**Documentation for Automation of Invigilator Duty Chart Generator Project**

## **//RULES FOR VALIDATION:**

1. Date should be in the format DD/MM/YY, DD-MM-YY, DD/MM/YYYY.
2. Names can be in uppercase, lowercase, camelcase.
3. Names length cannot be greater than 25.
4. Number of professors cannot be greater than 100.
5. Accept suffix names from this array:  
     
   String [“Mr”, “Mrs”, “Dr”, “Smt”, “Shri”, “Miss” ];
6. Department should be the following:

String [“Computer science”,”Computer technology and Applications”, “Information Technology”, “Civil”, “Electrical Communication”, “Mechanical”, “BioMedical Engineering”, “Industrial production engineering”, “Electrical engineering”, “Electronics and instrumentation”, “Humanities”, “Applied Mathematics”,”Applied Physics”,”Applied Chemistry”,”Management Studies”,”Pharmacy”, “CSE”, “IT”, “CE”, “EC”, “ME”, “BME” , “IPE”, “EE”, “EI”, “HU”, “AM”,”AC”,”MS”,”AP”];

# **Pseudo Code**

**Class header**

{

Date date;

int noOfStudents;

int noOfInvigilators;

Date getDate()

{

Return date;

}

Date setDate(Date newDate)

{

this.date = newDate;

}

int getNoOfStudents()

{

Return NoOfStudents;

}

int setNoOfStudents(int newNoOfStudents)

{

this.noOfStudents = newNoOfStudents;

}

int getNoOfInvigilators()

{

Return NoOfInvigilators;

}

int setNoOfInvigilators(int newNoOfInvigilators)

{

this.noOfInvigilators = newNoOfInvigilators;

}

}

**Class Faculty**{

String name;

int age;

String department;

String designation;

Boolean availability;

List duty;

String getName(){

Return Name;

}

String setName(String newName){

this.name = newName;

}

int getAge(){

Return Name;

}

int setAge(String newAge){

this.age = newAge;

}

String getDepartment(){

Return Name;

}

String setDepartment(String newDepartment){

this.department = newDepartment;

}

String getDesignation(){

Return designation;

}

String setDesignation(String newDesignation){

this.designation = newDesignation;

}

Boolean getAvailability(){

Return availability;

}

Boolean setAvailability(String newAvailability){

this.availability = newAvailability;

}

}

**Class Read\_file**{

Public:

Read\_file(string file\_path1, file\_path2, file\_path3){

//files : Ms excel(xlsx,xls) , Google sheet(xlsx,xls), Libreoffice (odf), Numbers(numbers)

String Path1 = file\_path1;

String Path2 = file\_path2;

String Path3 = file\_path3;

}

list Read\_xlsx(Path){

if (**isNamePresent**(name col)){

Listname = store(name col);

}

if (**isDesignationValid**(designation col)){

Listdes = store(des col);

}

if (**isDepartmentValid**(department col)){

Listdep = store(dep col);

}

if (**isDateValid**(date col)){

Listdate = store(date col);

}

Return lists;

}

list Read\_odf(string file\_path){

}//similar functions as Read\_xlsx

list Read\_num(string file\_path){}//similar functions as Read\_xlsx

}

List store( data\_field col);//stores the columns into array list

**Class Validate**{

Public File arr[]={File1,File2,File3}

**Bool isNamePresent()**

//This function checks that Name column is present in the excel file .

{

for( file : arr)

{

for(int j=0; j<=file.getPhysicalNumberofColumns; j++)

{

colName=file.getColumn(j);

if(colName==Name)

{

Return true;

}

Else{

Return false;

}

}

}

}

**Bool isDepartmentValid()**

//This function checks that Department column is present in the excel file .

{

for( file : arr)

{

for(int j=0; j<=file.getPhysicalNumberofColumns; j++)

{

colName=”file.getColumn(j)”;

if(colName==Department)

{

Return true;

}

Else{

Return false;

}

}

}

}

**Bool isDesignationValid()**

//This function checks that the Designation column is present in the excel file .

{

for( file : arr)

{

for(int j=0; j<=file.getPhysicalNumberofColumns; j++)

{

colName=”file.getColumn(j)”;

if(colName==Designation)

{

Return true;

}

Else

{

Return false;

}

}

}

}