

**KENDRIYA VIDYALAYA NO.2**

**FCI CAMPUS, GORAKHPUR**

**CERTIFICATE**

This is certified to be the bonafide work of PRAVIN KUMAR SHUKLA of class XII has successfully completed the project on BUS TICKET BOOKING SYSTEM during academic session **2019-2020** and has been carried out under my direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form a part of any other course undergone by the candidate.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of Student Signature of Teacher

**Name: PRAVIN KR. SHUKLA** **Name: REETESH KUMAR**

**Designation: PGT(Comp.SC)**

**ACKNOWLEDGEMENT**

I would like to express my kind gesture of gratitude to my teacher **Mr. Reetesh Kumar** as well as our principal **Mr. R.K. Mall** who gave me the golden opportunity to do this wonderful project on the topic BUS TICKET BOOKING SYSTEM which also helped me in doing a lot of research and I came to know about so many new things, I am really thankful to them.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

PRAVIN KR. SHUKLA

CLASS: XII

CONTENTS

1. Introduction ----------------------------------------------------------------------------------------------------
2. Objective of the project--------------------------------------------------------------------------------
3. Theoretical background-------------------------------------------------------------------------------
4. Problem definition and analysis----------------------------------------------------------------
5. System Implementation--------------------------------------------------------------------------------
   1. Hardware Used--------------------------------------------------------------------------------
   2. Software Used----------------------------------------------------------------------------------
6. System design and development---------------------------------------------------------------
   1. Database design-------------------------------------------------------------------------------
   2. Flow Diagram-----------------------------------------------------------------------------------
   3. Program code------------------------------------------------------------------------------------
   4. Outputs----------------------------------------------------------------------------------------------
7. User Manual----------------------------------------------------------------------------------------------------
   1. How to install---------------------------------------------------------------------------------
   2. Working with software----------------------------------------------------------------
8. References------------------------------------------------------------------------------------------------------

**1.INTRODUCTION**

This software project is developed to automate the functionalities for users of particular bus ticket booking. This project Bus Ticket Booking system has been developed on Python, tkinter and MySQL. The main aim for developing this project is to provide details of buses, agent who are booking tickets for customer’s journey which give bus number and departure time of the bus. This system can manage bus details , tour details and customer’s details. It can also maintain the details of booking time of the seat(s) or collecting time of the tickets, the booking date and the name of agent which is optional, by which the customer can reserve the seats for his journey. The main objective of this project is to provide the better work efficiency, security, accuracy, reliability, feasibility.

During coding and design of the software project **Python IDLE 8.1** weight of powerful front and tool is used for getting graphical user interface through Tkinter integrated library and coding simplicity full stop as a back and powerful open-source RDBMS- **MySQL** is used as per requirement of CBSE curriculum of computer science course.

**2. OBJECTIVE AND SCOPE OF THE PROJECT**

The objective of software project is to automate the functionalities for users of particular bus ticket booking. The main objective of this project is to provide the better work efficiency, security, accuracy, reliability, feasibility. The proposed software system is expected to do the following functionality –

* To provide a user-friendly graphical user interface(GUI) based integrated and centralised environment for MIS activities.
* The proposed system should maintain all the records and transactions and should generate the required reports and information when required.
* To provide graphical and user-friendly interface to interact with the centralised database based on client- server architecture.
* To identify the critical operation procedure and possibilities of simplification using modern it tools and practices.

Its current scope, the software enables users to retrieve and update the information from satellite database design with MySQL. This software does not require much training time of users due to Limited function and simplicity during the development of sports shop in computer system project **Python IDLE 8.1** full open source event driven form based development environment is used for modular design and future expandability of the system.

Despite of the best of effort of the developer of the following limitations and functional boundaries are visible which limits the scope of the application software.

* The software can store records and produce reports in free design format in soft copy there is no facility at to produce customised reports on the specified reports are covered.
* Some application area like deleting the record.
* There is no provision to calculate fine or penalty adjective for defaulter member however it can be developed easily with the help of reading module.

**2.1 FUTURE SCOPE**

So far as future scope of the project is concerned firstly it is open to any module expansion that is other modular functions that can be design and embedded to handle the user needed in future any part of the soft and reports can be modified independently without much effort.

**3. Theoretical Background**

***What is database?***

***Introduction and Concepts:***

A database is a collection of information related to a particular subject all purpose, such as tracking customer orders on maintaining a music collection. Using any RDBMS application software like my SQL server MySQL, Oracle, Sybase, etc. you can imagine all your information from a single database file. Within the file, divide your data into separate storage containers called tables. You may and retrieve the data using queries.

Table is a collection of data about a specific topic such as products or supplies. Using a separate table for each topic means you can store that data only once, which makes a database more efficient and reduce data entry errors. Table organises data into columns and rows.

Primary key is one or more fields value or values uniquely identify each record in a table. In a relationship, a primary key is used to refer to specific record in one table from another table. A primary key is called foreign key when it is referred to form another table.

To find and retrieve just data that meets condition you specify, including data from multiple tables, create a query. Query can also update or delete multiple records at same time and perform built in our custom calculations on a data.

***Role of RDBMS application program:***

A computer database works as electronic filing system, which has a large number of ways of crossing referencing and this allows the user many different ways in which to reorganize and retrieve data. Database can handle business inventory accounting and feeling and use the information in its files to prepare summaries, estimates and other reports the management of data in a data system is done by means of general-purpose software package called a database management system or DBMS full stop some commercially available DBMS are my SQL server MS Access integrate, Oracle. A database management system, therefore is a combination of hardware and software that can be used to setup and monitor the database, and can manage the updating and retrieval of database that has been stored in IT. Most of the database management system have the following capabilities:

* Creating of a table, addition, deletion, modification of records.
* Retrieving data collectively or selectively.
* The data stored can be stored on indexed at users’ discretion and direction.
* Various reports can be produced from the system. This may be standardized reports of that may be fit specifically generated according to specific user definition.
* Mathematical functions can be performed and the data stored in the database can be manipulated with these functions to perform the desired calculations.
* To maintain Data integrity and database use.

The DBMS interprets and process users request to retrieve information from a database. In the most cases a query request will have to penetrate several layers of software in DBMS and operating system before the physical database can be accessed the DBMS response to a query by invoking the appropriate subprograms, each of which performs its special functions to interpret the query or to locate the desired data in database and present it in the desired order for stock.

***What is MySql?***

The management of data in database system is done by means of general-purpose software package called a database management system for DBMS. Some commercially available RDBMS are MS SQL server, MS Access, Oracle and sybase.

My SQL, the most popular open-source SQL database management system, is developed, distribute, and supported by Oracle Corporation. My SQL is named after co-founder Monty Widenius's daughter, My. The name of MySQL Dolphin all logo is ‘Sakila'.

* *MySql is a database management system.*

Database is structured Collection of data. It may be anything from a simple shopping list to a picture gallery for the vast amounts of information in a corporate network. The add, access, and process data stored in a computer database, you need a database management system such as MySQL server.

Since computer are very good at handling large amount of data and database management system play a central role in computing, as standalone utilities, or as part of another application.

* *MySql is based on SQL.*

Relational database stores data in separate tables rather than putting all the data in one big store room. This at speed and flexibility. The SQL part of MySQL stands for structured query language. SQL is the most common standardized language used to access database and is defined as a n s / ISO SQL standard. The SQL standard has been evolving since 1986 in several versions. In this manual,”SQL-92" refers to the standard released in 1992,”SQL:1999” refers to the standard release in 1999 and “SQL:2003” refers to the current version of the standard.

* *MySql software is Open Source.*

Open source means that it is possible for anyone to use and modify the software. And you buddy can download the my SQL software from the internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. MySQL software use the GPL (GNU general public licence).

* *The MySql Database Server is very fast, reliable, and easy to use.*

If that is what you are looking for, you should give it a try. MySQL server also has a practical set of features develop in close corporations with our users. You can find a performance comparison of my SQL server with other database manager on our benchmark page. MySQL server has originally developed to handle large database much faster than existing solutions and has been successful used in highly demanding production environments for several years. Although under development, MySQL server today offers rich and useful setup functions. Its connectivity, speed and security make MySql Server highly suited for accessing database on the Internet.

* *MySql Server works in client /server or embedded systems.*

The MySQL software database is a client server system that consists of multiple threaded SQL server that supposed different backgrounds, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces.

***What is Python IDLE ?***

IDLE is Python’s Integrated Development and Learning Environment.

IDLE has the following features:

* Coded in 100% pure Python, using the **[tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) GUI toolkit**
* Cross-platform: works mostly the same on Windows, Unix, and macOS.
* Python shell window (interactive interpreter) with colorizing of code input, output, and error messages.
* Multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features.
* Search within any window, replace within editor windows, and search through multiple files (grep).
* Debugger with persistent breakpoints, stepping, and viewing of global and local namespaces.
* Configuration, browsers, and other dialogs.

IDLE has been criticized for various usability issues, including losing focus, lack of copying to clipboard feature, lack of line numbering options, and general user interface design; it has been called a "disposable" IDE, because users frequently move on to a more advanced IDE as they gain experience.

Author **Guido van Rossum** says IDLE stands for **"Integrated Development Environment"** and since Van Rossum named the language Python partly to honor British comedy group Monty Python, the name IDLE was probably also chosen partly to honor Eric Idle, one of Monty Python's founding members.

**4. Problem Definition and Analysis :**

The hardest part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult establishing The detailed technical requirement. Definition and applying good, complete requirements are hard to work and success in the endeavour has eluded many of us. Yet, we can continue to make progress. Problem definition describes the watt of system, not how. The quality of software product is only as good as process that creates it. Problem definition is one of the most crucial steps in the creation process. Without define of problem, developers do not know what to build, customers do not know what to accept, and there is no way to validate that the bill system satisfies the requirement.

Problem definition and analysis is the activity that encompasses learning about the problem to be solved, understanding the needs of customer and users, trying to find out who the user really is, and understanding all the constraints on the solution. It includes all activities related to the following:

1. Identification and documentation of customers or users needs.
2. Creation of a document that describes the external behavior and dissociation constants that will satisfy those needs.
3. Analysis and valuation of the requirements documents to ensure consistency, completeness, visibility etc.
4. Evolution of needs.

After the analysis of the functioning of a bus ticket booking system, the proposed system is accepted to do-

1. To provide a user-friendly, graphical user interface based integrated and centralised environment for computerized bus ticket booking system.
2. The proposed system should maintain all the records and transactions, and should generate the required reports and information when required.
3. To provide efficient and Secure information storage.
4. To provide graphical and user-friendly interface to interact with the centralised database based on client server architecture.
5. To identify the critical operations procedure and possibilities of simplification using modern it tools and practices.

**5. System Implementation**

*5.1 Hardware used:*

While developing the system, the used hardware are:

* PC with p4 processor having 1GB RAM and other required devices.

5.2 *Software used:*

* Microsoft Windows 7 or above as operating system.
* Android with 1GB RAM as secondary OS.
* Python 7.1 IDLE.
* MySQL- Workbench 8.0 CE and Command Line 8.0.
* MS Word 2007 for documentation.
* Google Chrome for troubleshooting.

**6.Sytem Design and Development**

6.1 *The Database Design:*

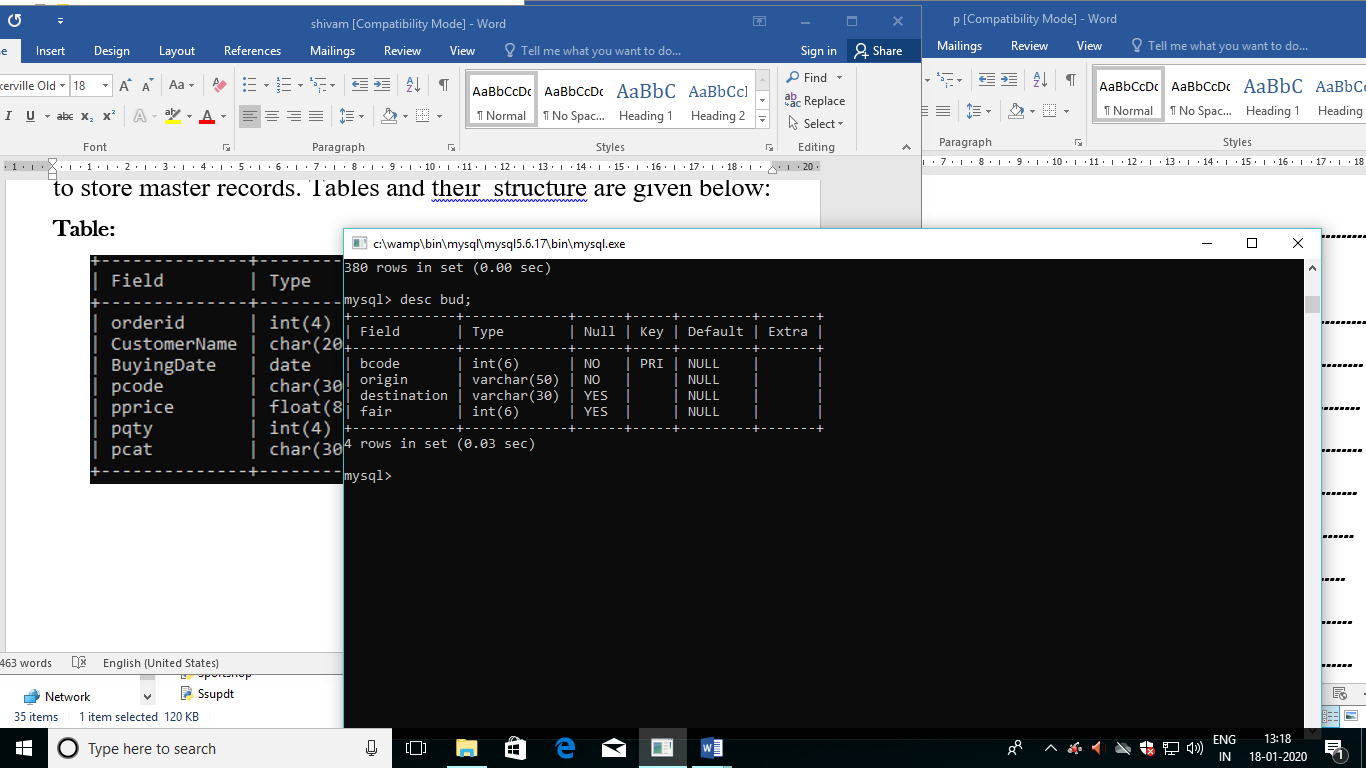
An important aspect of system design is the design of data storage structure. To begin with a logical model if data structure is developed first. The goodness of database design lies in the table structure and its relationship.

This software project maintains a database named **bub** which contains the following tables.

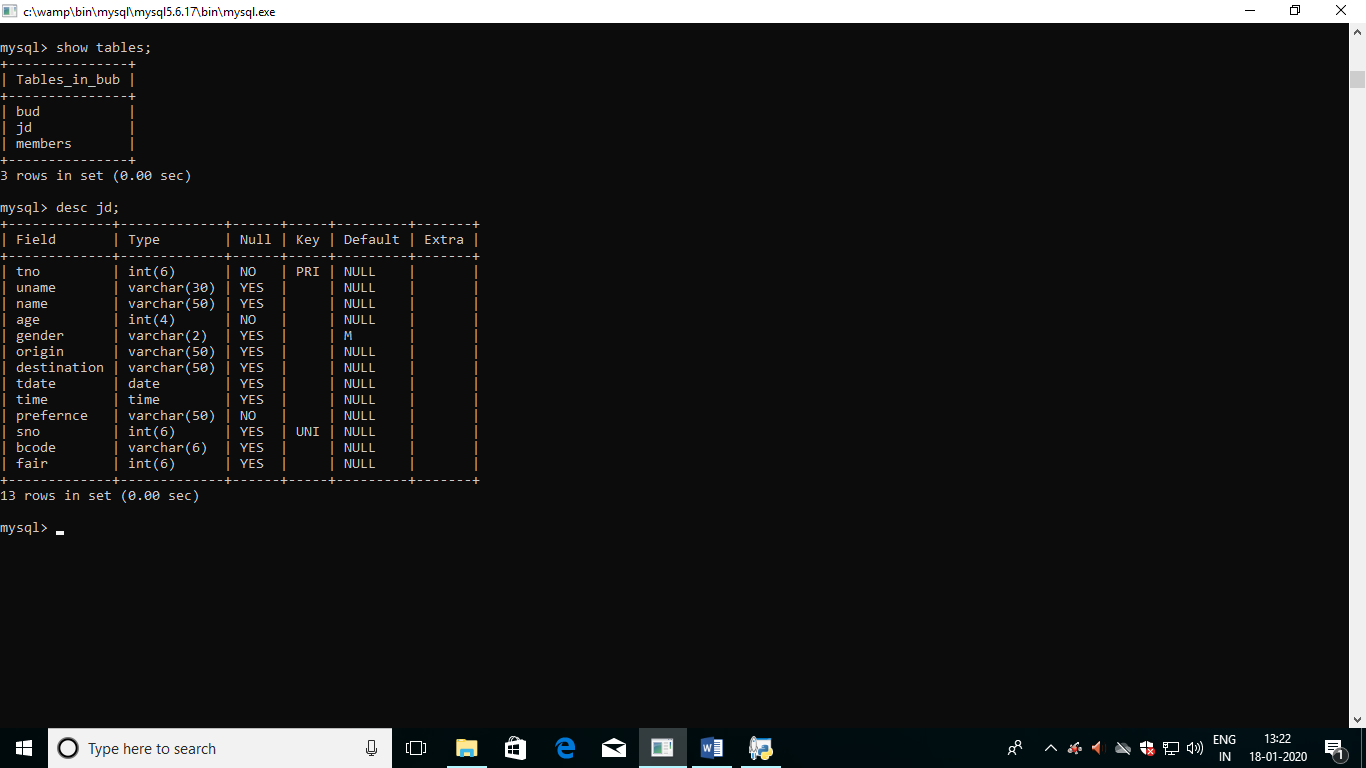
*Table Design:*

The database of Bus System contains 3 tables. The tables are normalised to minimize the redundancies of data and enforcing the validation rules of the organisation. Most of the tables are designed to store master records. Tables and their structure are given below:

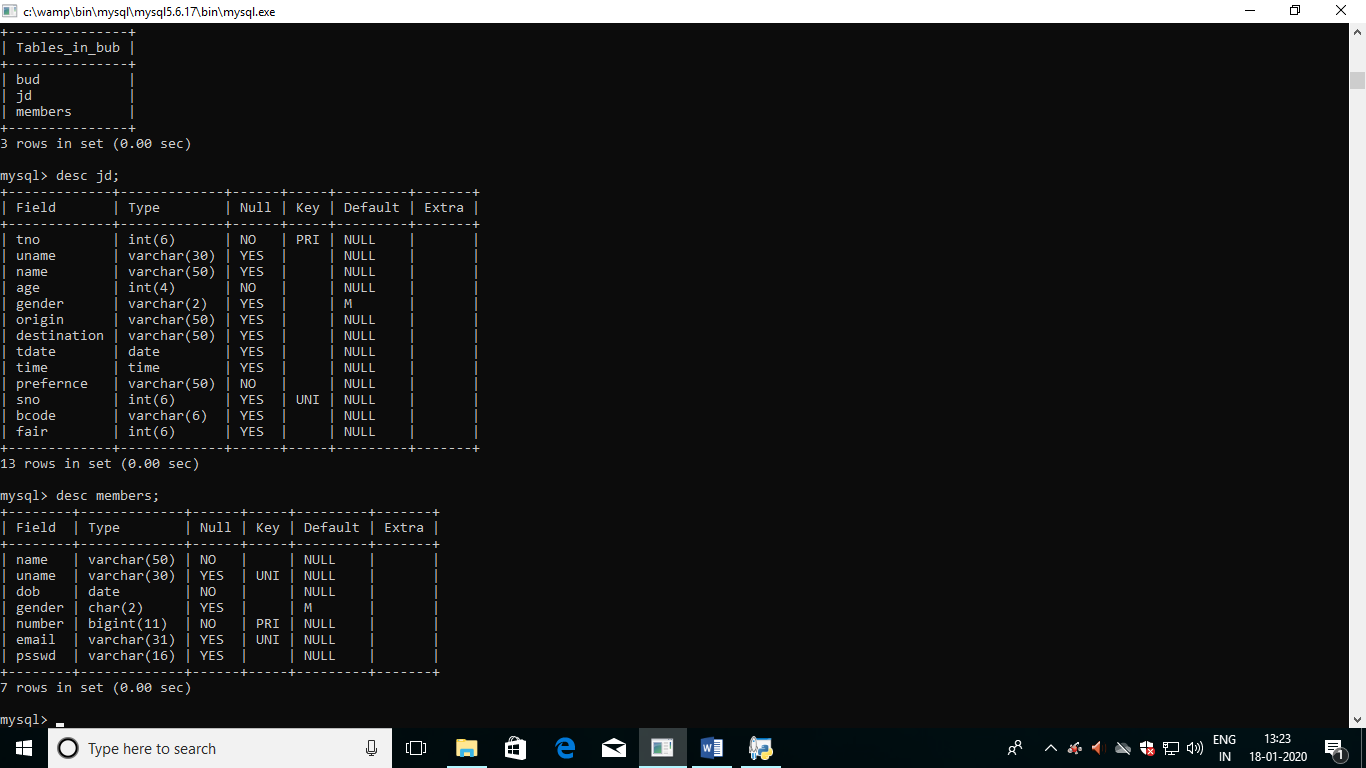
**Table: bud**



**Table: jd**



**Table: members**



**6.2 Flow Diagram**

**6.2 Program Code**

import mysql.connector as sqltor

from tkinter import\*

from functools import partial

import random

mycon=sqltor.connect(host='localhost',user='root',passwd='')

global cur

cur=mycon.cursor()

root = Tk()

def back(uname):

bac=Toplevel()

bac.geometry('400x400+0+0')

bac.title('Bus Ticket Booking')

photo1= PhotoImage(file='Capture.GIF')

Label(bac, image=photo1,bg="white").grid(row=0,column=0)

bac\_r = Label(bac)

bac\_r.grid(row=15,column=0)

def hell1(uname):

bac.withdraw()

third(uname)

partial(third,uname)

def hell2():

bac.withdraw()

show()

partial(show)

def show():

sh=Toplevel()

sh.geometry('400x400+0+0')

sh.title('Bus Ticket Booking')

photo1= PhotoImage(file='Capture.GIF')

Label(sh, image=photo1,bg="white").grid(row=0,column=0)

sh\_r = Label(sh)

sh\_r.grid(row=15,column=0)

Label(sh,text="Enter Ticket Number:",font="Times").grid(row=1,column=0,sticky=W)

s=StringVar()

tn = Entry(sh, textvariable=s,width=25).grid(row=1, column=0,sticky=E)

def m(s):

sch=Toplevel()

sh.withdraw()

sch.geometry('400x400+0+0')

sch.title('Booking Details')

sch\_r = Label(sch)

sch\_r.grid(row=15,column=0)

s=s.get()

try:

cur.execute("select \* from jd where tno={};".format(int(s)))

data=cur.fetchall()

if data==[]:

sch\_r.config(text="No tickets booked with this ticket Number")

else:

Label(sch,text="Ticket Number:").grid(row=1,column=0)

Label(sch,text="Traveller's Name:").grid(row=2,column=0)

Label(sch,text="Traveller's Age:").grid(row=3,column=0)

Label(sch,text="Gender:").grid(row=4,column=0)

Label(sch,text="Origin:").grid(row=5,column=0)

Label(sch,text="Destination:").grid(row=6,column=0)

Label(sch,text="Date:").grid(row=7,column=0)

Label(sch,text="Time:").grid(row=8,column=0)

Label(sch,text="Preference:").grid(row=9,column=0)

Label(sch,text="Seat Number:").grid(row=10,column=0)

Label(sch,text="Bus Code:").grid(row=11,column=0)

Label(sch,text="Fair:").grid(row=12,column=0)

tokra = Label(sch)

tokra.grid(row=1,column=2)

tokrb = Label(sch)

tokrb.grid(row=2,column=2)

tokrc = Label(sch)

tokrc.grid(row=3,column=2)

tokrd = Label(sch)

tokrd.grid(row=4,column=2)

tokre = Label(sch)

tokre.grid(row=5,column=2)

tokrf = Label(sch)

tokrf.grid(row=6,column=2)

tokrg = Label(sch)

tokrg.grid(row=7,column=2)

tokrh = Label(sch)

tokrh.grid(row=8,column=2)

tokri = Label(sch)

tokri.grid(row=9,column=2)

tokrj = Label(sch)

tokrj.grid(row=10,column=2)

tokrk = Label(sch)

tokrk.grid(row=11,column=2)

tokrl = Label(sch)

tokrl.grid(row=12,column=2)

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

except:

sch\_r.config(text="Invalid ticket number")

def l():

sch.withdraw()

sh.deiconify()

def cv():

sch.withdraw()

login()

a=Button(sch, text="Previous", command=l,bg="light cyan").grid(row=15, column=1)

a=Button(sch, text="Logout", command=cv,bg="light cyan").grid(row=16, column=1)

sch.mainloop()

def cv():

sh.withdraw()

login()

h=partial(m,s)

a=Button(sh, text="Check", command=h,bg="light cyan").grid(row=2, column=0)

a=Button(sh, text="Quit", command=sh.destroy,bg="light cyan").grid(row=2, column=0,sticky=E)

a=Button(sh, text="Logout", command=cv,bg="light cyan").grid(row=2, column=0,sticky=W)

sh.mainloop()

c1=partial(hell1,uname)

c2=partial(hell2)

a=Button(bac, text="Book Ticket", command=c1,bg="light cyan").grid(row=5, column=0)

a=Button(bac, text="Check Booking", command=c2,bg="light cyan").grid(row=6, column=0)

a=Button(bac, text="Quit", command=bac.destroy,bg="light cyan").grid(row=7, column=0)

bac.mainloop()

def third(uname):

thir=Toplevel()

thir.geometry('500x500+0+0')

thir.title("Select Your Journey")

tok\_r = Label(thir)

tok\_r.grid(row=15, column=0)

Label(thir,text="Enter Journey Details",font="Times").grid(row=0,column=1)

Label(thir,text="Traveller's Name:").grid(row=2,column=0)

Label(thir,text="Traveller's Age:").grid(row=3,column=0)

Label(thir,text="Gender:").grid(row=4,column=0)

Label(thir,text="Origin:").grid(row=5,column=0)

Label(thir,text="Destination:").grid(row=6,column=0)

Label(thir,text="Date:").grid(row=7,column=0)

Label(thir,text="Time:").grid(row=8,column=0)

Label(thir,text="Set Preferences:").grid(row=9,column=0)

origin=destination=["Delhi","Mumbai","Chennai","Gorakhpur","Lucknow","Jaipur","Agra","Bangalore","Goa","Varanasi","Bhopal","Manali","Hyderabad","Patna","Jammu","Pune","Amritsar","Chandigarh","Kota","Leh"]

vo=StringVar()

vd=StringVar()

vo.set("Delhi")

vd.set("Mumbai")

wo=OptionMenu(thir,vo,\*origin).grid(row=5,column=1)

wd=OptionMenu(thir,vd,\*destination).grid(row=6,column=1)

sp=["AC+Non-Sleeper","AC+Sleeper","Non-AC+Sleeper","Non-AC+Non-Sleeper"]

vs=StringVar()

vs.set("Non-AC+Sleeper")

ws=OptionMenu(thir,vs,\*sp).grid(row=9,column=1)

ame=StringVar()

ge=IntVar()

ge.set(25)

age=[x for x in range(1,131)]

enname = Entry(thir, textvariable=ame,width=25).grid(row=2, column=1)

wa=OptionMenu(thir,ge,\*age).grid(row=3,column=1)

global vg

vg=StringVar()

vg.set("M")

b=Radiobutton(thir,text='Male',variable=vg,font='none',value='M').grid(row=4,column=1,sticky=W)

b=Radiobutton(thir,text='Female',variable=vg,font='none',value='F').grid(row=4,column=1,sticky=E)

b=Radiobutton(thir,text='Third',variable=vg,font='none',value='T').grid(row=4,column=2)

v1 = StringVar()

v1.set("2020")

year=["2020"]

v2 = StringVar()

v2.set("01")

month=["01","02","03","04","05","06","07","08","09","10",'11',"12"]

v3=StringVar()

v3.set("01")

days=["01","02","03","04","05","06","07","08","09","10",'11',"12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"]

w1 = OptionMenu(thir,v1,\*year)

w1.grid(row=7,column=1,sticky=W)

w2 = OptionMenu(thir, v2,\*month)

w2.grid(row=7,column=1)

w3 = OptionMenu(thir,v3,\*days)

w3.grid(row=7,column=1,sticky=E)

t=StringVar()

t.set("16:00")

ti=[str(x)+":00" for x in range(4,25,4)]

w4 = OptionMenu(thir,t,\*ti)

w4.grid(row=8,column=1)

def No():

thir.deiconify()

ch.withdraw()

if len(str(v3))==1:

dob=str(v1)+'-'+str(v2)+'-'+'0'+str(v3)

cur.execute("select sno from jd where date='{}' and time='{}';".format(dob,t))

l=cur.fetchall()

def ch(ame,ge,vg,vo,vd,v1,v2,v3,t,vs):

enname=str(ame.get())

ge=int(ge.get())

vg=str(vg.get())

vo=str(vo.get())

vd=str(vd.get())

v1=str(v1.get())

v2=str(v2.get())

v3=str(v3.get())

t=str(t.get())

vs=str(vs.get())

re=Toplevel()

thir.withdraw()

re.geometry('800x700+0+0')

re.title('BUS TICKET BOOKING SYSTEM')

tokr = Label(re)

tokr.grid(row=14,column=2)

try:

cur.execute("create database if not exists bub;")

cur.execute("use bub;")

cur.execute('''create table if not exists jd

(tno int(6) primary key,

uname varchar(30) references members(uname),

name varchar(50),

age int(4) not null,

gender varchar(2) default 'M',

origin varchar(50) references bud(origin),

destination varchar(50) references bud(destination),

tdate date,

time time,

prefernce varchar(50) not null,

sno int(6) unique,

bcode varchar(6) references bud(bcode),

fair int(6) references bud(fair));''')

try:

dob=str(v1)+'-'+str(v2)+'-'+str(v3)

cur.execute("select tno,sno from jd where tdate='{}' and time='{}' and origin='{}' and destination='{}';".format(dob,t,vo,vd))

tn=cur.fetchall()

if tn==[]:

ttn=random.randint(10000,99999)

cur.execute("select bcode,fair from bud where origin='{}' and destination='{}';".format(vo,vd))

u=cur.fetchall()

s=random.randint(1,30)

cur.execute("insert into jd values({},'{}','{}',{},'{}','{}','{}','{}','{}','{}',{},'{}',{});".format(ttn,uname,enname,ge,vg,vo,vd,dob,t,vs,s,str(u[0][0]),u[0][1]))

mycon.commit()

cur.execute("select \* from jd where tno={}".format(ttn))

data=cur.fetchall()

Label(re,text="Ticket Number:").grid(row=1,column=0)

Label(re,text="Traveller's Name:").grid(row=2,column=0)

Label(re,text="Traveller's Age:").grid(row=3,column=0)

Label(re,text="Gender:").grid(row=4,column=0)

Label(re,text="Origin:").grid(row=5,column=0)

Label(re,text="Destination:").grid(row=6,column=0)

Label(re,text="Date:").grid(row=7,column=0)

Label(re,text="Time:").grid(row=8,column=0)

Label(re,text="Preference:").grid(row=9,column=0)

Label(re,text="Seat Number:").grid(row=10,column=0)

Label(re,text="Bus Code:").grid(row=11,column=0)

Label(re,text="Fair:").grid(row=12,column=0)

tokra = Label(re)

tokra.grid(row=1,column=2)

tokrb = Label(re)

tokrb.grid(row=2,column=2)

tokrc = Label(re)

tokrc.grid(row=3,column=2)

tokrd = Label(re)

tokrd.grid(row=4,column=2)

tokre = Label(re)

tokre.grid(row=5,column=2)

tokrf = Label(re)

tokrf.grid(row=6,column=2)

tokrg = Label(re)

tokrg.grid(row=7,column=2)

tokrh = Label(re)

tokrh.grid(row=8,column=2)

tokri = Label(re)

tokri.grid(row=9,column=2)

tokrj = Label(re)

tokrj.grid(row=10,column=2)

tokrk = Label(re)

tokrk.grid(row=11,column=2)

tokrl = Label(re)

tokrl.grid(row=12,column=2)

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

else:

bti=[]

for i in tn:

bti.append(i[0])

bsn=[]

for i in tn:

bsn.append(i[1])

try:

while True:

ttn=random.randint(10000,99999)

if ttn in bti:

continue

else:

cur.execute("select bcode,fair from bud where origin='{}' and destination='{}';".format(vo,vd))

u=cur.fetchall()

s=random.randint(1,30)

if s in bsn:

continue

else:

cur.execute("insert into jd values({},'{}','{}',{},'{}','{}','{}','{}','{}','{}',{},'{}',{});".format(ttn,uname,enname,ge,vg,vo,vd,dob,t,vs,s,str(u[0][0]),u[0][1]))

mycon.commit()

cur.execute("select \* from jd where tno={}".format(ttn))

data=cur.fetchall()

Label(re,text="Ticket Number:").grid(row=1,column=0)

Label(re,text="Traveller's Name:").grid(row=2,column=0)

Label(re,text="Traveller's Age:").grid(row=3,column=0)

Label(re,text="Gender:").grid(row=4,column=0)

Label(re,text="Origin:").grid(row=5,column=0)

Label(re,text="Destination:").grid(row=6,column=0)

Label(re,text="Date:").grid(row=7,column=0)

Label(re,text="Time:").grid(row=8,column=0)

Label(re,text="Preference:").grid(row=9,column=0)

Label(re,text="Seat Number:").grid(row=10,column=0)

Label(re,text="Bus Code:").grid(row=11,column=0)

Label(re,text="Fair:").grid(row=12,column=0)

tokra = Label(re)

tokra.grid(row=1,column=2)

tokrb = Label(re)

tokrb.grid(row=2,column=2)

tokrc = Label(re)

tokrc.grid(row=3,column=2)

tokrd = Label(re)

tokrd.grid(row=4,column=2)

tokre = Label(re)

tokre.grid(row=5,column=2)

tokrf = Label(re)

tokrf.grid(row=6,column=2)

tokrg = Label(re)

tokrg.grid(row=7,column=2)

tokrh = Label(re)

tokrh.grid(row=8,column=2)

tokri = Label(re)

tokri.grid(row=9,column=2)

tokrj = Label(re)

tokrj.grid(row=10,column=2)

tokrk = Label(re)

tokrk.grid(row=11,column=2)

tokrl = Label(re)

tokrl.grid(row=12,column=2)

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

break

except:

pass

except:

tokr.config(text="all tickets are booked")

except sqltor.errors.InterfaceError as ie:

try:

dob=str(v1)+'-'+str(v2)+'-'+str(v3)

cur.execute("select tno,sno from jd where tdate='{}' and time='{}' and origin='{}' and destination='{}';".format(dob,t,vo,vd))

tn=cur.fetchall()

if tn==[]:

ttn=random.randint(10000,99999)

cur.execute("select bcode,fair from bud where origin='{}' and destination='{}';".format(vo,vd))

u=cur.fetchall()

s=random.randint(1,30)

cur.execute("insert into jd values({},'{}','{}',{},'{}','{}','{}','{}','{}','{}',{},'{}',{});".format(ttn,uname,enname,ge,vg,vo,vd,dob,t,vs,s,str(u[0][0]),u[0][1]))

mycon.commit()

cur.execute("select \* from jd where tno={}".format(ttn))

data=cur.fetchall()

Label(re,text="Ticket Number:").grid(row=1,column=0)

Label(re,text="Traveller's Name:").grid(row=2,column=0)

Label(re,text="Traveller's Age:").grid(row=3,column=0)

Label(re,text="Gender:").grid(row=4,column=0)

Label(re,text="Origin:").grid(row=5,column=0)

Label(re,text="Destination:").grid(row=6,column=0)

Label(re,text="Date:").grid(row=7,column=0)

Label(re,text="Time:").grid(row=8,column=0)

Label(re,text="Preference:").grid(row=9,column=0)

Label(re,text="Seat Number:").grid(row=10,column=0)

Label(re,text="Bus Code:").grid(row=11,column=0)

Label(re,text="Fair:").grid(row=12,column=0)

tokra = Label(re)

tokra.grid(row=1,column=2)

tokrb = Label(re)

tokrb.grid(row=2,column=2)

tokrc = Label(re)

tokrc.grid(row=3,column=2)

tokrd = Label(re)

tokrd.grid(row=4,column=2)

tokre = Label(re)

tokre.grid(row=5,column=2)

tokrf = Label(re)

tokrf.grid(row=6,column=2)

tokrg = Label(re)

tokrg.grid(row=7,column=2)

tokrh = Label(re)

tokrh.grid(row=8,column=2)

tokri = Label(re)

tokri.grid(row=9,column=2)

tokrj = Label(re)

tokrj.grid(row=10,column=2)

tokrk = Label(re)

tokrk.grid(row=11,column=2)

tokrl = Label(re)

tokrl.grid(row=12,column=2)

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

else:

bti=[]

for i in tn:

bti.append(i[0])

bsn=[]

for i in tn:

bsn.append(i[1])

try:

while True:

ttn=random.randint(10000,99999)

if ttn in bti:

continue

else:

cur.execute("select bcode,fair from bud where origin='{}' and destination='{}';".format(vo,vd))

u=cur.fetchall()

s=random.randint(1,30)

if s in bsn:

continue

else:

cur.execute("insert into jd values({},'{}','{}',{},'{}','{}','{}','{}','{}','{}',{},'{}',{});".format(ttn,uname,enname,ge,vg,vo,vd,dob,t,vs,s,str(u[0][0]),u[0][1]))

mycon.commit()

cur.execute("select \* from jd where tno={}".format(ttn))

data=cur.fetchall()

Label(re,text="Ticket Number:").grid(row=1,column=0)

Label(re,text="Traveller's Name:").grid(row=2,column=0)

Label(re,text="Traveller's Age:").grid(row=3,column=0)

Label(re,text="Gender:").grid(row=4,column=0)

Label(re,text="Origin:").grid(row=5,column=0)

Label(re,text="Destination:").grid(row=6,column=0)

Label(re,text="Date:").grid(row=7,column=0)

Label(re,text="Time:").grid(row=8,column=0)

Label(re,text="Preference:").grid(row=9,column=0)

Label(re,text="Seat Number:").grid(row=10,column=0)

Label(re,text="Bus Code:").grid(row=11,column=0)

Label(re,text="Fair:").grid(row=12,column=0)

tokra = Label(re)

tokra.grid(row=1,column=2)

tokrb = Label(re)

tokrb.grid(row=2,column=2)

tokrc = Label(re)

tokrc.grid(row=3,column=2)

tokrd = Label(re)

tokrd.grid(row=4,column=2)

tokre = Label(re)

tokre.grid(row=5,column=2)

tokrf = Label(re)

tokrf.grid(row=6,column=2)

tokrg = Label(re)

tokrg.grid(row=7,column=2)

tokrh = Label(re)

tokrh.grid(row=8,column=2)

tokri = Label(re)

tokri.grid(row=9,column=2)

tokrj = Label(re)

tokrj.grid(row=10,column=2)

tokrk = Label(re)

tokrk.grid(row=11,column=2)

tokrl = Label(re)

tokrl.grid(row=12,column=2)

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

break

except:

pass

except:

tokr.config(text="Sorry all are booked")

else:

try:

dob=str(v1)+'-'+str(v2)+'-'+str(v3)

cur.execute("select tno,sno from jd where tdate='{}' and time='{}' and origin='{}' and destination='{}';".format(dob,t,vo,vd))

tn=cur.fetchall()

if tn==[]:

ttn=random.randint(10000,99999)

cur.execute("select bcode,fair from bud where origin='{}' and destination='{}';".format(vo,vd))

u=cur.fetchall()

s=random.randint(1,30)

cur.execute("insert into jd values({},'{}','{}',{},'{}','{}','{}','{}','{}','{}',{},'{}',{});".format(ttn,uname,enname,ge,vg,vo,vd,dob,t,vs,s,str(u[0][0]),u[0][1]))

mycon.commit()

cur.execute("select \* from jd where tno={}".format(ttn))

data=cur.fetchall()

Label(re,text="Ticket Number:").grid(row=1,column=0)

Label(re,text="Traveller's Name:").grid(row=2,column=0)

Label(re,text="Traveller's Age:").grid(row=3,column=0)

Label(re,text="Gender:").grid(row=4,column=0)

Label(re,text="Origin:").grid(row=5,column=0)

Label(re,text="Destination:").grid(row=6,column=0)

Label(re,text="Date:").grid(row=7,column=0)

Label(re,text="Time:").grid(row=8,column=0)

Label(re,text="Preference:").grid(row=9,column=0)

Label(re,text="Seat Number:").grid(row=10,column=0)

Label(re,text="Bus Code:").grid(row=11,column=0)

Label(re,text="Fair:").grid(row=12,column=0)

tokra = Label(re)

tokra.grid(row=1,column=2)

tokrb = Label(re)

tokrb.grid(row=2,column=2)

tokrc = Label(re)

tokrc.grid(row=3,column=2)

tokrd = Label(re)

tokrd.grid(row=4,column=2)

tokre = Label(re)

tokre.grid(row=5,column=2)

tokrf = Label(re)

tokrf.grid(row=6,column=2)

tokrg = Label(re)

tokrg.grid(row=7,column=2)

tokrh = Label(re)

tokrh.grid(row=8,column=2)

tokri = Label(re)

tokri.grid(row=9,column=2)

tokrj = Label(re)

tokrj.grid(row=10,column=2)

tokrk = Label(re)

tokrk.grid(row=11,column=2)

tokrl = Label(re)

tokrl.grid(row=12,column=2)

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

else:

bti=[]

for i in tn:

bti.append(i[0])

bsn=[]

for i in tn:

bsn.append(i[1])

try:

while True:

ttn=random.randint(10000,99999)

if ttn in bti:

continue

else:

cur.execute("select bcode,fair from bud where origin='{}' and destination='{}';".format(vo,vd))

u=cur.fetchall()

s=random.randint(1,30)

if s in bsn:

continue

else:

cur.execute("insert into jd values({},'{}','{}',{},'{}','{}','{}','{}','{}','{}',{},'{}',{});".format(ttn,uname,enname,ge,vg,vo,vd,dob,t,vs,s,str(u[0][0]),u[0][1]))

mycon.commit()

print('l')

cur.execute("select \* from jd where tno={}".format(ttn))

data=cur.fetchall()

print('k')

Label(re,text="Ticket Number:").grid(row=1,column=0)

Label(re,text="Traveller's Name:").grid(row=2,column=0)

Label(re,text="Traveller's Age:").grid(row=3,column=0)

Label(re,text="Gender:").grid(row=4,column=0)

Label(re,text="Origin:").grid(row=5,column=0)

Label(re,text="Destination:").grid(row=6,column=0)

Label(re,text="Date:").grid(row=7,column=0)

Label(re,text="Time:").grid(row=8,column=0)

Label(re,text="Preference:").grid(row=9,column=0)

Label(re,text="Seat Number:").grid(row=10,column=0)

Label(re,text="Bus Code:").grid(row=11,column=0)

Label(re,text="Fair:").grid(row=12,column=0)

print('lk')

tokra = Label(re)

tokra.grid(row=1,column=2)

tokrb = Label(re)

tokrb.grid(row=2,column=2)

tokrc = Label(re)

tokrc.grid(row=3,column=2)

tokrd = Label(re)

tokrd.grid(row=4,column=2)

tokre = Label(re)

tokre.grid(row=5,column=2)

tokrf = Label(re)

tokrf.grid(row=6,column=2)

tokrg = Label(re)

tokrg.grid(row=7,column=2)

tokrh = Label(re)

tokrh.grid(row=8,column=2)

tokri = Label(re)

tokri.grid(row=9,column=2)

tokrj = Label(re)

tokrj.grid(row=10,column=2)

tokrk = Label(re)

tokrk.grid(row=11,column=2)

tokrl = Label(re)

tokrl.grid(row=12,column=2)

print('kl')

tokra.config(text=data[0][0])

tokrb.config(text=data[0][2])

tokrc.config(text=data[0][3])

tokrd.config(text=data[0][4])

tokre.config(text=data[0][5])

tokrf.config(text=data[0][6])

tokrg.config(text=data[0][7])

tokrh.config(text=data[0][8])

tokri.config(text=data[0][9])

tokrj.config(text=data[0][10])

tokrk.config(text=data[0][11])

tokrl.config(text=data[0][12])

print('kkl')

break

except:

pass

except:

tokr.config(text="Sorry all tickets are")

def m():

re.withdraw()

thir.deiconify()

def cv():

re.withdraw()

login()

a=Button(re, text="Previous", command=m,bg="light cyan").grid(row=15, column=1)

a=Button(re, text="Logout", command=cv,bg="light cyan").grid(row=16, column=1)

re.mainloop()

def mp():

thir.withdraw()

back()

c=partial(ch,ame,ge,vg,vo,vd,v1,v2,v3,t,vs)

a=Button(thir, text="Confirm Booking", command=c,bg="light cyan").grid(row=10, column=1)

a=Button(thir, text="Previous", command=mp,bg="light cyan").grid(row=11, column=1)

thir.mainloop()

def register():

reg=Toplevel()

root.withdraw()

reg.geometry('800x700+0+0')

reg.title('BUS TICKET BOOKING SYSTEM')

photo1= PhotoImage(file='Capture.GIF')

Label(reg, image=photo1).grid(sticky=W)

top\_r=Label(reg)

top\_r.grid(row=12,column=0)

Label(reg,text="-------------------Register---------------",fg='black',font="Times").grid(row=2,column=0,sticky=W+N+S+E)

Label(reg,text='Name: ',fg='black',font="none").grid(row=3,column=0)

Label(reg,text='Username: ',fg='black',font="none").grid(row=4,column=0)

Label(reg,text='Date of Birth: ',fg='black',font="none").grid(row=5,column=0)

Label(reg,text='Gender: ',fg='black',font="none").grid(row=6,column=0)

global v

v=StringVar()

v.set("M")

b=Radiobutton(reg,text='Male',variable=v,fg='black',font='none',value='M').grid(row=6,column=0,sticky=E)

b=Radiobutton(reg,text='Female',variable=v,fg='black',font='none',value='F').grid(row=6,column=1,sticky=E)

b=Radiobutton(reg,text='Third',variable=v,fg='black',font='none',value='T').grid(row=6,column=2)

Label(reg,text='Number: ',fg='black',font="none").grid(row=7,column=0)

Label(reg,text='Email:',fg='black',font="none").grid(row=8,column=0)

Label(reg,text='Password: ',fg='black',font="none").grid(row=9,column=0)

Label(reg,text='Confirm Password: ',fg='black',font="none").grid(row=10,column=0)

global na

global un

global dob

global pd1

global pd2

global em

global nu

na=StringVar()

un=StringVar()

dob=StringVar()

pd1=StringVar()

pd2=StringVar()

em=StringVar()

nu=IntVar()

nu.set("")

global v1

v1 = StringVar()

v1.set("2000")

year=[str(i) for i in range(1940,2011)]

global v2

v2 = StringVar()

v2.set("01")

month=["01","02","03","04","05","06","07","08","09","10",'11',"12"]

global v3

v3=StringVar()

v3.set("1")

days=[ str(x) for x in range(1,32)]

w1 = OptionMenu(reg,v1,\*year)

w1.grid(row=5,column=0,sticky=E)

w2 = OptionMenu(reg, v2,\*month)

w2.grid(row=5,column=1,sticky=W)

w3 = OptionMenu(reg,v3,\*days)

w3.grid(row=5,column=2)

entryna=Entry(reg,textvariable=na).grid(row=3,column=0,sticky=E)

entryun=Entry(reg,textvariable=un).grid(row=4,column=0,sticky=E)

#entrydob=Entry(reg,textvariable=dob,fg="grey").grid(row=5,column=0,sticky=E)

entrynu=Entry(reg,textvariable=nu).grid(row=7,column=0,sticky=E)

entryem=Entry(reg,textvariable=em).grid(row=8,column=0,sticky=E)

entrypd1=Entry(reg, show="\*",textvariable=pd1).grid(row=9,column=0,sticky=E)

entrypd2=Entry(reg, show="\*",textvariable=pd2).grid(row=10,column=0,sticky=E)

def new1(a,u,v1,v2,v3,v,n,m,p1,p2):

try:

na=a.get()

un=u.get()

v1=v1.get()

v2=v2.get()

v3=v3.get()

v=v.get()

nu=n.get()

em=m.get()

pd=p1.get()

ps=p2.get()

try:

if pd==ps:

if len(str(v3))==1:

dob=str(v1)+'-'+str(v2)+'-'+'0'+str(v3)

cur.execute('create database if not exists bub;')

cur.execute('use bub;')

cur.execute("create table if not exists members( name varchar(50) not null,uname varchar(20) unique key,dob date not null,gender char(2) default 'M',number bigint(11) primary key,email varchar(31) unique key,psswd varchar(16));")

mycon.commit()

cur.execute("insert into members values('{}','{}','{}','{}',{},'{}','{}');".format(na,un,dob,v,nu,em,pd))

mycon.commit()

top\_r.config(text='''You have registered successfully

Now login to continue.''')

else:

dob=str(v1)+'-'+str(v2)+'-'+str(v3)

cur.execute('create database if not exists bub;')

cur.execute('use bub;')

cur.execute("create table if not exists members( name varchar(50) not null,uname varchar(20) unique key,dob date not null,gender char(2) default 'M',number bigint(11) primary key,email varchar(31) unique key,psswd varchar(16));")

mycon.commit()

cur.execute("insert into members values('{}','{}','{}','{}',{},'{}','{}');".format(na,un,dob,v,nu,em,pd))

mycon.commit()

top\_r.config(text='''You have registered

Now login to continue.''')

else:

top\_r.config(text="Password doesn't match")

except:

top\_r.config(text="Invalid Details Entered")

except:

top\_r.config(text="Invalid Details Entered")

def new2():

reg.withdraw()

login()

new=partial(new1,na,un,v1,v2,v3,v,nu,em,pd1,pd2)

a=Button(reg, text="REGISTER",command=new,bg="light cyan").grid(row=11, column=0,sticky=W)

a=Button(reg, text="QUIT",command=reg.destroy,bg="light cyan").grid(row=11, column=0)

a=Button(reg, text="LOGIN",command=new2,bg="light cyan").grid(row=11, column=0,sticky=E)

reg.mainloop()

def login():

top=Toplevel()

root.withdraw()

top.geometry('400x400+0+0')

top.title('BUS TICKET BOOKING SYSTEM')

photo1= PhotoImage(file='Capture.GIF')

Label(top, image=photo1).grid(sticky=W)

Label(top,text="-------------------Log In---------------",fg='black',font="Times").grid(row=2,column=0,sticky=W+N+S+E)

Label(top,text='Username: ',fg='black',font="none").grid(row=5,column=0)

Label(top,text='Password: ',fg='black',font="none").grid(row=6,column=0)

#Entry sites---------------------------------------------

tot\_r = Label(top)

tot\_r.grid(row=10, column=0)

global uname

uname = StringVar()

psswd = StringVar()

entryNum1 = Entry(top, textvariable=uname).grid(row=5, column=0)

entryNum2 = Entry(top,show ='\*', textvariable=psswd).grid(row=6, column=0)

def add\_r(tot\_r,n1, n2):

num1 = n1.get()

num2 = n2.get()

try:

cur.execute("create database if not exists bub;")

cur.execute("use bub;")

cur.execute("create table if not exists members( name varchar(50) not null,uname varchar(30) unique key,dob date not null,gender char(2) default 'M',number bigint(11) primary key,email varchar(31) unique key,psswd varchar(16));")

try:

cur.execute('select uname from members where number=9839381444;')

q=cur.fetchall()

if q==[]:

cur.execute("insert into members values('Pravin Kumar Shukla','133aspravin','2003-03-01','M',9839381444,'133aspravin@gmail.com','bus');")

mycon.commit()

st="select uname,psswd from members where uname = '{}' and psswd = '{}';".format(num1,num2)

cur.execute(st)

ah=cur.fetchall()

if ah==[]:

tot\_r.config("Invalid Username and/or Password")

else:

top.withdraw()

back(num1)

z=partial(back,num1)

except sqltor.MySQLInterfaceError:

st="select uname,psswd from members where uname= '{}' and psswd= '{}';".format(num1,num2)

cur.execute(st)

x=cur.fetchall()

if x==[]:

tot\_r.config(text="Invalid Username and/or Password")

else:

top.withdraw()

back(num1)

z=partial(back,num1)

else:

st="select uname,psswd from members where uname= '{}' and psswd= '{}';".format(num1,num2)

cur.execute(st)

x=cur.fetchall()

if x==[]:

tot\_r.config(text="Invalid Username and/or Password")

else:

top.withdraw()

back(num1)

z=partial(back,num1)

except sqltor.errors.InterfaceError as ie:

cur.execute("create database if not exists bub;")

cur.execute("use bub;")

cur.execute("create table if not exists members( name varchar(50) not null,uname varchar(30) unique key,dob date not null,gender char(2) default 'M',number bigint(11) primary key,email varchar(31) unique key,psswd varchar(16));")

try:

print('f')

cur.execute("insert into members values('Pravin Kumar Shukla','133aspravin','2003-03-01','M',9839381444,'133aspravin@gmail.com','bus');")

mycon.commit()

st="select uname,psswd from members where uname = '{}' and psswd = '{}';".format(num1,num2)

cur.execute(st)

ah=cur.fetchall()

if ah==[]:

tot\_r.config(text="Invalid Username and/or Password")

else:

top.withdraw()

back(num1)

z=partial(back,num1)

except:

st="select uname,psswd from members where uname= '{}' and psswd= '{}';".format(num1,num2)

cur.execute(st)

ah=cur.fetchall()

if ah==[]:

tot\_r.config(text="Invalid Username and/or Password")

else:

top.withdraw()

back(num1)

z=partial(back,num1)

def fg():

root.deiconify()

top.withdraw()

call\_add = partial(add\_r,tot\_r,uname,psswd)

addCal = Button(top, text="LOGIN", command=call\_add,bg="light cyan").grid(row=8, column=0,sticky=NW)

a=Button(top, text="Quit", command=top.destroy,bg="light cyan").grid(row=8, column=0)

a=Button(top, text="Main Menu", command=fg,bg="light cyan").grid(row=8, column=0,sticky=NE)

top.mainloop()

root.geometry('500x400+0+0')

root.title('BUS TICKET BOOKING SYSTEM')

photo1= PhotoImage(file='Capture.GIF')

Label(root, image=photo1,bg="white").grid(row=0,column=0)

a=Button(root, text="LOGIN", command=login,bg="light cyan").grid(row=5, column=0)

a=Button(root, text="REGISTER", command=register,bg="light cyan").grid(row=6, column=0)

a=Button(root, text="Quit", command=root.destroy,bg="light cyan").grid(row=7, column=0)

try:

cur.execute("create database if not exists bub;")

cur.execute("use bub;")

cur.execute("create table if not exists bud(bcode int(6) primary key,origin varchar(50) not null,destination varchar(30),fair int(6));")

try:

origin=["Delhi","Mumbai","Chennai","Gorakhpur","Lucknow","Jaipur","Agra","Bangalore","Goa","Varanasi","Bhopal","Manali","Hyderabad","Patna","Jammu","Pune","Amritsar","Chandigarh","Kota","Leh"]

destination=["Delhi","Mumbai","Chennai","Gorakhpur","Lucknow","Jaipur","Agra","Bangalore","Goa","Varanasi","Bhopal","Manali","Hyderabad","Patna","Jammu","Pune","Amritsar","Chandigarh","Kota","Leh"]

fair=[500,660,370,420,630,820,960,870,1520,1240,1420,900,400,600,1800]

for i in range(1,len(origin)+1):

for j in range(1,len(destination)+1):

if i!=j:

if len(str(j))==1:

j="0"+str(j)

cur.execute("insert into bud values( {},'{}','{}',{});".format(int(str(i)+str(j)),origin[i-1],destination[int(j)-1],fair[random.randint(0,14)]))

mycon.commit()

else:

cur.execute("insert into bud values( {},'{}','{}',{});".format(int(str(i)+str(j)),origin[i-1],destination[int(j)-1],fair[random.randint(0,14)]))

mycon.commit()

except:

pass

except sqltor.errors.InterfaceError as ie:

cur.execute("create database if not exists bub;")

cur.execute("use bub;")

cur.execute("create table if not exists bud(bcode int(6) primary key,origin varchar(50) not null,destination varchar(30),fair int(6));")

try:

origin=["Delhi","Mumbai","Chennai","Gorakhpur","Lucknow","Jaipur","Agra","Bangalore","Goa","Varanasi","Bhopal","Manali","Hyderabad","Patna","Jammu","Pune","Amritsar","Chandigarh","Kota","Leh"]

destination=["Delhi","Mumbai","Chennai","Gorakhpur","Lucknow","Jaipur","Agra","Bangalore","Goa","Varanasi","Bhopal","Manali","Hyderabad","Patna","Jammu","Pune","Amritsar","Chandigarh","Kota","Leh"]

fair=[500,660,370,420,630,820,960,870,1520,1240,1420,900,400,600,1800]

for i in range(1,len(origin)+1):

for j in range(1,len(destination)+1):

if i!=j:

cur.execute("insert into bud values({},'{}','{}',{});".format(int(str(i)+str(j)),origin[i-1],destination[j-1],fair[random.randint(1,15)]))

mycon.commit()

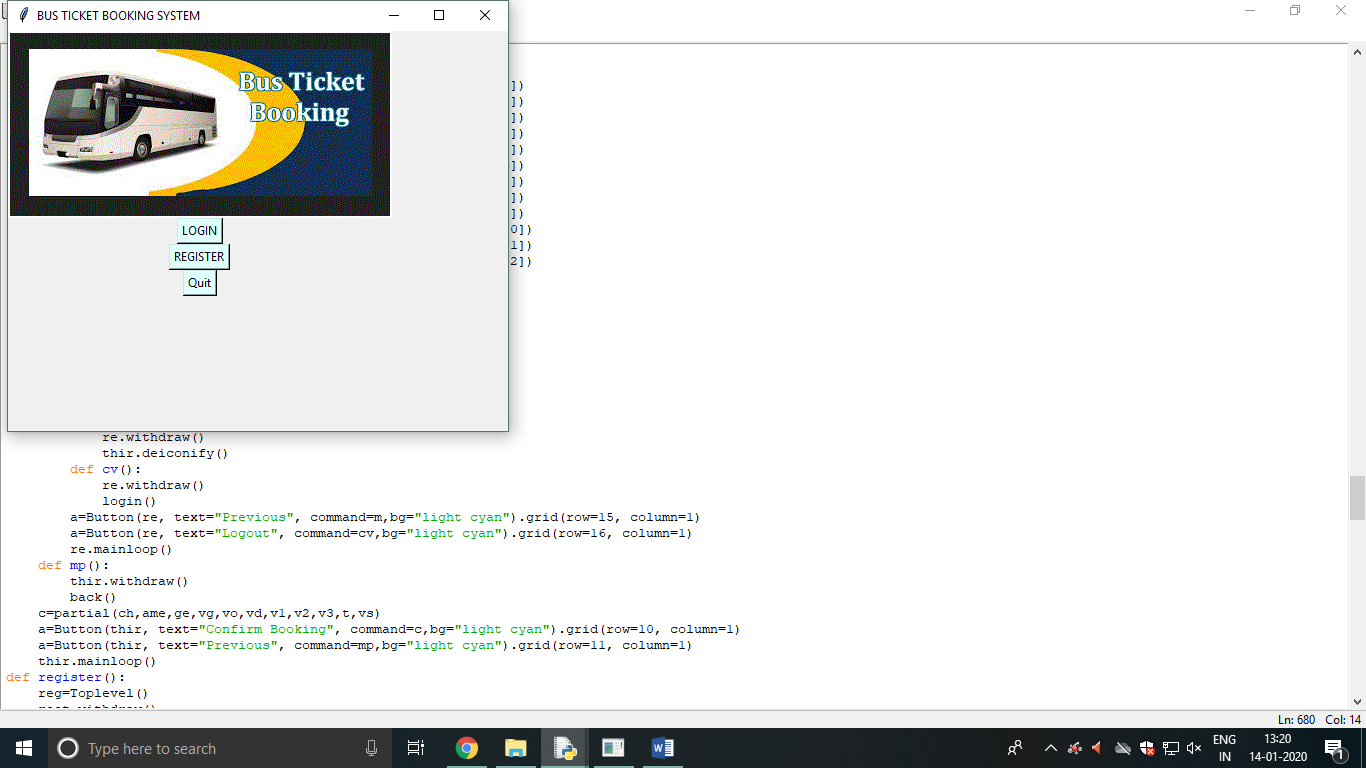
except:

print('h')

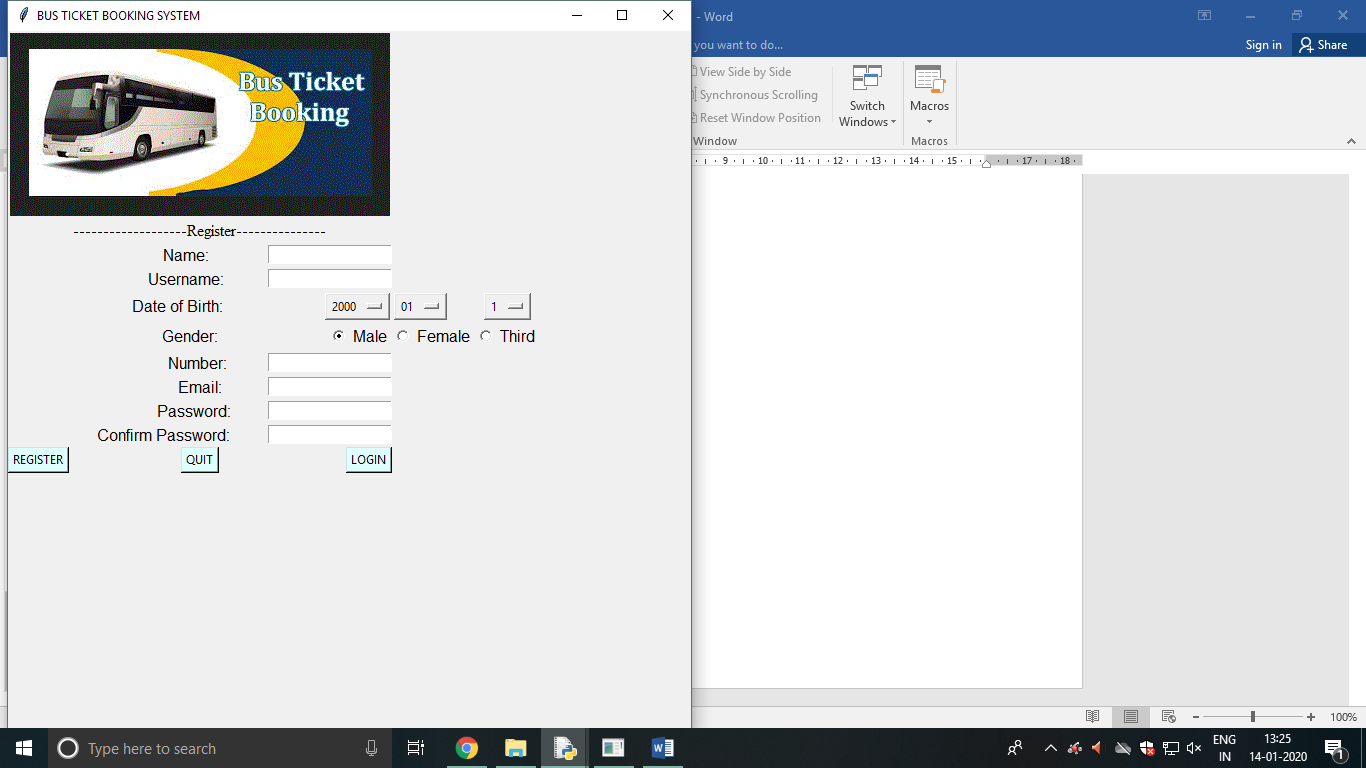
root.mainloop()

**6.3 Outputs**

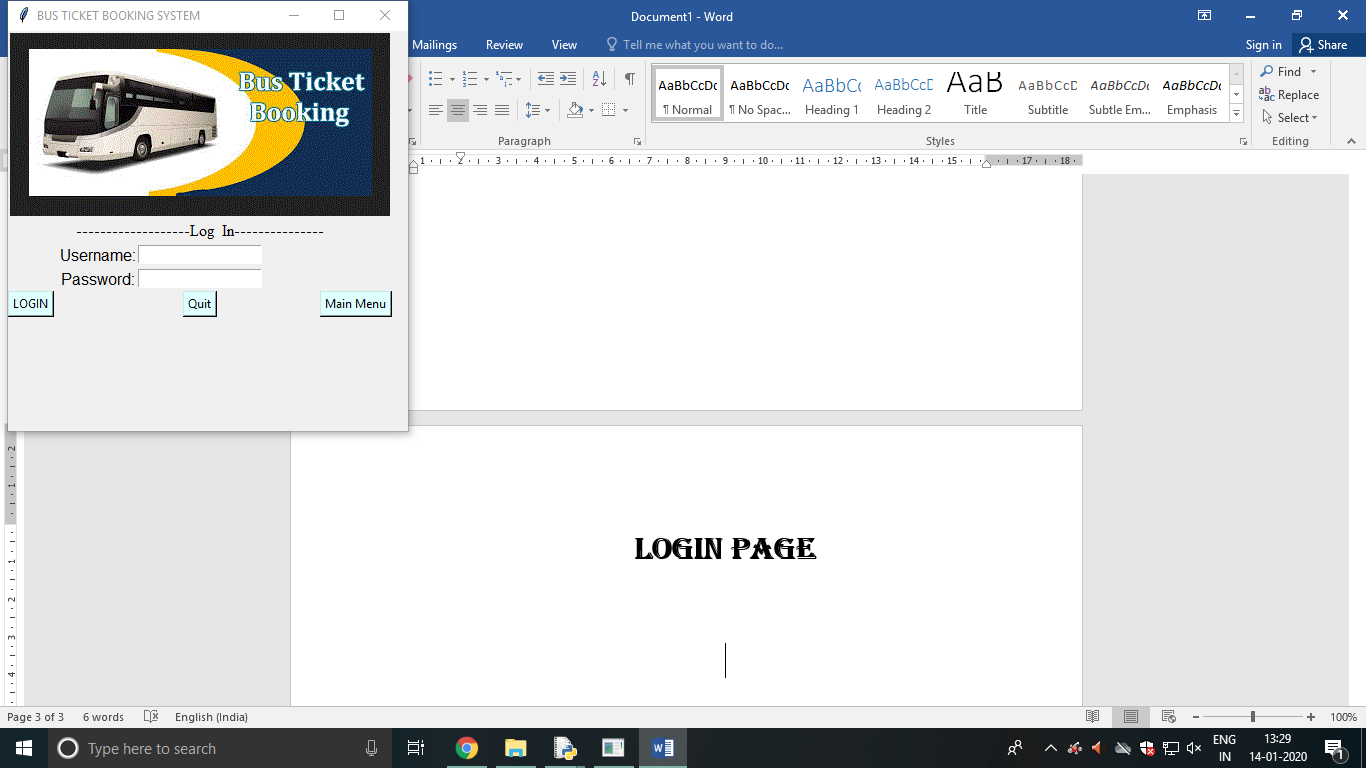
**main page**



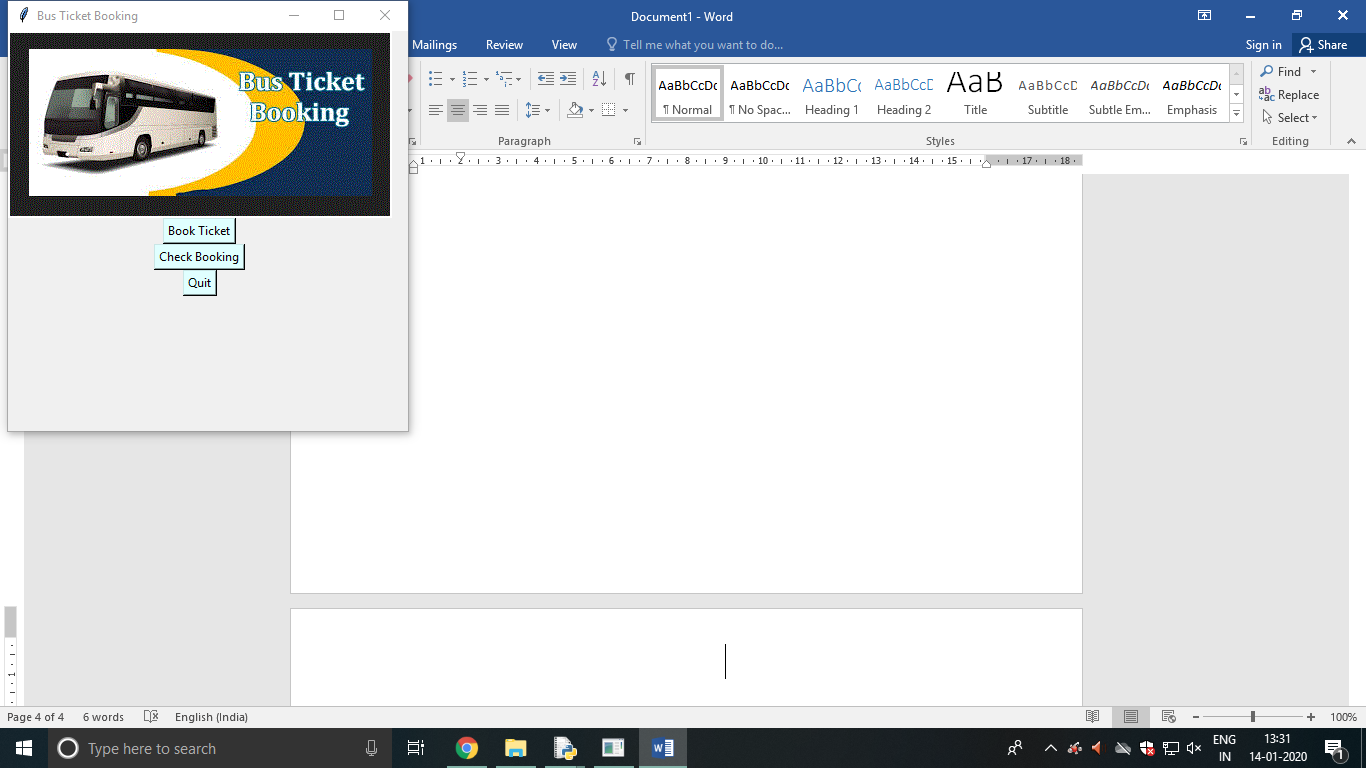
**Register page**



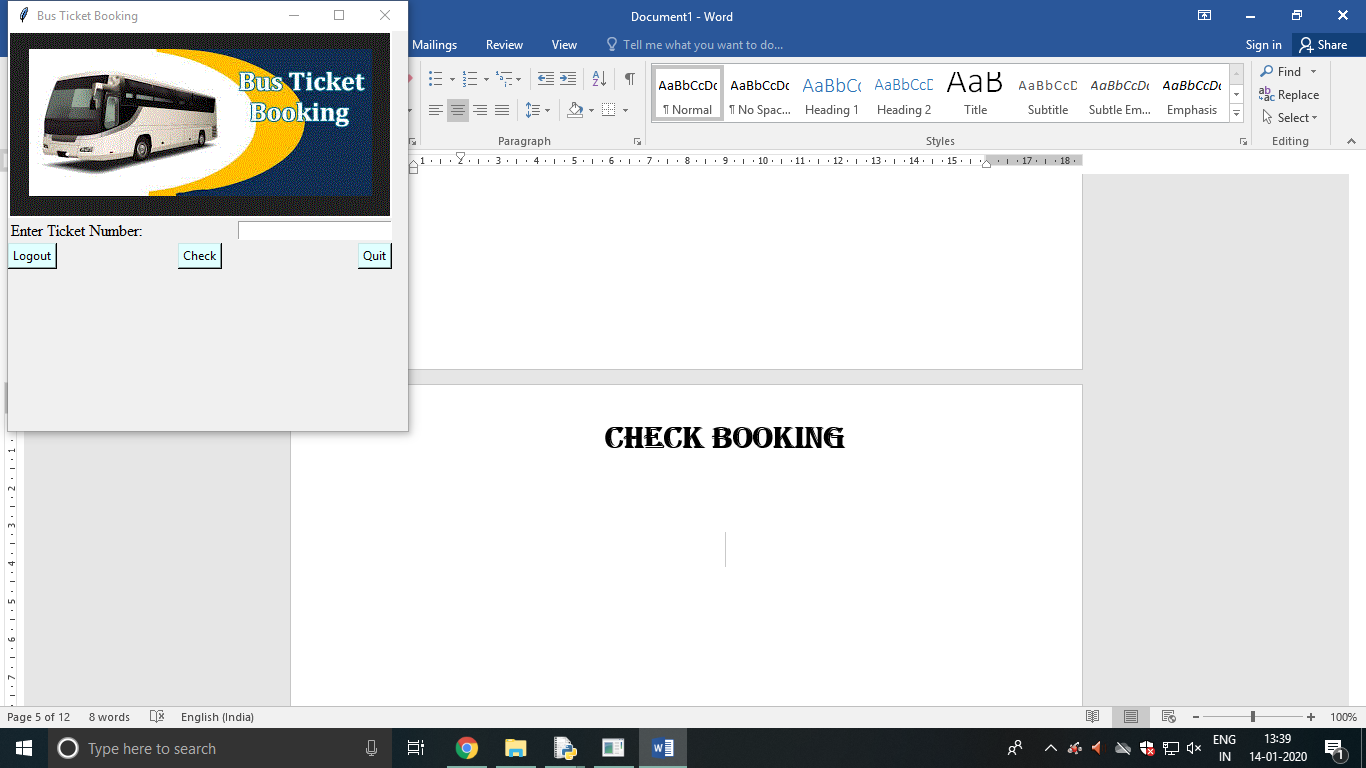
**Login page**



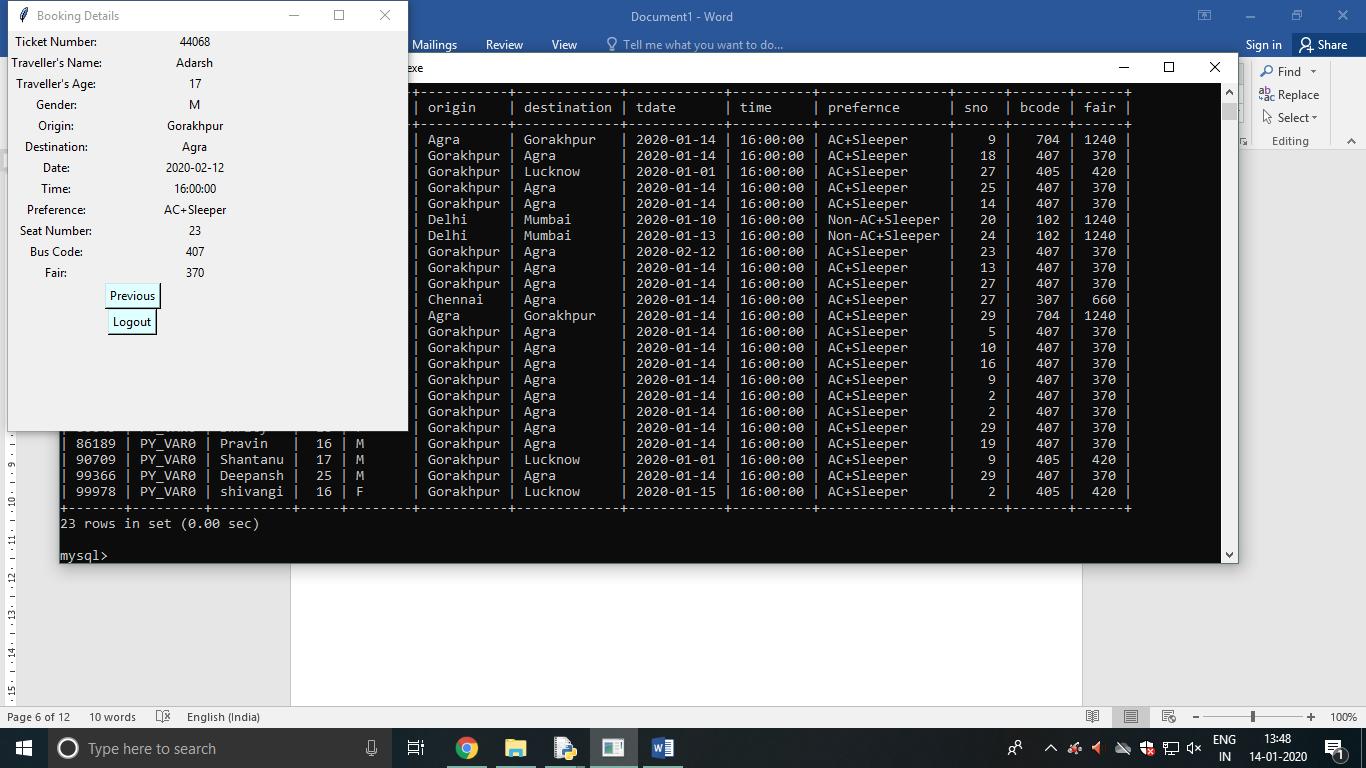
**Option menu**



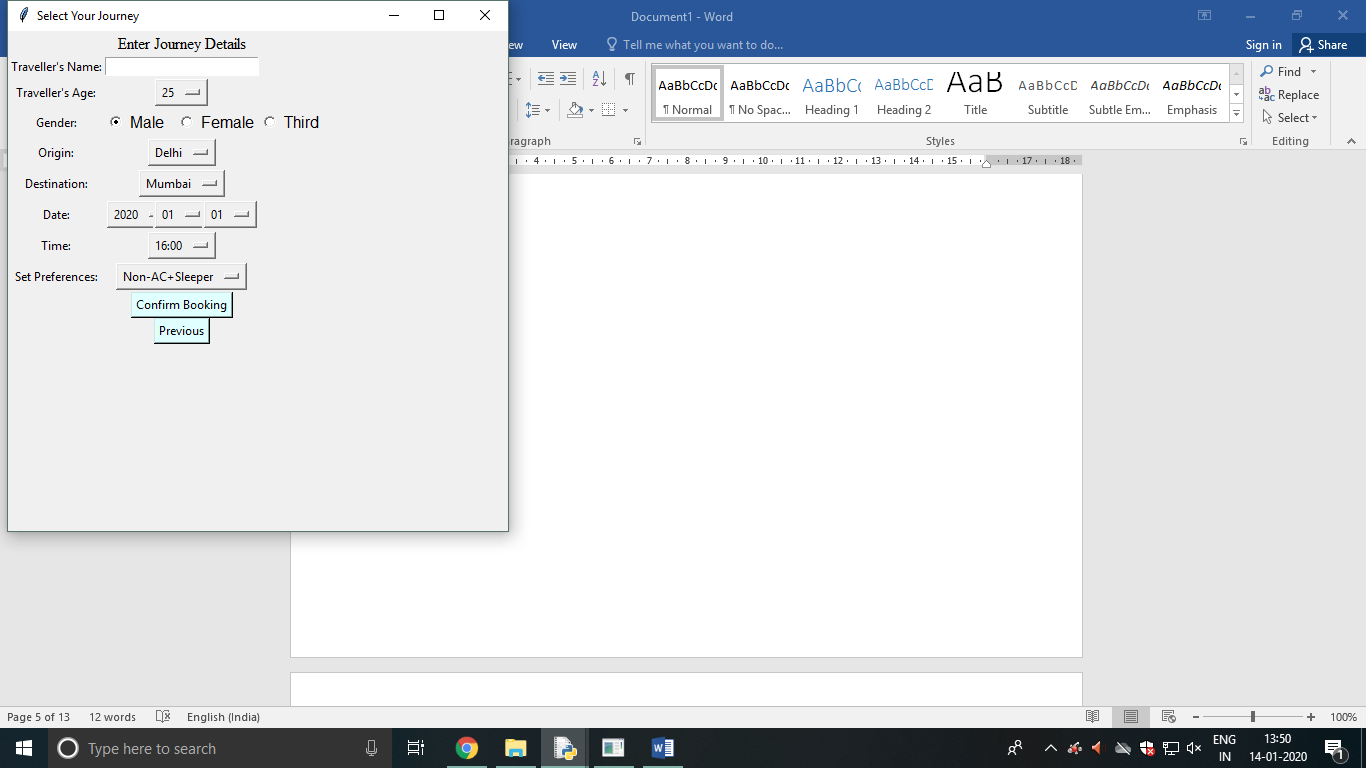
**Check booking**



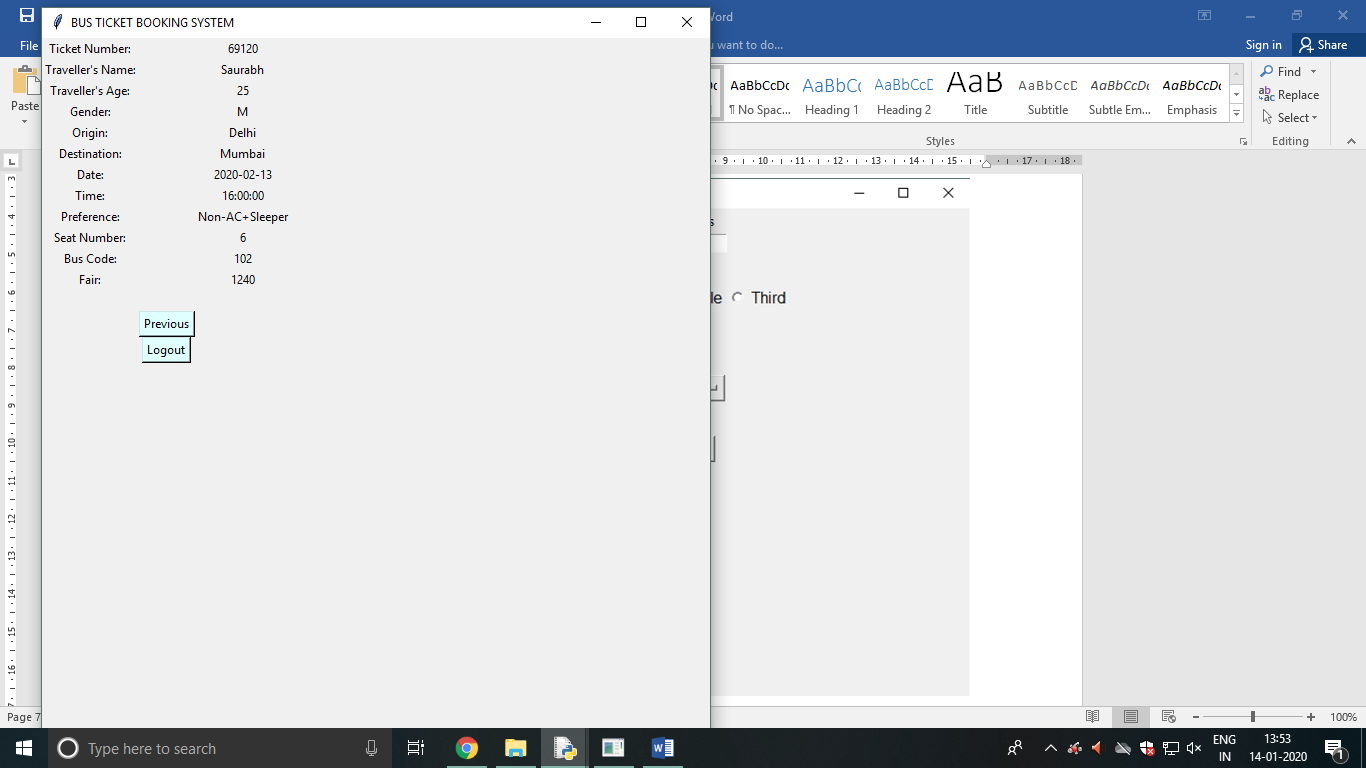
**BOOKING Details**



**Book ticket**



**Journey details**



**User Manual**

**How to install software:**

* **Hardware Requirements:**
* **Intel Pentium/Cleron or similar processor based pc at client/server END.**
* **128 MB RAM and 4GB HDD space for databases desirable.**
* **Standard I/O devices like keyboard and mouse etc.**
* **Printer is needed for hard copy reports.**
* **Local area network(LAN) is required for client server installation.**
* **SOFTWARE REQUIREMENT:**
* **Windows 2000/XP OS is desirable.**
* **Python IDLE should be installed**
* **Mysql version 6.1 with library Database must be present at machine.**

**REFERENCES**

**In order to work on this project titled- Bus Ticket Booking System, the above mentioned details are relevant to the subject. These are referred by me during the various phases of development of the project.**

**1-** **https://stackoverflow.com**

**2- https://www.tutorialspoint.com**

**3- Computer Science with Python-XII by Sumita Arora**

**4- Computer Science with Python-XI by Sumita Arora**

**5- https://www.learnpython.org**

**6- https://www.python.org**

**Other than the above-mentioned books and websites, the suggestions and supervision of my teacher and my class experience helped me a lot to develop this software project.**