_dict['screenfield'] = data['screenfield']  
 validation\_dict['propertyvalidation'] = property\_validation(data)  
 validation\_dict['CodeValidation'] = code\_validators(data)  
 output.append(copy.deepcopy(validation\_dict))  
 validation\_dict.clear()  
  
 # print(json.dumps(output, indent=4))  
 # print(len(output))  
 # print(len(screenfields\_list))  
 return output

This function is to fetch screenfield collection. If screenfield is not empty then we create a json with property validation, code validation, file name, application and screenfield.

4.

def property\_validation(record):  
 *"""  
 In this function we are adding non empty validation properties.* ***:param*** *record: record is a non empty screenfield json* ***:return****: returns a string by adding validation properties  
 """* validation\_property = ''  
 if record['pattern'] == '' and record['maxlength'] == '' and record['required'] == '':  
 validation\_property = ''  
 if record['pattern'] != '' and record['maxlength'] != '' and record['required'] != '':  
 validation\_property = "pattern=" + record["pattern"] + "," + 'maxlength=' + record[  
 'maxlength'] + "required=" + str(record["required"])  
  
 if record['pattern'] != '' and record['maxlength'] != '' and record['required'] == '':  
 validation\_property = "pattern=" + record["pattern"] + "," + "malength=" + record[  
 'maxlength']  
  
 if record['pattern'] != '' and record['maxlength'] == '' and record['required'] != '':  
 validation\_property = "pattern=" + record["pattern"] + "," + "required=" + str(  
 record["required"])  
  
 if record['pattern'] == '' and record['maxlength'] != '' and record['required'] != '':  
 validation\_property = "maxlength=" + record["maxlength"] + "," + "required=" + str(  
 record["required"])  
  
 if record['pattern'] != '' and record['maxlength'] == '' and record['required'] == '':  
 validation\_property = "pattern=" + record["pattern"]  
  
 if record['pattern'] == '' and record['maxlength'] != '' and record['required'] == '':  
 validation\_property = "maxlength=" + record["maxlength"]  
  
 if record['pattern'] == '' and record['maxlength'] == '' and record['required'] != '':  
 validation\_property = "required=" + str(record["required"])  
 return validation\_property

This function will take one by one non empty screenfield and adds all validation properties required and returns a validation property string.

5.

def code\_validation(id**,** file):  
 *"""  
 This function searches for script tag and src attribute in jsp files and if there is any if block with screenfiled id then  
 we are fetching that if block.* ***:param*** *id: id is creenfield* ***:param*** *file: jsp file having that screenfield* ***:return****: returns a list of if block , if present otherwise it returns a empty string  
 """* f = open(file**,** 'r')  
 screenfield = id  
 js\_files = getallfiles(filespath**,** extentions=['.js'])  
 codeValidation = []  
 script\_flag = False  
 if\_flag = False  
 count = **0** for line in f.readlines():  
 if line.\_\_contains\_\_('<script>'):  
 script\_flag = True  
 # print(line,file)  
  
 if line.\_\_contains\_\_('</script'):  
 # print(line)  
 script\_flag = False  
 if script\_flag:  
 if line.\_\_contains\_\_('if') and line.\_\_contains\_\_(screenfield):  
 print(line)  
 if\_flag = True  
 if if\_flag:  
 if\_script\_list.append(line)  
 if line.\_\_contains\_\_('{'):  
 count += **1** if line.\_\_contains\_\_('}'):  
 count -= **1** if count == **0** and if\_flag:  
 if\_flag = False  
 codeValidation = if\_script\_list  
  
 if line.\_\_contains\_\_('script') and line.\_\_contains\_\_('src'):  
 if\_js\_flag = False  
 if\_js\_count = **0** if\_js\_list = []  
 for js\_file in js\_files:  
 # print(js\_file)  
 if line.\_\_contains\_\_('js/' + js\_file.split("\\")[-**1**]) and not line.\_\_contains\_\_('http'):  
 file\_js = open(js\_file)  
 for line in file\_js.readlines():  
 if line.strip().startswith('if') and line.\_\_contains\_\_(screenfield):  
 # print(line)  
 if\_js\_flag = True  
 if if\_js\_flag:  
 if\_js\_list.append(line)  
 if line.\_\_contains\_\_('{'):  
 if\_js\_count += **1** if line.\_\_contains\_\_('}'):  
 if\_js\_count -= **1** if if\_js\_count == **0** and if\_js\_flag:  
 if\_js\_flag = False  
 if if\_js\_list == []:  
 codeValidation = ''  
 else:  
 codeValidation = if\_js\_list  
  
 return codeValidation

This function takes screenfield id and required jsp file as parameters. Then it will searches for script tag and src attribute. If there is any if block between script tag with screen field id then that block is considered as code validation. If there is src attribute, we open that particular js file and search for if block with screenfield id. If there is any such if block then that will be our code validation list.

6.

def code\_validators(record):  
 *"""  
 this function is to take non empty screenfield and fetching particular file where screenfield is present* ***:param*** *record: non empty screenfiled json* ***:return****: returns a code validation list or empty string if not present  
 """* result = []  
 validation\_code = []  
 files = getallfiles(filespath**,** extentions)  
  
 if record['screenfield'] != '':  
 for file in files:  
 if file.split("\\")[-**1**] == record['filename']:  
 result = code\_validation(record['screenfield']**,** file)  
 if result != []:  
 validation\_code = result  
  
 if result == []:  
 validation\_code = str('')  
 return validation\_code

This function takes one by one non empty screenfield and searches for file name in screenfield record in our files list. If condition satisfies then that id and file name is sent to above function.

def dbinsertfunction(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  
 """* output = get\_screenfield\_data()  
  
 # 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": [  
 "filename"**,** "application"**,** "screenfield"**,** "propertyvalidation"**,** "codeValidation"  
  
 ]})  
 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 function is to update database if output is not empty.

if \_\_name\_\_ == '\_\_main\_\_':  
 output = get\_screenfield\_data()  
 dbinsertfunction(dbname**,** collectionname)  
 pd.DataFrame(output).to\_excel("outputs\\validation\_report.xlsx"**,** index=False)  
 json.dump(output**,** open('outputs\\validation\_report.json'**,** 'w')**,** indent=**4**)

code to call main function and it creates json file and excel sheet for our output.