

Project Report

on

Parsing of input to generate

ENTITY RELATIONSHIP DIAGRAM

Submitted in partial fulfillment of the requirement

for the degree of

Bachelor of Technology (B.Tech)

In

Computer Science & Engineering



Guided By :-

- **Ms. Anjali Kapoor**

Assistant Professor

Department of CSE & IT

- **Ms. Aakanshi Gupta**

Submitted By :-

Karandeep Singh (01510402715)

Tulika (35110407216)

Ritika Chauhan (35410402715)

Ashwin Balaji (00710402715)

Amity School of Engineering and Technology
580, Delhi-Palam Vihar Road, Bijwasan, New Delhi-110061
[Affiliated to Guru Gobind Singh Indraprastha University, Delhi]
(June-July,2017)

CERTIFICATE

This is to certify that the Project Report entitled “PARSING OF INPUT TO GENERATE ENTITY RELATIONSHIP DIAGRAM “ comprehends the authentic work of In-house summer training accomplished by the team of listed students.

Name	Enrollment number
1. Karandeep Singh	01510402715
2. Tulika	35110407216
3. Ritika Chauhan	35410402715
4. Ashwin Balaji	00710402715

This training partially fulfills their Bachelor of Technology course requirement after second year. This project is completed at Amity School of Engineering & Technology, an institution affiliated to Guru Gobind Singh Indraprastha University , Delhi.

Ms.Anjali Kapoor

Ms.Aakanshi Gupta

ACKNOWLEDGEMENT

Before we get into thick things, we would like to add a few words of appreciation for people who have been part of this project right from its inception. It is to them we owe our deepest gratitude.

It gives us immense pleasure in presenting this project report on “ENTITY RELATIONSHIP DIAGRAM”. The success of this project is the result of sheer hard work and determination put in by our team under the guidance of our mentors. I hereby take this opportunity to add a special note of thanks for **Late. Prof. B.P.Singh** , Senior Director, **Prof. (Dr.) Rekha Aggarwal** , **Prof. M.N.Gupta** , HOD, Dept of CSE/IT , for their constant encouragement and guidance. Without their insights and support this project wouldn't have kicked started and neither would have reached fruitfulness.

We extend our gratitude and would like to express our profound thanks to **Ms. Anjali Kapoor** , Asst. Professor , Dept of CSE/IT , **Ms. Aakanshi Gupta** and **Mr. Amrit Nath Thulal** for their timely suggestion, valuable guidance , encouragement and help for completing this work. We would also like to thank our lab assistants for their support in providing all the softwares which were needed throughout this project. At the end, we would like to express our sincere thanks to all others who helped us directly or indirectly during this project work.

Karandeep Singh

(01510402715)

Tulika

(35110407216)

Ritika Chauhan

(35410402715)

Ashwin Balaji

(00710402715)

ABSTRACT

mappER is a windows application which enables user to input a string of data , which is in a preset format , which further gets parsed to generate Entity Relationship Diagram which is .png image file.

It uses **Python IDLE** for interpreting the input string entered by user. It further employs **GraphViz** application module to convert parsed input into .png file. Also, **Tkinter** python module is used for designing the front end of the application.

OBJECTIVE

The aim of **mappER** is to provide with the great convenience for drawing ER Diagrams which further help in designing good databases. Further **mappER** can be put to use for comparing different sorts of ER Diagrams which can be generated for a single situation or an organization which otherwise requires a lot of time, knowledge, labor & experience, if done manually. further provides a basic structure to visualize database.

It further provides a basic structure to visualize database. It is easy to understand for the users. By providing more relevant information by the user a better ER diagram will be plot,

TABLE OF CONTENTS

Certificate	i
Acknowledgement	ii
Abstract	iii
Objective	iv
1. INTRODUCTION	1
1.1 Purpose	1
1.2 System Overview	1
1.3.Problem Statement	1
1.4.DFD of the project	2
1.5.Goal & Vision	2
2. REQUIREMENTS SPECIFICATION	3
2.1.User Characteristic	3
2.2. Dependencies	3
2.3.Hardware Requirements	4
2.4.Constraints & Assumptions	5
3.STRATEGY	6
3.1. Activity List	6
3.2.Gantt Chart	7
3.3. Pert Chart	8
4.OUTLOOK	9
5. TESTING	10
FUTURE WORK	17
SUMMARY	18
REFERENCE	19
APPENDIX	20

LIST OF FIGURES

Figure 1.1- DATA FLOW DIAGRAM

Figure 3.1- Activity List

Figure 3.2- GANTT Chart

Figure 3.3- PERT Chart

Figure 4.1- GUI Outlook

Figure 4.2-OUTPUT

Figure 4.3-HOSPITAL(Example of GUI)

Figure 4.4-SCHOOL(Example of GUI)

Figure 5.1- Input Testing

Figure 5.2- Test Result

Chapter 1

INTRODUCTION

This section gives a scope description and the overview of everything included in this Project Report. Also, the purpose for this document is described and system overview along with goal and vision are listed.

1.1 PURPOSE

The purpose of this document is to give a detailed description of **mappER** Project. It will illustrate the purpose and complete declaration for the development of system. This document is primarily intended to anyone who wants to get an overview of how mappER works, its outcomes and possible usages in future.

1.2 SYSTEM OVERVIEW

mappER takes as input a textual string and generates Entity Relationship Diagram in .png file format. This process is done by employing **Python IDLE** which helps in taking strings as inputs and parsing them. Also , the conversion of the parsed string to .png file is employed by **GraphViz module**.

1.3 PROBLEM STATEMENT

Whenever we want to design a database for any organization or for any procedure , the first thing require is the ER-Diagram to get the basic idea of database design. So , basically ER-Diagram helps in visualizing databases. Till today, we have to spend a lot of time and energy on designing ER-Diagrams. **mappER** makes this easy by designing ER-Diagrams for them by taking inputs.

1.4 DFD OF THE PROJECT

As apparent from the DFD of the project :

- Parsing of input

Textual string entered by the user is parsed using PYTHON IDLE

- Generating .png image file

The parsed string is then converted into .png file containing ER-Diagram using **GraphViz**.

▪ Graphic User Interface

This is an user interface which takes inputs from the user shows the output, with an interactive output.

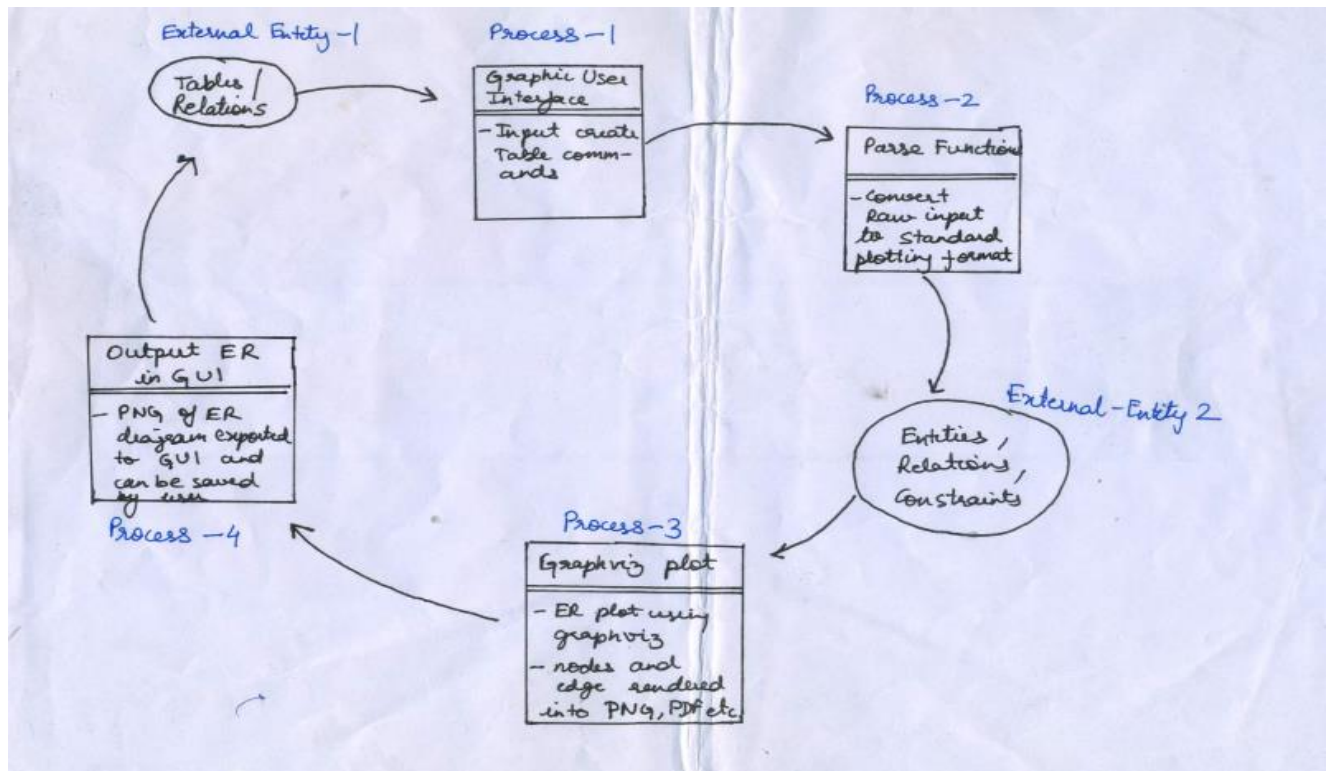


Fig-1.1 DFD

1.5 GOALS AND VISION

Goal of **mappER** is to provide with the great convenience for drawing ER Diagrams which further help in designing good databases.

Further **mappER** can be put to use for comparing different sorts of ER Diagrams which can be generated for a single situation or an organization which otherwise requires a lot of time, knowledge, labor & experience, if done manually.

Chapter 2

REQUIREMENT SPECIFICATIONS

2.1 USER CHARACTERISTICS

There is only one particular type of user that interact effectively with this system

- ◆ Users who wish to generate ER Diagrams and possess sufficient and sound knowledge of databases.

The users interact with the system through a user cordial Graphical User Interface. The GUI of this application presents a form for user to be filled in by the user. The form contains three fields and three buttons. The format for input of data in these fields is already provided on the GUI. On clicking the “Generate ER Diagram” button the user gets the desired ER Diagram on the screen. While if the user clicks on “Abort” button the data in the fields gets deleted and the user can enter data again. There is special “Help” button present on GUI containing all the information about the various keyword to be used while entering the data in the fields.

2.2 DEPENDENCIES

mappER is a Windows application which comprises of a frontend and a backend. Thses are designed using softwares described below:-

For backend

➤ **GraphViz**

GraphViz is an open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important application networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.

- **Features**

GraphViz was specifically customised to plot ER models from dynamic inputs. GraphViz layout programme takes description of graphs in simple text language and makes diagrams in formalike PDF,

images, SVG for browsers etc. GraphViz also has great many features of colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes.

The important role played by GraphViz in backend coding is that it facilitates in converting parsed string input into .png image file.

➤ **Python IDLE**

IDLE stands for Integrated Development Learning Environment is an integrated development environment for Python which has been bundled with the default implementation of the language.

- **Features**

- Multi-window text editor with syntax highlighting, auto completion, smart indent etc.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persistent breakpoints, and call stack visibility.

For Frontend

➤ **Tkinter**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI.

Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task.

2.3 HARDWARE REQUIREMENT

To access this application, the first and foremost need is of a PC/Laptop with at least 32 bit Windows. Then the other requirements are -:

- Python 3.6.1 should already be installed.
- Graphviz application should already be imported.

2.4 CONSTRAINTS

mappER works only for “English” language & will not work with any other language.

It assumes that the user is familiar with the concepts of databases and will enter only the relevant and accurate data.

The user must remain very careful during the input of data in the fields because if not entered in the pre-set format the ER Diagram may not be generated.

Chapter 3

STRATEGY

3.1 ACTIVITY LIST

	A	B	C	D
1	Task Name	Start Date	Duration	End Date
2	Project Selection	12-Jun-17	2	13-Jun-17
3	Project Name Selection & Strategising Events	13-Jun-17	1	13-Jun-17
4	Learning Python	13-Jun-17	6	18-Jun-17
5	Installation Graphviz	14-Jun-17	1	14-Jun-17
6	Phase-3 Working on Graphviz	15-Jun-17	9	23-Jun-17
7	Phase-2 Working on user input Ver-2	19-Jun-17	4	22-Jun-17
8	Phase-2 User input Ver-2(Parsing)	23-Jun-17	4	26-Jun-17
9	Learning GUI	22-Jun-17	6	27-Jun-17
10	Phase-2 Merging user input and Pycharm code	26-Jun-17	1	26-Jun-17
11	Working on GUI	27-Jun-17	7	3-Jul-17
12	Presentation	29-Jun-17	4	2-Jul-17

Fig 3.1- Activity list

3.2 GANTT CHART



Fig. 3.2- Gantt Chart

3.3 PERT CHART

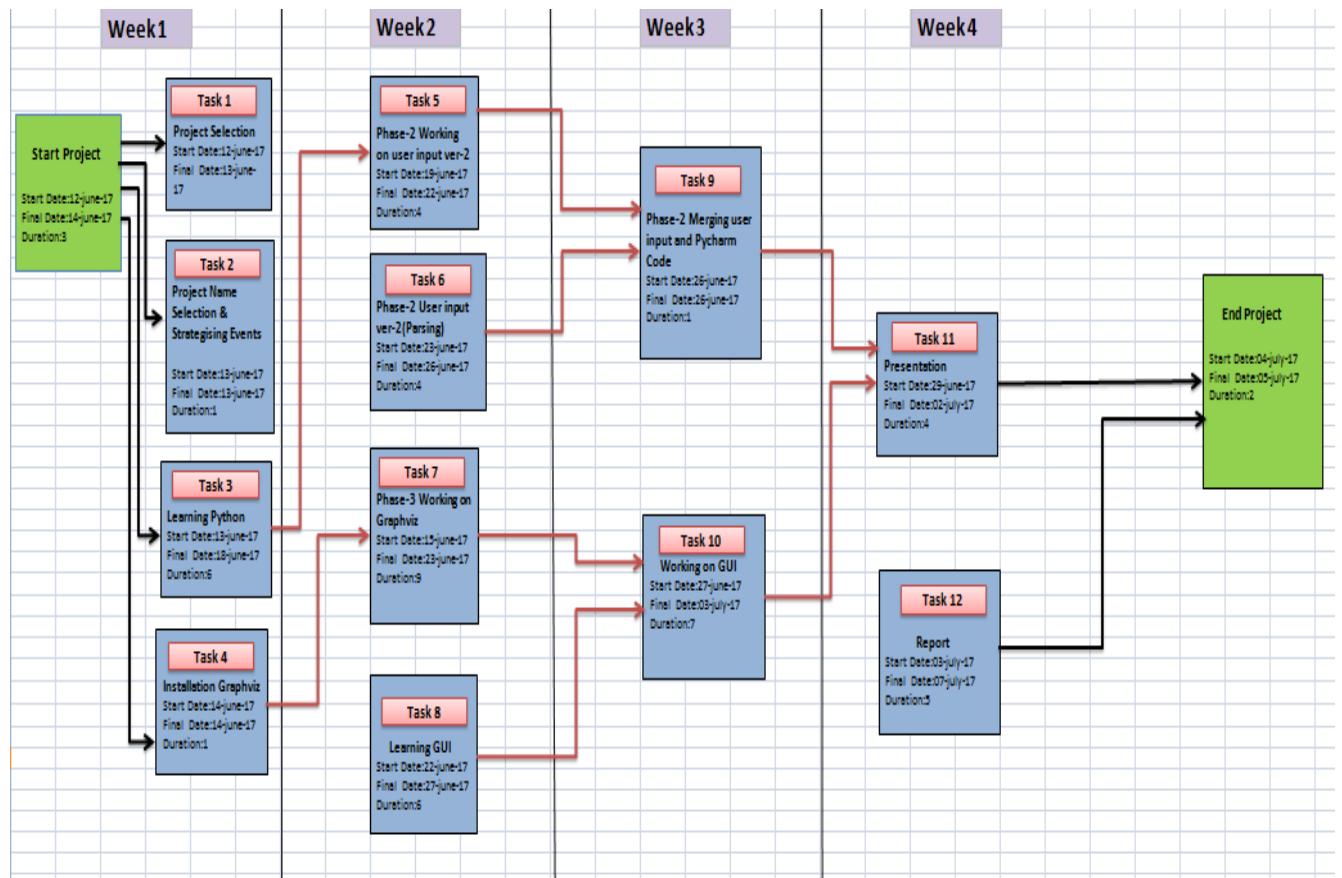


Fig. 3.3- Pert Chart

Chapter 4

4.1 OUTLOOK

The screenshot shows the mappER application window titled "mappER --->> Entity Relationship Model". The interface has a light orange background and contains the following elements:

- mappER** logo at the top center.
- Input prompt: "Enter the relationships and their types in the given format: Relationship1[Type]Relationship2[Type]....." with an empty text box below it.
- Input prompt: "Enter in the given format: Entity1[Relationship1,Relationship2..]Entity2[Relationship1,Relationship2..]..." with an empty text box below it.
- Input prompt: "Enter the entities and cardinalities wrt to the relationships : Entity1[Cardinality1,Cardinality2...]Entity2[Cardinality1,Cardinality2..].." with an empty text box below it.
- Input prompt: "Enter the total number of Entities:" with an empty text box below it.
- Input prompt: "Enter in the given format: Entity/Type[Attribute1(KeyConstraint1-AttributeConstraint2)..]..." with a large empty text box below it.
- Two buttons at the bottom: "Generate ER-Diagram" (purple) and "Abort" (red).

The Windows taskbar at the bottom shows the system clock as 00:33 on 07-Jul-17.

Fig-4.1 GUI OUTLOOK

4.2 OUTPUT

The screenshot shows the mappER application window titled "mappER --->> Entity Relationship Model (Not Responding)". The interface has a light orange background and contains the following elements:

- mappER** logo at the top center.
- Input prompt: "Enter the relationships and their types in the given format: Relationship1[Type]Relationship2[Type]....." with the text `teacher[identify]student[oc]` entered.
- Input prompt: "Enter in the given format: Entity1[Relationship1,Relationship2..]Entity2[Relationship1,Relationship2..]..." with the text `teacher[teaches,student]sta` entered.
- Input prompt: "Enter the entities and cardinalities wrt to the relationships : Entity1[Cardinality1,Cardinality2...]Entity2[Cardinality1,Cardinality2..].." with the text `teacher[1,N]student[N]` entered.
- Input prompt: "Enter the total number of Entities:" with the text `2` entered.
- Input prompt: "Enter in the given format: Entity/Type[Attribute1(KeyConstraint1-AttributeConstraint2)..]..." with the text `teacher S[sid(P-N),tname(N-M)]` and `student S[sid(P-N),tname(N-M)]` entered.
- Two buttons at the bottom: "Generate ER-Diagram" (purple) and "Abort" (red).

4.2 EXAMPLES OF GUI

1. HOSPITAL

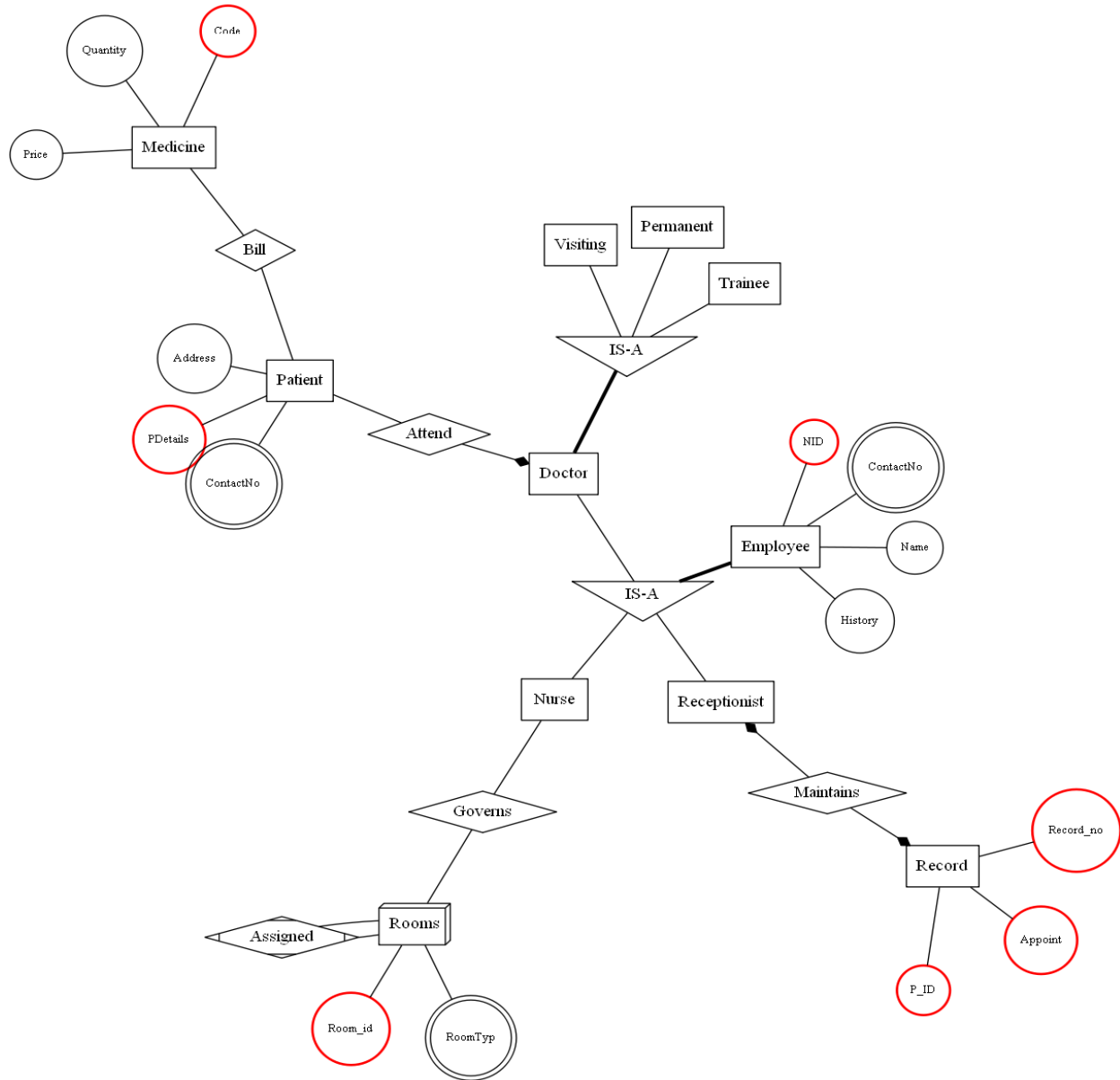


Fig 4.3-HOSPITAL(Example of GUI)

2. SCHOOL

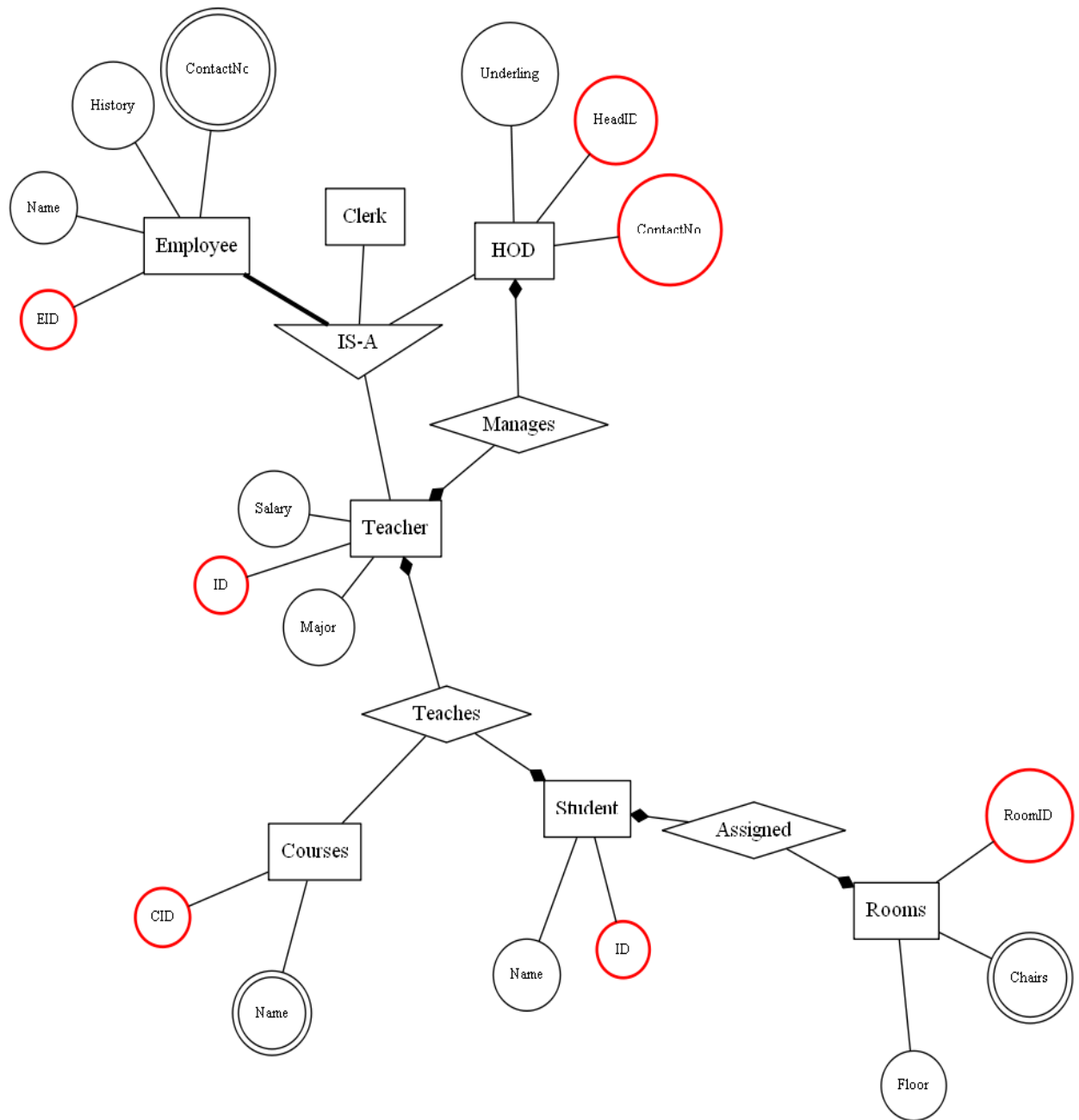


Fig 4.3-SCHOOL(Example of GUI)

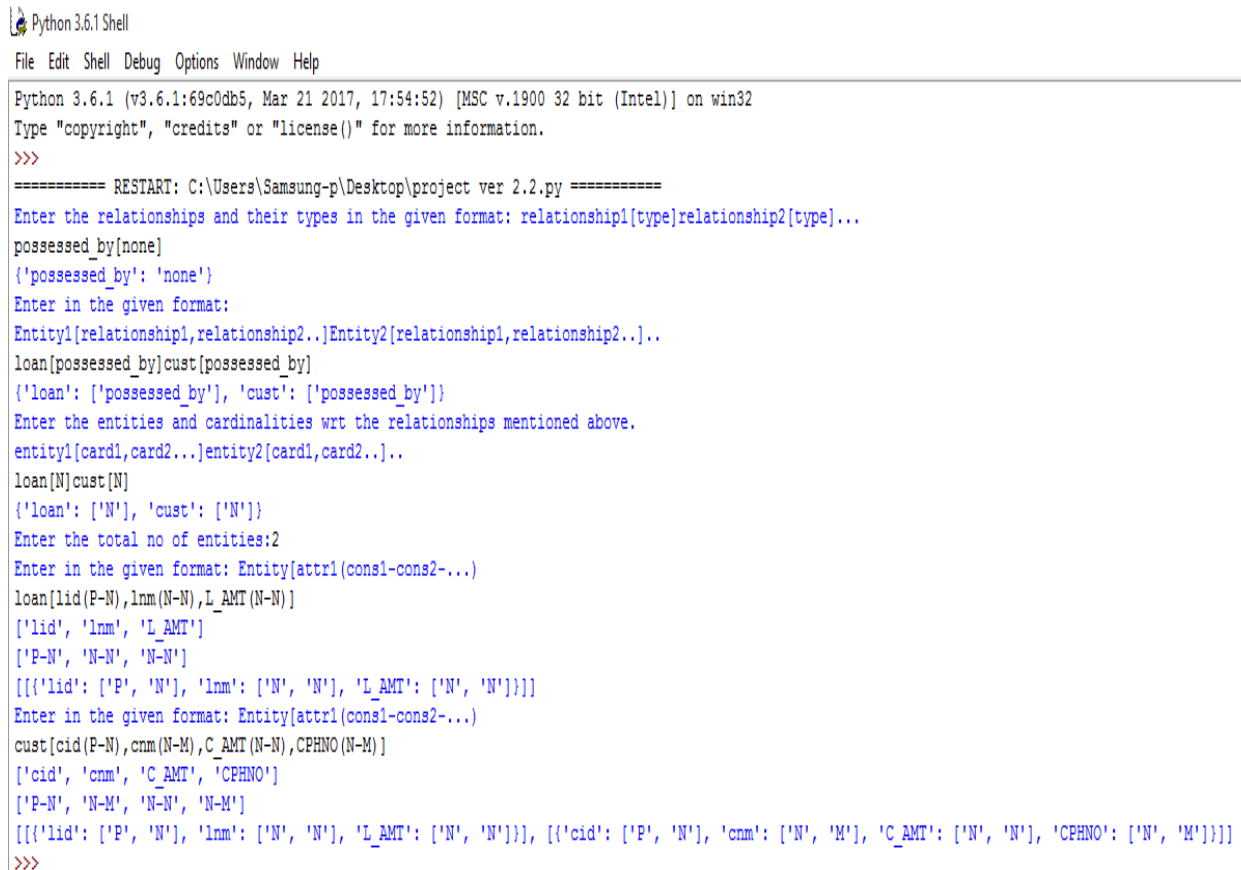
Chapter 5

TESTING

5.1 TEST PLAN

Unit Testing: Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually.

5.1.1 INPUT TESTING



```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Samsung-p\Desktop\project ver 2.2.py =====
Enter the relationships and their types in the given format: relationship1[type]relationship2[type]...
possessed_by[none]
{'possessed_by': 'none'}
Enter in the given format:
Entity1[relationship1,relationship2..]Entity2[relationship1,relationship2..]..
loan[possessed_by]cust[possessed_by]
{'loan': ['possessed_by'], 'cust': ['possessed_by']}
Enter the entities and cardinalities wrt the relationships mentioned above.
entity1[card1,card2...]entity2[card1,card2..]..
loan[N]cust[N]
{'loan': ['N'], 'cust': ['N']}
Enter the total no of entities:2
Enter in the given format: Entity[attr1(cons1-cons2-...)]
loan[lid(P-N),lnm(N-N),L_AMT(N-N)]
['lid', 'lnm', 'L_AMT']
['P-N', 'N-N', 'N-N']
[[{'lid': ['P', 'N'], 'lnm': ['N', 'N'], 'L_AMT': ['N', 'N']}]
Enter in the given format: Entity[attr1(cons1-cons2-...)]
cust[cid(P-N),cnm(N-M),C_AMT(N-N),CPHNO(N-M)]
['cid', 'cnm', 'C_AMT', 'CPHNO']
['P-N', 'N-M', 'N-N', 'N-M']
[[{'lid': ['P', 'N'], 'lnm': ['N', 'N'], 'L_AMT': ['N', 'N']}, {'cid': ['P', 'N'], 'cnm': ['N', 'M'], 'C_AMT': ['N', 'N'], 'CPHNO': ['N', 'M']}]
>>>
```

Fig-5.1.1-Input Testing

5.1.2 TEST RESULT

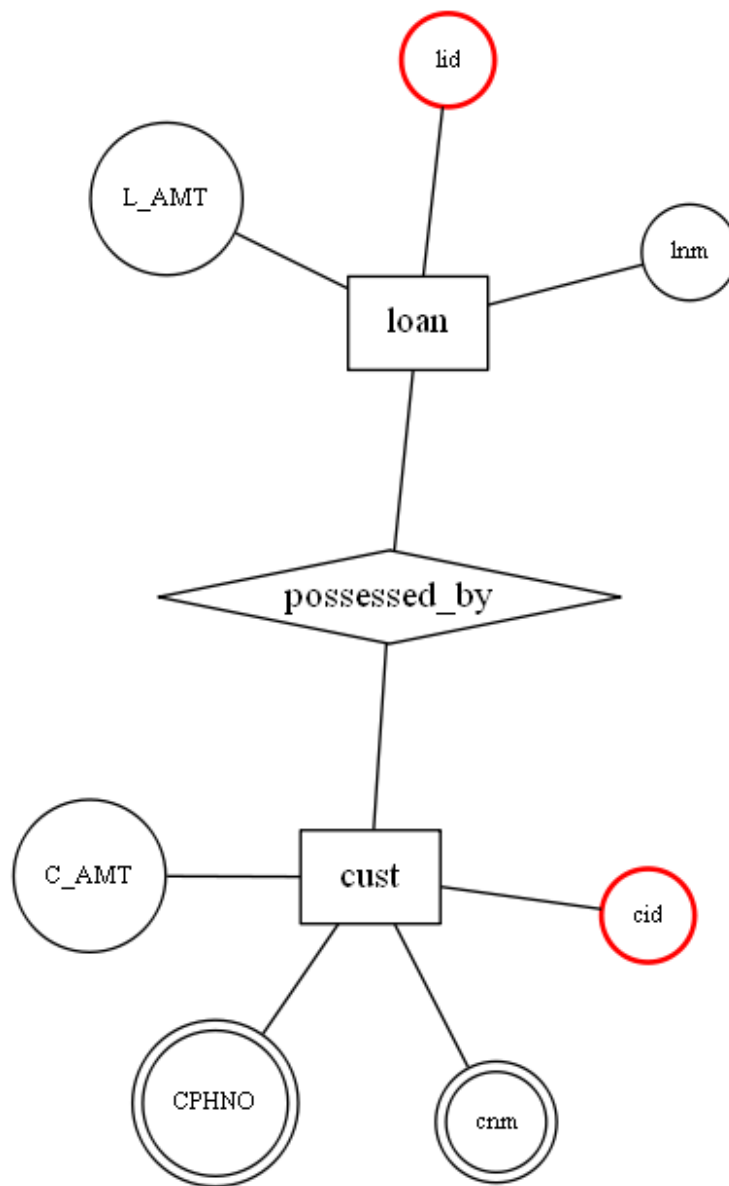


Fig 5.1.2-Test Result

FUTURE WORK

Some of the possible amendments and improvements in this system are:

1. More complex ER diagrams.
2. Fixing errors in current GUI for the application.
3. Minimize inputs to plot ER diagrams.

SUMMARY

Application like mappER are used for text-to-image converter for the future use. It is a database model. Its aim is to create a ER Diagram in .png file format according to the relevant information which is filled by the user. It further provides a basic structure to visualize database. It is easy to understand for the users. By providing more relevant information by the user a better ER diagram will be plot,

REFERENCES

- <http://www.graphviz.org/Home.php>
- <https://pythonprogramming.net>
- <https://www.tutorialspoint.com/pyqt/>
- <https://docs.python.org/3/library/tkinter.html/>
- <https://stackoverflow.com/>

APPENDIX

CODING

➤ INPUT CODE

```
temp=[]
print('Enter the relationships and their types in the given format:
relationship1[type]relationship2[type]...')
string3=input()
replacements3=('[' ,']')
for l in replacements3:
    string3=string3.replace(l,' ')
temp=string3.split()
L14=[]
L15=[]
for u in range(0,len(temp),2):
    L14.append(temp[u])
for y in range(1,len(temp),2):
    L15.append(temp[y])
listREL={ }
for k in range(len(L14)):
    d3[L14[k]]=L15[k]
print(listREL)

temp1=[]
print('Enter in the given format:')
print('Entity1[relationship1,relationship2..]Entity2[relationship1,relationship2..]..')
string=input()
replacement=('[' ,']')
for i in replacement:
    string=string.replace(i,' ')
temp1=string.split()
L0=[]#list of entities
L1=[]#list of all other things of temp1
for q in range(0,len(temp1),2):
    L0.append(temp1[q])
for w in range(1,len(temp1),2):
    L1.append(temp1[w])
listEN={ }
for e in range(len(L0)):
    listEN[L0[e]]=L1[e].split(',')
print(listEN)
```



```

temp2=[]
print('Enter the entities and cardinalities wrt the relationships mentioned above.')
print('entity1[card1,card2...]entity2[card1,card2..].')
string1=input()
replacements=('[' ,']')
for r in replacements:
    string1=string1.replace(r,' ')
temp2=string1.split()
L2=[]
L3=[]
for t in range(0,len(temp2),2):
    L2.append(temp2[t])
for o in range(1,len(temp2),2):
    L3.append(temp2[o])
listCAR={}
for p in range(len(L2)):
    listCAR[L2[p]]=L3[p].split(',')
print(listCAR)

listAT=[]
n=int(input("Enter the total no of entities:"))
for x in range(n):
    L=[]
    print('Enter in the given format: Entity[attr1(cons1-cons2-...)\n')
    string=input()
    replacements=('[' ,'( ',' )',',')
    for r in replacements:
        string = string.replace(r,' ')
    L=string.split()
    L4=[]
    L5=[]
    L6=[]
    L7=[]
    L4.append(L[0])
    L5.append(L[1])
    for v in range(2,len(L),2):
        L6.append(L[v])
    for i in range(3,len(L),2):
        L7.append(L[i])
    d={}
    L8=[]
    for y in range(len(L3)):
        d[L6[y]]=L7[y].split('-')
    L8.append(d)
    listAT.append(L8)
print(listAT)

```

➤ GRAPHVIZ CODE

```
listEN = EntRel
listREL = RelRel
listCAR = EntCar
listAT = Atr

#function define and render
from graphviz import *
import os
os.environ["PATH"] += os.pathsep + 'C:/Graphviz2.38/bin/'

def relation(x,r):
    for key,value in r.items():
        if value=='identify':
            type='Mdiamond'
        else:
            type='diamond'
        x.node(key,shape=type)

def entity(x,l,c,rel):

    for key,value in l.items():
        count = 0 # number of classifications
        if 'W' in c[key]:
            sh='box3d'
        else:

            sh='rect'
            x.node(key, shape=sh)
            for i in range(0,len(value)):

                if c[key][i]=='N':    #cardinality is many
                    point='none'
                else:
                    point='diamond'
                if 'W' in c[key] and rel[value[i]]=='identify':    #if entity is weak , hence
                    cardinality is always one-many
                    point='none'
```

```

        x.edge(value[i],key, len='1.50', dir='both', arrowhead='none',
arrowtail=point)
        if value[i] in list(l):

            if count==0:
                new=key+'cls'
                x.node(new,label='IS-A',shape='invtriangle')
                x.attr('node',shape='rect')
                x.edge(key,new,dir='none',penwidth='3')
                x.edge(value[i],new,dir='none')
                count=1
            else:
                x.edge(value[i], new, dir='none')
        else:
            x.edge(key,value[i],dir='both',arrowhead='none',arrowtail=point)

```

```

def attribute(x,e,a):

```

```

    temp=list(e)

```

```

    for i in range(0,len(a)):
        for j in range(0,len(a[i])):
            if len(a[i][j])!=0:
                id=0
                for atr,ctr in a[i][j].items():
                    sty = 'solid' #style
                    shp = 'circle' #shape
                    #Keys
                    if ctr[0]=='P':
                        col='red'
                        wid = '2'
                    elif ctr[0]=='F':
                        col='blue'
                        wid = '2'
                    else:
                        col='black'
                        wid = '1'
                    #AttributeType
                    if ctr[1]=='D':
                        sty = 'dashed'
                    elif ctr[1]=='M':
                        shp = 'doublecircle'
                    foo=str(temp[i])+str(id) #temp id variable
                    # print(foo)

```

```

x.node(foo,label=str(atr),color=col,penwidth=wid,shape=shp,style=sty,fontsize='10'
)
        x.edge(temp[i],foo,dir='none')
        id+=1

check=0 #check Cardinality and Entity list: number of relation must match
cardinalities given
for keyE in listEN.keys():

    if len(listEN[keyE]) != len(listCAR[keyE]):
        print("Relations-Cardinality Mismatch for - ",keyE)
        check=1
if check==0:
    page = Digraph(name='Model', engine='neato')
    relation(page, listREL)
    entity(page,listEN,listCAR,listREL)
    attribute(page, listEN, listAT)
    page.edge_attr.update(len='1.3')
    page.format = 'png'
    page.render('test-output/Final.gv', view=True)

```

➤ GRAPHICAL USER INTERFACE CODE(GUI)

```

from tkinter import *
from graphviz import *
import os
os.environ["PATH"] += os.pathsep + 'C:/Graphviz2.38/bin/'
root=Tk()

#=====BASIC_ELEMENTS_
OF_THIS_PROGRAM=====
=====

variable=StringVar()
variable1=StringVar()
variable2=StringVar()
variable3=StringVar()
variable4=StringVar()


top=Frame(root)
bottom=Frame(root)

top.pack(side=TOP)

```

```

bottom.pack(side=BOTTOM)

root.title("mappER --->> Entity Relationship Model")

variable=StringVar()

#=====BASIC_ELEMENTS_
OF_THIS_PROGRAM_END=====
=====

#=====SCREEN_SIZE_DEFI
NITION=====
=====

root.geometry("1000x600+0+0")
# ( width x height + x +y )

#=====SCREEN_SIZE_DEFI
NITION_ENDS=====
=====

Screen_bg = Frame(root, relief='flat',bg="moccasin")
Screen_bg.place(height=720, width=1370, x=0, y=0) # ( " order should be like this " )

# [ no pack() required because we need this code to be implemented till end ]

#=====TITLE_CONFIGURA
TION=====
=====

Title_label=Label(root,text='mappER',font=("Arial Rounded MT Bold",40,"normal"),
    relief='flat',fg='midnight blue',bg='moccasin').pack(side=TOP)

#=====TITLE_CONFIGURA
TION_END=====
=====

label1 = Label(root,text=' Enter the relationships and their types in the given format:
Relationship1[Type]Relationship2[Type]..... ',
    width=100,height=2,font=('ariel round MT bold',12,'bold'),
    bg='moccasin',relief='flat',fg='midnight blue')

entry_rel_type = Entry(root,textvariable=variable1,font=('times new
roman',12,'normal'),bg='antique white',fg='black',
    justify='left',relief='sunken')

```

```

label2 = Label(root,text='Enter in the given format:
Entity1[Relationship1,Relationship2..]Entity2[Relationship1,Relationship2..]....',
               width=100,height=2,font=('ariel round MT bold',12,'bold'),
               bg='moccasin',relief='flat',fg='midnight blue')

entry_ent_rel = Entry(root,textvariable=variable2,font=('times new
roman',12,'normal'),bg='antique white',fg='black',
                    justify='left',relief='sunken')

label3 = Label(root,text='Enter the entities and cardinalities wrt to the relationships :
Entity1[Cardinality1,Cardinality2...]Entity2[Cardinality1,Cardinality2..]..',
               width=150,height=2,font=('ariel round MT bold',12,'bold'),
               bg='moccasin',relief='flat',fg='midnight blue')

entry_ent_card = Entry(root,textvariable=variable3,font=('times new
roman',12,'normal'),bg='antique white',fg='black',
                    justify='left',relief='sunken')

label4 = Label(root,text='Enter the total number of Entities:',
               width=100,height=2,font=('ariel round MT bold',12,'bold'),
               bg='moccasin',relief='flat',fg='midnight blue')

entry_tot_ent = Entry(root,textvariable=variable,font=('times new
roman',12,'normal'),bg='antique white',fg='black',
                    justify='left',relief='sunken')

label5 = Label(root,text=('Enter in the given format: Entity/Type[Attribute1(KeyConstraint1-
AttributeConstraint2)...]...'),
               width=100,height=2,font=('ariel round MT bold',12,'bold'),
               bg='moccasin',relief='flat',fg='midnight blue')

#entry_ent_attr = Text(root,width=70,height=9,font=('times new roman',12,'normal'),bg='antique
white',fg='black',relief='sunken')

entry_ent_attr = Entry(root,textvariable=variable4,font=('times new
roman',12,'normal'),bg='antique white',fg='black',
                    justify='left',relief='sunken')

def DATA():
    temp=[]
    string3=entry_rel_type.get()

```

```

replacements3=('[' ,']')
for l in replacements3:
    string3=string3.replace(l, ' ')
temp=string3.split()
L14=[]
L15=[]
for u in range(0,len(temp),2):
    L14.append(temp[u])
for y in range(1,len(temp),2):
    L15.append(temp[y])
listREL={}
for k in range(len(L14)):
    listREL[L14[k]]=L15[k]
print(listREL)

```

```

temp1=[]
string=entry_ent_rel.get()
replacement=('[' ,']')
for i in replacement:
    string=string.replace(i, ' ')
temp1=string.split()
L0=[]#list of entities
L1=[]#list of all other things of temp1
for q in range(0,len(temp1),2):
    L0.append(temp1[q])
for w in range(1,len(temp1),2):
    L1.append(temp1[w])
listEN={}
for e in range(len(L0)):
    listEN[L0[e]]=L1[e].split(',')
print(listEN)

```

```

temp2=[]
string1=entry_ent_card.get()
replacements=('[' ,']')
for r in replacements:
    string1=string1.replace(r, ' ')
temp2=string1.split()
L2=[]
L3=[]
for t in range(0,len(temp2),2):
    L2.append(temp2[t])
for o in range(1,len(temp2),2):
    L3.append(temp2[o])
listCAR={}
for p in range(len(L2)):

```

```

        listCAR[L2[p]]=L3[p].split(',')
    print(listCAR)
    exc()
def exc():
    listAT=[]
    n=input(entry_tot_ent.get())
    for x in range(n):
        L=[]
        string=entry_ent_attr.get()
        replacements=('/', '[' , '(' , ')', ',', ';')
        for r in replacements:
            string = string.replace(r, ' ')
        L=string.split()
        L4=[]
        L5=[]
        L6=[]
        L7=[]
        L4.append(L[0])
        L5.append(L[1])
        for v in range(2,len(L),2):
            L6.append(L[v])
        for i in range(3,len(L),2):
            L7.append(L[i])
        d={}
        d1={}
        L8=[]
        for y in range(len(L3)):
            d[L6[y]]=L7[y].split('-')
        L8.append(d)
        listAT.append(L8)
        d1.append(listAT)
        #entry_ent_attr.insert(listAT,' ')

#=====EXIT_FUNCTION=====
=====

def Exit_Program():
    root.destroy()

#Cancel_label=Label(root,textvariable=variable)
cancel_button=Button(root,text="Abort",font=("Times new roman",15,"bold"),

height=1,width=30,bd=4,fg='red2',bg="burlywood",relief='raised',command=Exit_Program)

```



```
#=====EXIT_FUNCTION_
ENDS=====
=====
```

```
#=====GRAPHVIZ_FUNC
TION=====
=====
```

```
def GRD_Program():
    DATA()
    pbar=Progressbar(root,mode='indeterminate')
    def relation(x,r):
        for key,value in r.items():
            if value=='identify':
                type='Mdiamond'
            else:
                type='diamond'
            x.node(key,shape=type)
    def entity(x,l,c,rel):

        for key,value in l.items():
            count = 0 # number of classifications
            if 'W' in c[key]:
                sh='box3d'
            else:
                sh='rect'
            x.node(key, shape=sh)
            for i in range(0,len(value)):

                if c[key][i]=='N':    #cardinality is many
                    point='none'
                else:
                    point='diamond'
                if 'W' in c[key] and rel[value[i]]=='identify':    #if entity is weak , hence cardinality is
always one-many
                    point='none'
                    x.edge(value[i],key, len='1.50', dir='both', arrowhead='none', arrowtail=point)
            if value[i] in list(l):
                if count==0:
                    new=key+'cls'
                    x.node(new,label='IS-A',shape='invtriangle')
                    x.attr('node',shape='rect')
                    x.edge(key,new,dir='none',penwidth='3')
                    x.edge(value[i],new,dir='none')
                    count=1
```

```

        else:
            x.edge(value[i], new, dir='none')
        else:
            x.edge(key,value[i],dir='both',arrowhead='none',arrowtail=point)

def attribute(x,e,a):
    temp=list(e)

    for i in range(0,len(a)):
        for j in range(0,len(a[i])):
            if len(a[i][j])!=0:
                id=0
                for atr,ctr in a[i][j].items():
                    sty = 'solid' #style
                    shp = 'circle' #shape
                    #Keys
                    if ctr[0]=='P':
                        col='red'
                        wid = '2'
                    elif ctr[0]=='F':
                        col='blue'
                        wid = '2'
                    else:
                        col='black'
                        wid = '1'
                    #AttributeType
                    if ctr[1]=='D':
                        sty = 'dashed'
                    elif ctr[1]=='M':
                        shp = 'doublecircle'
                    foo=str(temp[i])+str(id) #temp id variable
                    # print(foo)

x.node(foo,label=str(atr),color=col,penwidth=wid,shape=shp,style=sty,fontsize='10')
    x.edge(temp[i],foo,dir='none')
    id+=1

#Test Lists
'''
listREL = {'Assigned':'none','Manages':'none','Teaches':'none'}
listEN =
{'Employee':['Teacher','Clerk','HOD'],'Student':['Teaches','Assigned'],'Teacher':['Teaches','Manag
es'],'HOD':['Manages'],
    'Courses':['Teaches'],'Rooms':['Assigned'],'Clerk':[]}
```

```

listCAR =
{'Employee':['N','N','N'],'Student':['1','1'],'HOD':['1'],'Courses':['N'],'Rooms':['1'],'Teacher':['1','1'],
'Clerk':[]} #1:one , N:many ,W:weak->always many-one
listAT = [
    [{'EID': ['P', 'N'],'History': ['N', 'N'],'Name':['N','N'],'ContactNo':['N','M']}],
    [{'Name':['N','N'],'ID':['P','N']}], [{'ID':['P','N'],'Salary':['N','N'],'Major':['N','N']}],

[{'HeadID':['P','N'],'Underling':['N','N'],'ContactNo':['P','N']}], [{'CID':['P','N'],'Name':['N','M']}],
    [{'RoomID':['P','N'],'Chairs':['N','M'],'Floor':['N','N']}]]
    ]
'''

check=0 #check Cardinality and Entity list: number of relation must match cardinalities given
for keyE in listEN.keys():

    if len(listEN[keyE]) != len(listCAR[keyE]):
        print("Relations-Cardinality Mismatch for - ",keyE)
        check=1

if check==0:
    page = Digraph(name='Model', engine='neato')
    relation(page, listREL)
    entity(page,listEN,listCAR,listREL)
    attribute(page, listEN, listAT)
    page.edge_attr.update(len='1.3')
    page.format = 'png'
    page.render('test-output/School.gv', view=True)

#GenerateERD_label=Label(root,textvariable=variable)
generateERD_button=Button(root,text="Generate ER-Diagram",font=("Times new
roman",15,"bold"),

height=1,width=30,bd=4,fg='maroon4',bg="burlywood",relief='raised',command=GRD_Program
)

#=====GRAPHVIZ_PROG
RAM_FINISH=====
=====

#strong button position,#button position(first cum first serve,different literals NO
EFFECT(TOP,LEFT)(TOP,RIGHT),(BOTTOM,LEFT)(BOTTOM,RIGHT),WORK[(LEFT,LE
FT)(RIGHT,RIGHT)])

generateERD_button.pack(in_=bottom,side=LEFT,pady=25)
cancel_button.pack(in_=bottom,side=LEFT,padx=25)

#=====

```

```
label1.pack(side=TOP)
entry_rel_type.pack(side=TOP)
label2.pack(side=TOP)
entry_ent_rel.pack(side=TOP)
label3.pack(side=TOP)
entry_ent_card.pack(side=TOP)
label4.pack(side=TOP)
entry_tot_ent.pack(side=TOP)
label5.pack(side=TOP)
entry_ent_attr.pack(side=TOP)
```

```
#=====ROOT_LOOP=====
=====
```

```
root.mainloop()
```

```
#you just can't use them both grid and pack on widgets that share the same parent.
```

