	REGISTRATION NO.	COLLEGE
AAYUSH GADIA	100436	UNIVERSITY OF KALYANI
AVIK DUTTA	1001410899	TECHNO INDIA UNIVERSITY
MOINAK NANDI	304201500900605	UEM
SAYANTAN ROYCHOWDHURY	151040110481	IEM
SHU	151150110101	IMT
SUM		K
VIKA CHO		IMT

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

INDEX

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Prasun Neogy for his exemplary guidance, monitoring and constant encouragement throughout the course of the project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

PROJECT OBJECTIVE

Developing A Bus Ticket Reservation System with Predictive Analysis

The Primary Goals Consists of:

- Providing passengers the various details about bus service for their destination like available buses, fare, seat availability, time of journey and various other on-board facilities.
- The bus service ratings and hassle-free booking.
- Showing passengers the probable dates of ticket availability during peak times.

REQUIRED SPECIFICATION

The Required Specifications are:

- Python 3 and above
- SQLite

DATABASE DESIGN

DB Browser for SQLite - /home/sayantan/python-project/bus_db.db

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: fare_chart

New Record | Delete Record

	bus_id	source_time	stop_1_time	fare_1	stop_2_time	fare_2	estination_tim	fare_d
1	S1	06:30	07:30	200	NULL	NULL	08:30	300
2	AC1	06:00	07:00	300	NULL	NULL	08:00	450
3	SL1	20:00	21:00	270	NULL	NULL	22:00	430
4	S2	05:00	06:00	200	06:30	270	08:00	350
5	SL2	23:00	00:00 +1	270	00:30 +1	350	03:00 +1	450
6	S3	09:00	NULL	NULL	NULL	NULL	13:00	400
7	AC3	10:00	NULL	NULL	NULL	NULL	14:00	400
8	S4	09:30	11:00	250	NULL	NULL	13:30	450
9	SL4	19:30	21:00	300	NULL	NULL	23:30	470
10	S5	07:30	09:00	200	11:00	300	14:30	400
11	AC5	09:00	11:30	300	13:30	400	15:30	500
12	V5	17:00	18:00	350	19:30	490	20:00	570
13	S6	08:30	10:00	200	NULL	NULL	14:00	370
14	SL6	18:30	20:00	300	NULL	NULL	22:00	450
15	S7	09:30	11:00	250	NULL	NULL	15:00	300
16	V7	08:00	09:30	350	NULL	NULL	13:00	500
17	S8	07:00	NULL	NULL	NULL	NULL	13:00	330
18	AC8	11:00	NULL	NULL	NULL	NULL	16:30	450
19	SL8	21:00	NULL	NULL	NULL	NULL	02:00 +1	380
20	S9	07:30	09:30	300	NULL	NULL	12:00	400

1 - 21 of 24 | Go to: 1

DB Browser for SQLite - /home/sayantan/python-project/bus_db.db

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: reservation_table

New Record | Delete Record

	route_id	bus_id	username	starting	ending	date	seat_no	amount	ticket_no	reserved_on
1	MID1	AC5	mn	Kolaghat	Midnapore	23/06/2017	23	400	23062017A...	19/06/2017
2	MID1	AC5	mn	Kolaghat	Midnapore	02/07/2017	23	400	02072017A...	28/06/2017
3	MID1	AC5	mn	Kolaghat	Midnapore	10/07/2017	23	400	10072017A...	08/07/2017
4	AS1	SL1	ad	Kolkata	Bardhaman	25/07/2017	30	270	25072017SL...	17/07/2017
5	AS2	SL2	sr	Kolkata	Bardhaman	25/07/2017	30	270	25072017SL...	17/07/2017
6	AS2	SL2	ag	Kolkata	Bardhaman	25/07/2017	31	270	25072017SL...	17/07/2017
7	MID1	S5	src	Kolkata	Midnapore	25/07/2017	31	270	25072017S...	17/07/2017
8	MID1	AC5	src	Kolkata	Midnapore	25/08/2017	27	350	25082017A...	17/07/2017
9	MID1	S5	sr	Kolaghat	Kharagpur	25/08/2017	20	300	25082017S...	17/07/2017
10	MID1	S5	ag	Kolkata	Kolaghat	25/08/2017	20	250	25082017S...	17/07/2017
11	DUR2	V10	mn	Howrah	Durgapur	01/08/2017	1	500	01082017V...	17/07/2017
12	DUR2	V10	sr	Howrah	Durgapur	01/08/2017	2	480	01082017V...	17/07/2017
13	MID1	V5	du	Kolkata	Kolaghat	28/07/2017	1	350	28072017V...	19/07/2017
14	MID1	V5	du	Kolkata	Kolaghat	28/07/2017	2	350	28072017V...	19/07/2017
15	MID1	V5	du	Kolkata	Kolaghat	28/07/2017	3	350	28072017V...	19/07/2017
16	MID1	V5	du	Kolkata	Kolaghat	28/07/2017	4	350	28072017V...	19/07/2017
17	MID1	V5	du	Kolkata	Kolaghat	28/07/2017	5	350	28072017V...	19/07/2017
18	MID1	V5	du	Kolkata	Kolaghat	28/07/2017	7	350	28072017V...	19/07/2017
19	MID1	S5	du	Kolaghat	Midnapore	03/08/2017	1	200	03082017S...	19/07/2017
20	MID1	S5	du	Kolaghat	Midnapore	03/08/2017	2	200	03082017S...	19/07/2017

1 - 21 of 58 | Go to: 1

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

DB Browser for SQLite - /home/sayantan/python-project/bus_db.db

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: revenue_table

New Record | Delete Record

	mode	username	ticket_no	date	amount	count_or_pena
	Filter	Filter	Filter	Filter	Filter	Filter
1	reservation	mn	23062017A...	19/06/2017	400	0
2	reservation	mn	02072017A...	28/06/2017	400	0
3	reservation	mn	10072017A...	08/07/2017	400	0
4	reservation	mn	17072017A...	17/07/2017	400	0
5	reservation	mn	27072017A...	17/07/2017	400	0
6	cancellation	mn	17072017A...	17/07/2017	-400	0
7	cancellation	mn	27072017A...	17/07/2017	-400	0
8	reservation	mn	10072017SL...	10/07/2017	300	0
9	reservation	mn	15072017SL...	14/07/2017	300	0
10	reservation	mn	18072017SL...	17/07/2017	300	0
11	cancellation	mn	15072017SL...	17/07/2017	-300	0
12	cancellation	mn	10072017SL...	17/07/2017	-300	0
13	cancellation	mn	18072017SL...	17/07/2017	-300	0
14	reservation	ad	25072017SL...	17/07/2017	270	0
15	reservation	sr	25072017SL...	17/07/2017	270	0
16	reservation	ag	25072017SL...	17/07/2017	270	0
17	reservation	src	25072017S...	17/07/2017	270	0
18	reservation	src	25072017S...	17/07/2017	270	0
19	reservation	src	25072017S...	17/07/2017	270	0
20	cancellation	src	25072017S...	17/07/2017	-270	0

1 - 21 of 80 | Go to: 1

UTF-8

DB Browser for SQLite - /home/sayantan/python-project/bus_db.db

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: bus_table

New Record

	bus_id	route_id	type	total_seats
	Filter	Filter	Filter	Filter
1	S1	AS1	Ordinary	58
2	AC1	AS1	Air conditio...	45
3	SL1	AS1	Sleeper	40
4	S2	AS2	Ordinary	56
5	SL2	AS2	Sleeper	40
6	S3	ML1	Ordinary	40
7	AC3	ML1	Air conditio...	45
8	S4	ML2	Ordinary	58
9	SL4	ML2	Sleeper	45
10	S5	MID1	Ordinary	58
11	AC5	MID1	Air conditio...	45
12	V5	MID1	Volvo	40
13	S6	MID2	Ordinary	50
14	SL6	MID2	Sleeper	40
15	S7	HL1	Ordinary	56
16	V7	HL1	Volvo	40
17	S8	HL2	Ordinary	45
18	AC8	HL2	Air conditio...	40
19	SL8	HL2	Sleeper	40
20	S9	DUR1	Ordinary	58

1 - 21 of 24 | Go to: 1

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

DB Browser for SQLite - /home/sayantan/python-project/bus_db.db

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Modify Table | Delete Table

Name	Type	Schema
Tables (8)		
bus_table		CREATE TABLE bus_table(bus_id char(5) PRIMARY KEY, route_id char(5), type char(3), total_seats int(3))
cancellation_table		CREATE TABLE cancellation_table(cancellation_date text, username text, route_id char(5), bus_id char(5), starting text, ending text, reservation_
fare_chart		CREATE TABLE fare_chart(bus_id char(5) PRIMARY KEY, source_time text, stop_1_time text, fare_1 int(5), stop_2_time text, fare_2 int(5), destina
reservation_table		CREATE TABLE reservation_table(route_id char(5), bus_id char(5), username text, starting text, ending text, date text, seat_no int(3), amount int
revenue_table		CREATE TABLE revenue_table(mode , username text, ticket_no text, date text, amount int(5), discount_or_penalty)
route_table		CREATE TABLE route_table(route_id char(5) PRIMARY KEY, source text, stop_1 text, stop_2 text, destination)
user_activities		CREATE TABLE user_activities(username text PRIMARY KEY, time_tables text, buses_between_stops text, reservations text, cancellations text)
user_details		CREATE TABLE user_details(name text, username text PRIMARY KEY, type char(5), password text, security_ques text, security_answer text, payn
Indices (7)		
sqlite_autoindex_bus_ta...		
sqlite_autoindex_cancell...		
sqlite_autoindex_fare_c...		
sqlite_autoindex_reserv...		
sqlite_autoindex_route_...		
sqlite_autoindex_user_a...		
sqlite_autoindex_user_d...		
Views (0)		
Triggers (0)		

DB Browser for SQLite - /home/sayantan/python-project/bus_db.db

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: cancellation_table

	ıncellation_dal	username	route_id	bus_id	starting	ending	ɜservation_dat	seat_no	ticket_no	nount_forfeite
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	17/07/2017	mn	MID1	AC5	Kolaghat	Midnapore	17/07/2017	23	17072017A...	0
2	17/07/2017	mn	MID1	AC5	Kolaghat	Midnapore	27/07/2017	23	27072017A...	0
3	17/07/2017	mn	AS2	SL2	Kolkata	Durgapur	15/07/2017	24	15072017SL...	0
4	17/07/2017	mn	AS2	SL2	Kolkata	Durgapur	10/07/2017	24	10072017SL...	0
5	17/07/2017	mn	AS2	SL2	Kolkata	Durgapur	18/07/2017	24	18072017SL...	0
6	17/07/2017	src	MID1	S5	Kolkata	Midnapore	25/07/2017	32	25072017S...	0
7	17/07/2017	src	MID1	S5	Kolkata	Midnapore	25/07/2017	30	25072017S...	0
8	17/07/2017	src	DUR2	V10	Howrah	Durgapur	14/08/2017	35	14082017V...	50
9	19/07/2017	du	MID1	V5	Kolkata	Kolaghat	28/07/2017	6	28072017V...	0
10	19/07/2017	du	MID1	V5	Kolkata	Kolaghat	28/07/2017	8	28072017V...	0
11	19/07/2017	src	AS1	AC1	Kolkata	Asansol	23/07/2017	11	23072017A...	0

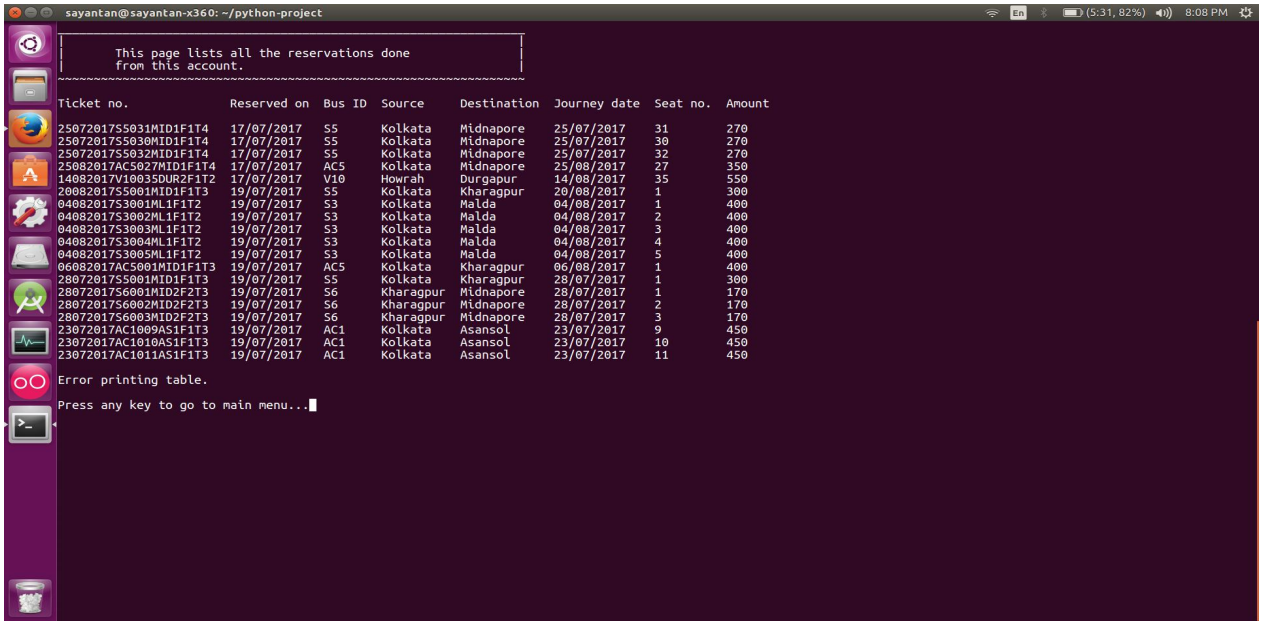
|< < 1 - 11 of 11 > >| Go to: 1

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

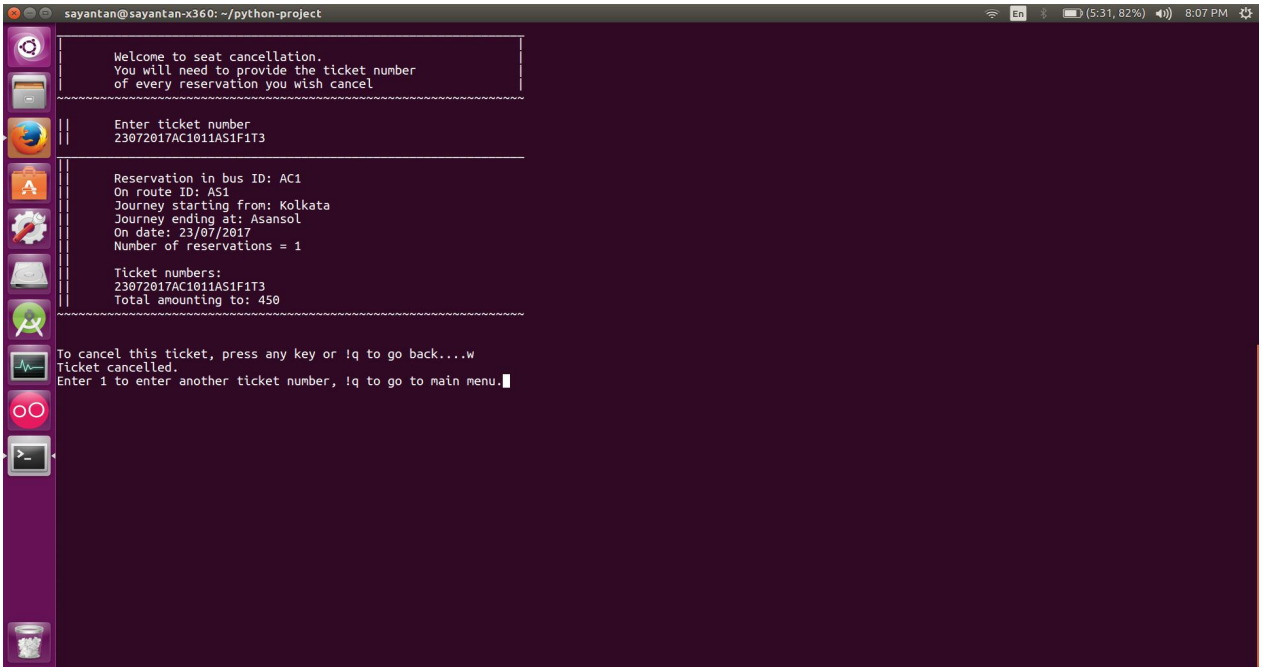
SCREENSHOTS

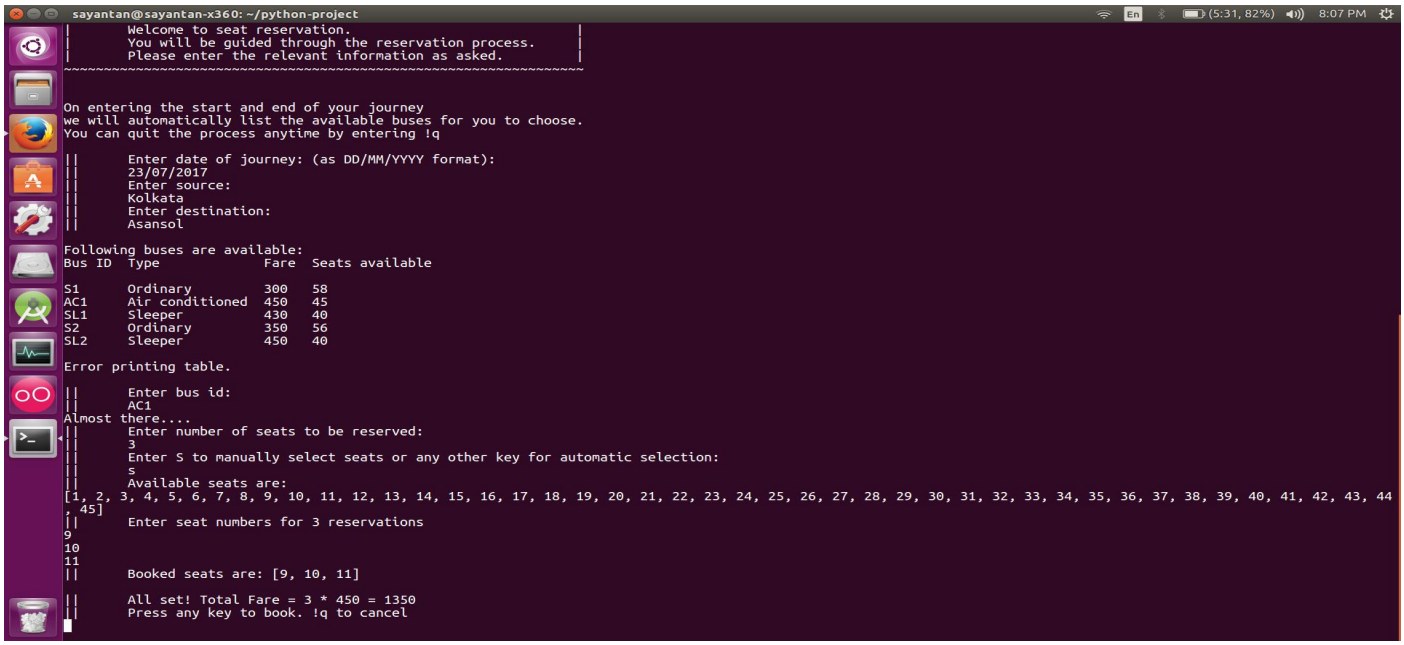


Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

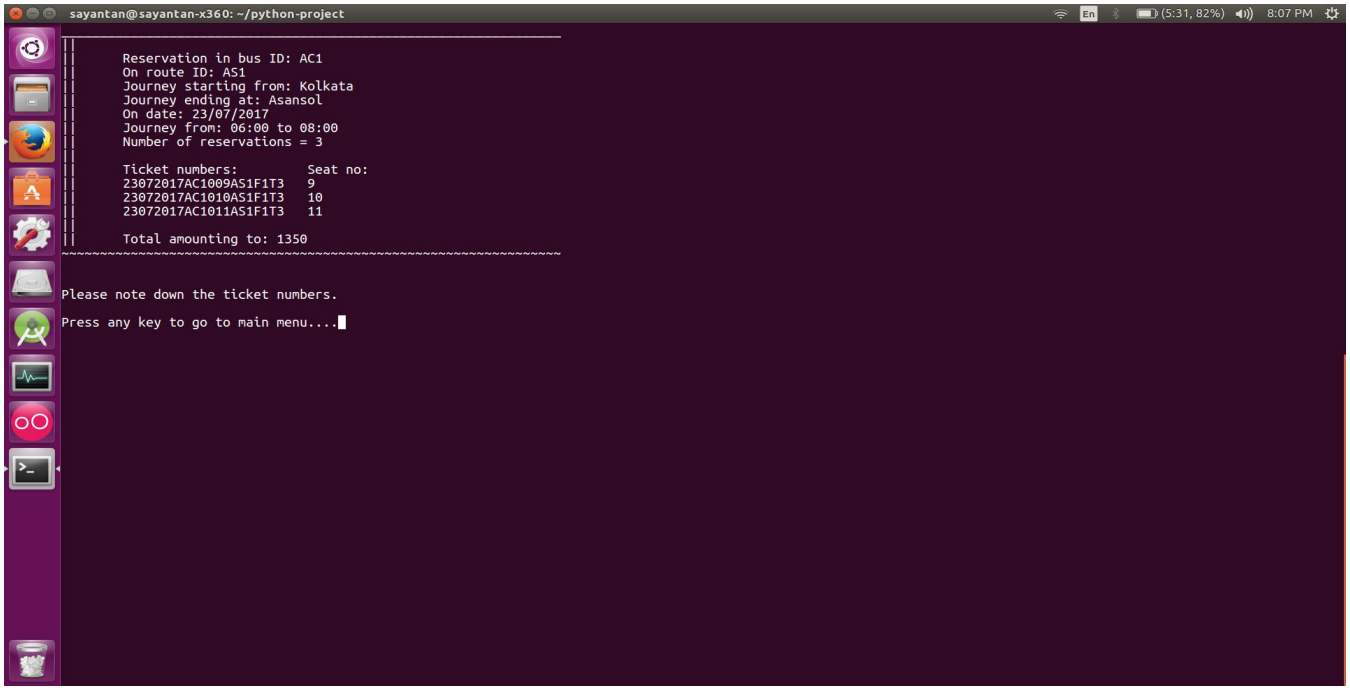




Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017



Document sign date :Jul 26, 2017

```
sayantan@sayantan-x360: ~/python-project
_____________________________________________________________
|                                                           |
|       Enter source, destination, date and bus ID of journey   |
|       to get available seats before planning reservation      |
|       Example: Howrah to Haldia on 20/08/2017 in AC8          |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


Enter !q to cancel anytime.

||      Enter source of journey:
||      Kolkata
||      Enter destination of journey:
||      Asansol
||      Enter date of journey: (as DD/MM/YYYY format):
||      23/07/2017

Following buses are available:
Bus ID  Seats available

S1      58
AC1     45
SL1     40
S2      56
SL2     40

Error printing table.

||      Enter bus ID:
||      AC1
||      Available seats are:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45]
||      Enter 1 to see more seat availabilities or !q for main menu.
```

```
sayantan@sayantan-x360: ~/python-project
_____________________________________________________________
|                                                           |
|       Welcome                                             |
|                                                           |
|       Please select the appropriate option:               |
|       1) Bus time table                                   |
|       2) Buses between stops                              |
|       3) Seat availability                                |
|                                                           |
|       Login and get access to                             |
|       Seat booking                                        |
|       and much more...                                    |
|                                                           |
|       L) LOGIN                                            |
|       Not registered yet? Getting an account is easy      |
|       S) SIGN UP >>>>                                     |
|                                                           |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter option (1,2,3,L,S):
```

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017

CODE

Input Output

```
'''
Use this file for basic input and output.
Do not execute this file. Import this as any other python module. Call the
appropriate methods for related jobs.

Syntax:
import ipop

Keep this file in the same directory as the main executable.
'''

from ast import literal_eval
import os

# getUserData(data_types, request_message = '', error_message = None, retry =
True, validation = None, quittable = True) -> returns a variable
'''
This function is an alternative to the native input() method.
It combines the features of try, except in case of invalid data, and also a
validation expression.

data_types []    :          allowed type of data requested from the user
      {list of data types: int|float|bool|str}
request_message {str}:           (optional) message shown to user before input
error_message {str}:        (optional) error message to be shown to user in case
of invalid data type entered
retry {bool}:               (optional) keep trying until a valid data is entered
           {True|False}
validation []:              (optional) additional expression (written with
respect to variable 'x') to be evaluated for valid data. Write in same prder of
data_types[]
quittable {bool}:           (optional) show an option to cancel the input or
abort the program           {True|False}

      Example 1: we want to get a float data

            import ipop
            my_float_data = ipop.getUserData([float], "Enter a floating data",
"Wrong data entered!")

      Example 2: we want to get an integer greater than 25

            import ipop
            my_int_data = ipop.getUserData([int], "enter an int greater than 25",
"wrong data", True, ['x > 25'] )
```

Example 3: we want to get an int or float, if integer is entered, it should be less than 5

```
import ipop
my_data = ipop.getUserData([float, int], "enter an integer less than 5 or a float", "wrong input", True, ['True', 'x < 5'])
```

*** note that since float is the first of 'data_types', and any float can be accepted, the first 'validation' is 'True'.
*** similarly we give the validation expression of integer data type as it is next to float in 'data_types'

Example 4: we want to get any integer OR a float greater than 100 OR a string starting with 'hello'

```
import ipop
my_data = ipop.getUserData([str, float, int], "enter data", "wrong data", True, ['x.startswith("hello")', 'x > 100', 'True'])
```

```
'''
def getUserData(data_types, request_message = '', error_message = None, retry = True, validation = None, quittable = True):
	r = True
	invalid_flag = 1
	try:
		t = data_types[0]
	except:
		print ("Please enter a data_type in square braces! example: [int]")
		return None

	while r and invalid_flag == 1:

		r = retry

		print (request_message, end='')

		x = input()

		if quittable:
			if x == '!q':
				print ("Input cancelled\n")
				return None
			elif x == '!!q':
				print ("Program aborted!\n")
				exit()

		if x != '':
			try:
				x = literal_eval(x)
			except:
				pass


			for i in range(len(data_types)):
				dt = data_types[i]
				if type(x) == dt:
					try:
						op = validation[i]
```

```
                                    invalid_flag = 0
                                    break
                              try:
                                    if eval(op):
                                          invalid_flag = 0
                                          break
                              except:
                                    print ("Validation error. Please properly
check the input expression.\n")
                                    return None

            if invalid_flag == 1:
                  if error_message != None:
                        print(error_message)

      if invalid_flag == 0:
            return x
      else:
            return None

# cls():
'''
This method clears the screen
'''
def cls():
      _ = os.system("clear")

def print_table(headers_and_content):

      #try:
            headers = headers_and_content[0]
            content = headers_and_content[1]

            fdigits = []

            for h in headers:
                  fdigits.append(len(h))

            for i in range(len(headers)):
                  for line in content:
                        l = len(str(line[i]))
                        if l > fdigits[i]: fdigits[i] = l

            for i in range(len(fdigits)):
                  fdigits[i] = fdigits[i] + 2

            for i in range(len(headers)):
                  sf = '{:' + str(fdigits[i]) + '}'
                  print(sf.format(headers[i]), end='')

            print('\n')

            for c in content:
                  for i in range(len(fdigits)):
                        sf = '{:' + str(fdigits[i]) + '}'
                        print(sf.format(str(c[i])), end='')
                  print()
            print()
```

```
	#except:
		print('Error printing table.')
```

Calculation of date

```
import time
import datetime

def isValidTransactionDate(date):

	nowstring = time.strftime('%Y %m %d')
	nowstring = nowstring.split(' ')
	dtnow = datetime.datetime(int(nowstring[0]), int(nowstring[1]),
int(nowstring[2]))

	try:
		datelist = date.split('/')
		dtdate = datetime.datetime(int(datelist[2]), int(datelist[1]),
int(datelist[0]))
	except: dtdate = None

	if len(date) == 10 and dtdate != None and dtdate >= dtnow: return True
	else: return False

def isPreviousDate(date):

	nowstring = time.strftime('%Y %m %d')
	nowstring = nowstring.split(' ')
	dtnow = datetime.datetime(int(nowstring[0]), int(nowstring[1]),
int(nowstring[2]))

	try:
		datelist = date.split('/')
		dtdate = datetime.datetime(int(datelist[2]), int(datelist[1]),
int(datelist[0]))
	except: dtdate = None

	if len(date) == 10 and dtdate != None and dtdate <= dtnow: return True
	else: return False

def compareDates(date1, date2):

	try:
		dt1list = date1.split('/')
		dt2list = date2.split('/')
		dtdate1 = datetime.datetime(int(dt1list[2]), int(dt1list[1]),
int(dt1list[0]))
		dtdate2 = datetime.datetime(int(dt2list[2]), int(dt2list[1]),
int(dt2list[0]))
	except: return False

	if len(date1) == len(date2) == 10 and dtdate1 <= dtdate2: return True
	else: return False
```

Managing Database

```
import sqlite3
```

```
import os
import time
import datetime

import calc
import ipop

# init() -> returns an int
'''
initialises the cursor and connection. Returns:
0: database file was present and not remade. connection is successfully opened.
1: database was made connection successfully opened.
2: database could not be opened.
'''
def init():
      global conn
      global curs
      global dbname
      dbname = 'bus_db.db'
      r = 2
      #checks if the file is already present
      if os.path.isfile(dbname):
            r = 0
      try:
            conn = sqlite3.connect(dbname)
            curs = conn.cursor()
            r = 1 if r != 0 else 0
      except:
            r = 2
      return r


# show_table_names()
'''
Returns names of all the tables. For debugging purpose.
'''
def get_table_names():
      table_names = None
      if init() != 2:
            try:
                  tblcmd = "SELECT name FROM sqlite_master WHERE type='table'"
                  curs.execute(tblcmd)
                  table_names = curs.fetchall()
            except:
                  table_names = 2

      return table_names

# get_table(table_name)
'''
Returns contents of the table with name table_name. Headers and content are
sent separately.

Example:

mdb.get_table('route_table')

(['route_id', 'source', 'stop_1', 'stop_2', 'destination'], [('AS1', 'Kolkata',
'Bardhaman', '', [illegible] man', 'Durgapur',
```

```
'Asansol'), ('ML1', 'Kolkata', '', '', 'Malda'), ('ML2', 'Kolkata', 'Bardhaman',
'', 'Malda'), ('MID1', 'Kolkata', 'Kolaghat', 'Kharagpur', 'Midnapore'),
('MID2', 'Kolkata', 'Kharagpur', '', 'Midnapore'), ('HL1', 'Howrah', 'Kolaghat',
'', 'Haldia'), ('HL2', 'Howrah', '', '', 'Haldia'), ('DUR1', 'Howrah',
'Bardhaman', '', 'Durgapur'), ('DUR2', 'Howrah', '', '', 'Durgapur')])

'''
def get_table(table_name):

      data = []
      headers = []
      if init() != 2:
            try:
                  tblcmd = "SELECT * FROM " + table_name
                  curs.execute(tblcmd)
                  data = curs.fetchall()
                  headers = [data[0] for data in curs.description]
            except: pass
      return headers, data



# isTablePresent(table_name {str}) -> returns bool
'''
Returns True if table with table_name is present, else False. For debugging
purpose and internal use.
'''
def isTablePresent(table_name):
      presence = False
      if init() != 2:
            try:
                  tblcount = "SELECT count(*) FROM sqlite_master WHERE
type='table' AND name='" + table_name + "'"
                  curs.execute(tblcount)
                  c = curs.fetchall()
                  if c != [(0,)]:
                        presence = True
            except:
                  pass
      return presence

# create_route_table(n {0|1}, table_name {str}) -> returns an int
'''
Creates the bus_table. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made

n : pass 1 to recreate table
'''
def create_bus_table(n = 0, table_name = 'bus_table'):
      r = 2
      if isTablePresent(table_name) == False or n == 1:

            try:

                  # delete old table if user selects to recreate data
                  tbldelete = "DROP TABLE IF EXISTS " + table_name
                  c
```

```
                # create table
                tblcreate = "CREATE TABLE " + table_name + "(bus_id char(5)
PRIMARY KEY, route_id char(5), type char(3), total_seats int(3))"
                curs.execute(tblcreate)


                #insert records
                tblins = "INSERT INTO " + table_name + " values(?, ?, ?, ?)"

                curs.execute(tblins, ('S1', 'AS1', 'Ordinary', 58))
                curs.execute(tblins, ('AC1', 'AS1', 'Air conditioned', 45))
                curs.execute(tblins, ('SL1', 'AS1', 'Sleeper', 40))
                curs.execute(tblins, ('S2', 'AS2', 'Ordinary', 56))
                curs.execute(tblins, ('SL2', 'AS2', 'Sleeper', 40))
                curs.execute(tblins, ('S3', 'ML1', 'Ordinary', 40))
                curs.execute(tblins, ('AC3', 'ML1', 'Air conditioned', 45))
                curs.execute(tblins, ('S4', 'ML2', 'Ordinary', 58))
                curs.execute(tblins, ('SL4', 'ML2', 'Sleeper', 45))
                curs.execute(tblins, ('S5', 'MID1', 'Ordinary', 58))
                curs.execute(tblins, ('AC5', 'MID1', 'Air conditioned', 45))
                curs.execute(tblins, ('V5', 'MID1', 'Volvo', 40))
                curs.execute(tblins, ('S6', 'MID2', 'Ordinary', 50))
                curs.execute(tblins, ('SL6', 'MID2', 'Sleeper', 40))
                curs.execute(tblins, ('S7', 'HL1', 'Ordinary', 56))
                curs.execute(tblins, ('V7', 'HL1', 'Volvo', 40))
                curs.execute(tblins, ('S8', 'HL2', 'Ordinary', 45))
                curs.execute(tblins, ('AC8', 'HL2', 'Air conditioned', 40))
                curs.execute(tblins, ('SL8', 'HL2', 'Sleeper', 40))
                curs.execute(tblins, ('S9', 'DUR1', 'Ordinary', 58))
                curs.execute(tblins, ('SL9', 'DUR1', 'Sleeper', 50))
                curs.execute(tblins, ('S10', 'DUR2', 'Ordinary', 45))
                curs.execute(tblins, ('SL10', 'DUR2', 'Sleeper', 40))
                curs.execute(tblins, ('V10', 'DUR2', 'Volvo', 40))
                conn.commit()
                r = 1
            except:
                r = 2
        else:
            r = 0
        return r, table_name

# create_route_table(n {0|1}, table_name {str}) -> returns an int

'''
Creates the route_table. Returns:
        0 if table was present
        1 if table was not present and created
        2 if table could not be made

n : pass 1 to recreate table
'''
def create_route_table(n = 0, table_name = 'route_table'):
        r = 2
        if isTablePresent(table_name) == False or n == 1:

            try:

                # [illegible] recreate data
```

```
                tbldelete = "DROP TABLE IF EXISTS " + table_name
                curs.execute(tbldelete)

                # create table
                tblcreate = "CREATE TABLE " + table_name + "(route_id char(5)
PRIMARY KEY, source text, stop_1 text, stop_2 text, destination)"
                curs.execute(tblcreate)


                #insert records
                tblins = "INSERT INTO " + table_name + "
values(?, ?, ?, ?, ?)"

                curs.execute(tblins, ('AS1', 'Kolkata', 'Bardhaman', '',
'Asansol'))
                curs.execute(tblins, ('AS2', 'Kolkata', 'Bardhaman',
'Durgapur', 'Asansol'))
                curs.execute(tblins, ('ML1', 'Kolkata', '', '', 'Malda'))
                curs.execute(tblins, ('ML2', 'Kolkata', 'Bardhaman', '',
'Malda'))
                curs.execute(tblins, ('MID1', 'Kolkata', 'Kolaghat',
'Kharagpur', 'Midnapore'))
                curs.execute(tblins, ('MID2', 'Kolkata', 'Kharagpur', '',
'Midnapore'))
                curs.execute(tblins, ('HL1', 'Howrah', 'Kolaghat', '',
'Haldia'))
                curs.execute(tblins, ('HL2', 'Howrah', '', '', 'Haldia'))
                curs.execute(tblins, ('DUR1', 'Howrah', 'Bardhaman', '',
'Durgapur'))
                curs.execute(tblins, ('DUR2', 'Howrah', '', '', 'Durgapur'))
                conn.commit()
                r = 1
           except:
                r = 2
      else:
           r = 0
      return r, table_name


# create_fare_chart(n {0|1}, table_name {str}) -> returns an int

'''
Creates the fare chart + time table. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made

n : pass 1 to recreate table
'''
def create_fare_chart(n = 0, table_name = 'fare_chart'):
      r = 2
      if isTablePresent(table_name) == False or n == 1:

           try:

                # delete old table if user selects to recreate data
                tbldelete = "DROP TABLE IF EXISTS " + table_name
                curs.execute(tbldelete)
```

```
                # create table
                tblcreate = "CREATE TABLE " + table_name + "(bus_id char(5)
PRIMARY KEY, source_time text, stop_1_time text, fare_1 int(5), stop_2_time
text, fare_2 int(5), destination_time text, fare_d int(5))"
                curs.execute(tblcreate)


                #insert records
                tblins = "INSERT INTO " + table_name + "
values(?, ?, ?, ?, ?, ?, ?, ?)"

                # fares in order: Ordinary, AC, Sleeper, Volvo. Example: for
AS1,
                curs.execute(tblins, ('S1', '06:30', '07:30', 200, None, None,
'08:30', 300))
                curs.execute(tblins, ('AC1', '06:00', '07:00', 300, None, None,
'08:00', 450))
                curs.execute(tblins, ('SL1', '20:00', '21:00', 270, None, None,
'22:00', 430))
                curs.execute(tblins, ('S2', '05:00', '06:00', 200, '06:30',
270, '08:00', 350))
                curs.execute(tblins, ('SL2', '23:00', '00:00 +1', 270, '00:30
+1', 350, '03:00 +1', 450))
                curs.execute(tblins, ('S3', '09:00', None, None, None, None,
'13:00', 400))
                curs.execute(tblins, ('AC3', '10:00', None, None, None, None,
'14:00', 400))
                curs.execute(tblins, ('S4', '09:30', '11:00', 250, None, None,
'13:30', 450))
                curs.execute(tblins, ('SL4', '19:30', '21:00', 300, None, None,
'23:30', 470))
                curs.execute(tblins, ('S5', '07:30', '09:00', 200, '11:00',
300, '14:30', 400))
                curs.execute(tblins, ('AC5', '09:00', '11:30', 300, '13:30',
400, '15:30', 500))
                curs.execute(tblins, ('V5', '17:00', '18:00', 350, '19:30',
490, '20:00', 570))
                curs.execute(tblins, ('S6', '08:30', '10:00', 200, None, None,
'14:00', 370))
                curs.execute(tblins, ('SL6', '18:30', '20:00', 300, None, None,
'22:00', 450))
                curs.execute(tblins, ('S7', '09:30', '11:00', 250, None, None,
'15:00', 300))
                curs.execute(tblins, ('V7', '08:00', '09:30', 350, None, None,
'13:00', 500))
                curs.execute(tblins, ('S8', '07:00', None, None, None, None,
'13:00', 330))
                curs.execute(tblins, ('AC8', '11:00', None, None, None, None,
'16:30', 450))
                curs.execute(tblins, ('SL8', '21:00', None, None, None, None,
'02:00 +1', 380))
                curs.execute(tblins, ('S9', '07:30', '09:30', 300, None, None,
'12:00', 400))
                curs.execute(tblins, ('SL9', '19:30', '21:30', 370, None, None,
'00:30 +1', 450))
                curs.execute(tblins, ('S10', '06:00', None, None, None, None,
'10:30', 430))
                curs.execute(tblins, ('SL10', '00:20', None, None, None, None,
'04:00', 480))
```

```
                curs.execute(tblins, ('V10', '12:00', None, None, None, None,
'15:45', 500))

                conn.commit()
                r = 1

          except:
                r = 2
     else:
          r = 0
     return r, table_name



# validate_route(route_id {str}, starting {str}, ending {str}) -> returns tuple
'''
Checks if starting and ending positions are feasible for a route.
If present: returns a tuple of (beginning stop index, ending stop index)
If not found: returns None

Example:

mdb.validate_route('MID1', 'Kolaghat', 'Kharagpur')
(1, 2)

mdb.validate_route('MID1', 'Kolaghat', 'Malda') -> returns None
'''
def validate_route(route_id, starting, ending):

     r = None

     r1, rtn = create_route_table(0)

     if r1 != 2:

          # get all available route_id
          curs.execute("SELECT route_id FROM " + rtn)
          routes = curs.fetchall()
          rt = (route_id,)

          if rt in routes:

                # get the source, stops and destination of the selected
route_id
                curs.execute("SELECT source, stop_1, stop_2, destination FROM
" + rtn + " WHERE route_id=" + "'" + route_id + "'")
                places = curs.fetchall()
                places = places[0]    #fetchall() returns a list with only one
tuple element. This line extracts that tuple.

                # Remove all None from route
                places = [i for i in places if i != '']

                if starting in places and ending in places:

                     # both starting and ending must be present in the
route_id
```

```
                        e = places.index(ending)

                        if s < e:

                              # also starting should be before ending
                              r = (s, e)

      return r

# getRouteFromBusID(bus_id {str}) -> returns str
'''
Takes a bus_id and returns its route_id.

Example:

mdb.getRouteFromBusID('AC8')
'HL2'
'''
def getRouteFromBusID(bus_id):

      r = ''
      r1, rtn = create_bus_table(0)
      if r1 != 2:
            try:
                  curs.execute("SELECT route_id FROM " + rtn + " WHERE bus_id='"
+ bus_id + "'")
                  rids = curs.fetchall()
                  if rids != []:
                        r = rids[0][0]
            except: pass

      return r

# getFare(bus_id {str}, source {str}, destination {str}) -> returns int
'''
Takes a bus_id and returns journey fare from source to destination.

Example:

mdb.getFare('S5', 'Kolkata', 'Kharagpur')
300
'''
def getFare(bus_id, source, destination):

      fare = 0
      r1, table_name = create_fare_chart(0)
      route_id = getRouteFromBusID(bus_id)

      if route_id != 0:
            t = validate_route(route_id, source, destination)#index of source
and destination as in route_table
            if t != None:
                  try:
                        s = t[0]
                        e = t[1]

                        tblfares = "SELECT fare_1, fare_2, fare_d FROM " +
table_name + " WHERE bus_id='" + bus_id + "'"
```

```
                        fares = curs.fetchall()
                        fares = [i for i in fares[0] if i != None]
                        fares = [0] + fares   # source has zero fare
                        fare = fares[e] - fares[s]  #fare calculated by
subtracting starting from ending
                  except:
                        fare = ''

      return fare

# getTime(bus_id {str}, source {str}, destination {str}) -> returns str
'''
Takes a bus_id and returns journey time from source to destination.

Example:

mdb.getTime('S5', 'Kolkata', 'Kharagpur')
'03:30'
'''
def getTime(bus_id, source, destination):

      time = 0
      r1, table_name = create_fare_chart(0)
      route_id = getRouteFromBusID(bus_id)

      if route_id != 0:
            t = validate_route(route_id, source, destination)#index of source
and destination as in route_table
            if t != None:
                  try:
                        s = t[0]
                        e = t[1]

                        tbltimes = "SELECT source_time, stop_1_time, stop_2_time,
destination_time FROM " + table_name + " WHERE bus_id='" + bus_id + "'"
                        curs.execute(tbltimes)
                        times = curs.fetchall()
                        times = [i for i in times[0] if i != None]
                        ts, te = times[s], times[e]
                        # operations performed if journey extends next day:
example: ts = '23:00', te = '02:30 +1'
                        te = te.split(' +')                    # te = ['02:30',
'1']
                        if len(te) == 1: te = te[0]
                        elif len(te) == 2:
                              te = str(int(te[0].split(':')[0]) + 24*int(te[1])) +
':' + te[0].split(':')[1]   # te = (02 + 24*1):(30) = '26:30'
                        ts = int(ts.split(':')[0])*60 + int(ts.split(':')[1])  #
ts = '23:00' = 23*60 + 30
                        te = int(te.split(':')[0])*60 + int(te.split(':')[1])  #
te = '26:30' = 26*60 + 30
                        td = te - ts
                        time = '{:02d}'.format(int(td/60)) + ':' +
'{:02d}'.format(int(td%60))# converting to hours and minutes
                  except:
                        time = ''

      return time
```

```
# getBusType(bus_id {str}) -> returns str
'''
Takes a bus_id and returns its type.

Example:

mdb.getBusType('V10')
'Volvo'
'''
def getBusType(bus_id):

      btype = None
      r1, table_name = create_bus_table(0)
      if r1 != 2:
            try:
                  curs.execute("SELECT type FROM " + table_name + " WHERE
bus_id='" + bus_id + "'")
                  btype = curs.fetchall()
                  btype = btype[0][0] if btype != [] else None
            except: pass
      return btype


# create_revenue_table(n {0|1}, table_name {str}) -> returns an int
'''
Creates the frevenue_table. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made

n : pass 1 to recreate table
'''
def create_revenue_table(n = 0, table_name = 'revenue_table'):
      r = 2
      if isTablePresent(table_name) == False or n == 1:
            try:

                  # delete old table if user selects to recreate data
                  tbldelete = "DROP TABLE IF EXISTS " + table_name
                  curs.execute(tbldelete)

                  # create table
                  tblcreate = "CREATE TABLE " + table_name + "(mode , username
text, ticket_no text, date text, amount int(5), discount_or_penalty)"
                  curs.execute(tblcreate)
                  conn.commit()
                  r = 1
            except:
                  r = 2
      else:
            r = 0
      return r, table_name

# add_revenue(mode {'reservation'|'cancellation'}, username {str}, ticket_no
{str}, amount {int}, discount_or_penalty {int}) -> returns bool
'''
Used to add revenue to revenue_table. Returns True if successfully recorded
else False. For internal use.
'''
```

```
def add_revenue(mode, username, ticket_no, amount, discount_or_penalty = 0):

      success = False
      r1, revenueTName = create_revenue_table(0)
      if r1 != 2:
            try:
                  tblins = "INSERT INTO '" + revenueTName + "'
values(?, ?, ?, ?, ?, ?)"
                  curs.execute(tblins, (mode, username, ticket_no,
time.strftime('%d/%m/%Y'), amount, discount_or_penalty))
                  conn.commit()
                  success = True
            except:
                  success = False

      return success

# create_reservation_table(n {0|1}, table_name {str}) -> returns an int

'''
Creates the reservation_table. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made
n : pass 1 to recreate table
'''
def create_reservation_table(n = 0, table_name = 'reservation_table'):
      r = 2
      if isTablePresent(table_name) == False or n == 1:

            try:

                  # delete old table if user selects to recreate data
                  tbldelete = "DROP TABLE IF EXISTS " + table_name
                  curs.execute(tbldelete)

                  # create table
                  tblcreate = "CREATE TABLE " + table_name + "(route_id char(5),
bus_id char(5), username text, starting text, ending text, date text, seat_no
int(3), amount int(5), ticket_no text PRIMARY KEY, reserved_on text)"
                  curs.execute(tblcreate)
                  conn.commit()
                  r = 1
            except:
                  r = 2
      else:
            r = 0

      return r, table_name


# makeTicket (bus_id {str}, starting {str}, ending {str}, date {str}, seat_no
{int}) -> returns ticket number as string
'''
This method combines the inputs and route_id from given bus_id and provides a
ticket number.
'''
def makeTicket(bus_id, starting, ending, date, seat_no):
```

```
      route_id = getRouteFromBusID(bus_id)
      ticket_no = 0

      if route_id != '':

            indices = validate_route(route_id, starting, ending)   # get
starting and ending indices
            if indices != None:

                  d = ''.join(date.split('/'))     # 13/07/2017 will be formatted
to 13072017

                  # processing ticket
                  ticket_no = d + bus_id + '{:03d}'.format(seat_no) + route_id +
'F' + str(indices[0]+1) + 'T' + str(indices[1]+1)

      return ticket_no

# isReservationPossible(bus_id {str}, starting {str}, ending {str}, date {str},
seat_no {int}) -> returns bool
'''
This method returns if a requested reservation overlaps with a previous
reservation

Example: assume seat 20 is booked in S5 from Kolaghat to Kharagpur on
25/08/2017

mdb.isReservationPossible('S5', 'Kolkata', 'Midnapore', '25/08/2017', 20)
False

mdb.isReservationPossible('S5', 'Kolkata', 'Kolaghat', '25/08/2017', 20)
True

mdb.isReservationPossible('S5', 'Kolkata', 'Kolaghat', '25/08/2017', 110)    #
invalid seat
False
'''
def isReservationPossible(bus_id, starting, ending, date, seat_no):

      possibility = False
      route_id = getRouteFromBusID(bus_id)
      r1, reserveTName = create_reservation_table(0)

      if r1 != 2 and route_id != '' :

            r2, bus_table_name = create_bus_table(0)
            curs.execute("SELECT total_seats FROM " + bus_table_name + " WHERE
bus_id='" + bus_id + "'")
            total_seats = curs.fetchall()
            total_seats = total_seats[0][0]

            indices = validate_route(route_id, starting, ending)

            if indices != None and calc.isValidTransactionDate(date) and 0 <
seat_no <= total_seats:

                  try:
```

```
                        # getting probable clashable routes
                        tblcmd = "SELECT starting, ending FROM '" + reserveTName
+ "' WHERE bus_id='" + bus_id + "' AND seat_no='" + str(seat_no) + "' AND
date='" + date + "'"
                        curs.execute(tblcmd)
                        similarReservations = curs.fetchall()

                        currentStartingIndex = indices[0]
                        currentEndingIndex = indices[1]
                        currentStops = set(range(currentStartingIndex,
currentEndingIndex))  # a set is made with the range from starting index to
ending index

                        for similar in similarReservations:

                              i = validate_route(route_id, similar[0], similar[1])

                              similarStartingIndex = i[0]
                              similarEndingIndex = i[1]
                              similarStops = set(range(similarStartingIndex,
similarEndingIndex))  # a set is made for all similar reservations

                              # comparing the two sets. If no common is found,
then reservation is possible
                              if len(currentStops & similarStops) != 0:
                                    possibility = False
                                    break

                  except:
                        pass

      return possibility


# add_reservation(bus_id {str}, username {str}, starting {str}, ending {str},
date {str}, seat_no {int}, amount {int}) -> returns ticket number

'''
Used to add reservation records to reservation_table. Returns:
      ticket_no if reservation was added
      0 if reservation could not be added

Example:

mdb.add_reservation('S5', 'ag', 'Kolkata', 'Kolaghat', '25/08/2017', 20, 250)
'25082017S5020MID1F1T2'

mdb.add_reservation('S5', 'ag', 'Kolkata', 'Kolaghat', '25/08/2015', 20, 250)
      # invalid date
0

mdb.add_reservation('S5', 'ag', 'Kolkata', 'Delhi', '25/08/2017', 20, 250)
      #invalid route
0

mdb.add_reservation('S5', 'abc', 'Kolkata', 'Kolaghat', '25/08/2017', 20, 250)
      #username not registered
0
```

```
mdb.add_reservation('S5', 'ag', 'Kolkata', 'Kolaghat', '25/08/2017', 110, 250)
      #seat number not present
0
'''
def add_reservation(bus_id, username, starting, ending, date, seat_no, amount):
      ticket_no = 0
      r1, table_name = create_reservation_table(0)
      r2, cancelTName = create_cancellation_table(0)
      r3, user_activities_table = create_user_activities_table(0)
      route_id = getRouteFromBusID(bus_id)

      if r1 != 2 and r2 != 2 and r3 != 2 and route_id != '' and
checkUsernamePresence(username, user_activities_table) and
isReservationPossible(bus_id, starting, ending, date, seat_no):

            ticket_no = makeTicket(bus_id, starting, ending, date, seat_no)
            if ticket_no != 0:
                  try:

                        tblins = "INSERT INTO " + table_name + "
values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
                        curs.execute(tblins, (route_id, bus_id, username,
starting, ending, date, seat_no, amount, ticket_no, time.strftime("%d/%m/%Y")))

                        # delete from cancellation_table
                        try: curs.execute("DELETE FROM " + cancelTName + " WHERE
ticket_no='" +  ticket_no + "'")
                        except: pass

                        conn.commit()

                        # add to revenue
                        if add_revenue('reservation', username, ticket_no, amount,
(getFare(bus_id, starting, ending) - amount)) == False:
                              print ('Error adding revenue.')

                        # add to user_activities
                        reservation_string = (ticket_no + '_' +
time.strftime("%d/%m/%Y") + '_' + bus_id + '_' + starting + '_' + ending + '_'
+ date + '_' + str(seat_no) + '_' + str(amount))
                        if change_user_activity(username, 'reservations',
reservation_string, 1) != 1:
                              print ('Error adding to user_activities.')

                  except: ticket_no = 0
      return ticket_no

# create_cancellation_table(n {0|1}, table_name {str}) -> returns an int

'''
Creates the cancellation_table. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made
n : pass 1 to recreate table
'''
def create_cancellation_table(n = 0, table_name = 'cancellation_table'):
      r = 2
```

```
	if isTablePresent(table_name) == False or n == 1:

		try:

			# delete old table if user selects to recreate data
			tbldelete = "DROP TABLE IF EXISTS " + table_name
			curs.execute(tbldelete)


			# create table
			tblcreate = "CREATE TABLE " + table_name + "(cancellation_date
text, username text, route_id char(5), bus_id char(5), starting text, ending
text, reservation_date text, seat_no text, ticket_no text PRIMARY KEY,
amount_forfeited int(3))"
			curs.execute(tblcreate)
			conn.commit()
			r = 1

		except:
			r = 2
	else:
		r = 0

	return r, table_name

# ticketDetails(ticket_no {str}, ch {1|2}) -> returns a tuple

'''
Verifies if an entry with the given ticket number is present in the table_name.
Returns:
	None if no entry was found with the given ticket number
	a tuple with all information of the entry, if found
For internal use.
'''
def ticketDetails(ticket_no, ch = 1):
	r = None
	r1, reserv = create_reservation_table(0)
	r2, cancel = create_cancellation_table(0)
	if ch == 1: table_name = reserv
	elif ch == 2: table_name = cancel
	if init() != 2:
		try:
			tblcmd = "SELECT * FROM '" + table_name + "' WHERE
ticket_no='" + ticket_no + "'"
			curs.execute(tblcmd)
			r = curs.fetchall()
			r = None if r == [] else r[0]
		except:
			pass
	return r

# add_cancellation(ticket_no {str}, amount_forfeited {int}) -> returns an int

'''
Used to add a cancellation record to cancellation_table. Also removes the
specific entry from reservation_table. Returns:
	0 if there is no reservation with the given ticket_no
	1 if record was successfully processed
	2 if there w
```

```
Example:

mdb.add_reservation('V10', 'src', 'Howrah', 'Durgapur', '14/08/2017', 35, 550)
'14082017V10035DUR2F1T2'
mdb.add_cancellation('14082017V10035DUR2F1T2', 50)
1
'''
def add_cancellation(ticket_no, amount_forfeited = 0):

      r = 2

      r1, cancelTName = create_cancellation_table(0)
      r2, reservTName = create_reservation_table(0)

      details = ticketDetails(ticket_no, 1)

      if details == None:
            r = 0

      elif r1 != 2 and r2 != 2:

            try:

                  username = details[2]
                  route_id = details[0]
                  bus_id = details[1]
                  starting = details[3]
                  ending = details[4]
                  reservation_date = details[5]
                  seat_no = details[6]
                  amount = details[7]

                  if calc.isValidTransactionDate(reservation_date):
                        # removing from reservation_table
                        tblremove = "DELETE FROM '" + reservTName + "' WHERE
ticket_no='" + ticket_no + "'"
                        curs.execute(tblremove)

                        # adding to cancellation_table
                        tbladd = "INSERT INTO '" + cancelTName + "'
values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

                        curs.execute(tbladd, (time.strftime("%d/%m/%Y"), username,
route_id, bus_id, starting, ending, reservation_date, seat_no, ticket_no,
amount_forfeited))
                        conn.commit()

                        # add to revenue_table
                        if add_revenue('cancellation', username, ticket_no, -
(amount - amount_forfeited), amount_forfeited) != 1:
                              print('Error adding revenue.')


                        # add to user_activities
                        cancellation_string = ticket_no + '_' +
time.strftime("%d/%m/%Y") + '_' + bus_id + '_' + starting + '_' + ending + '_'
+ reservation_date + '_' + str(seat_no) + '_' + str(amount)
```

```
                        if change_user_activity(username, 'cancellations',
cancellation_string, 1) != 1:
                              print ('Error adding to user_activities.')

                        r = 1

                  else: r = -1

            except: r = 2

      return r


# create_user_details_table(n {0|1}, table_name {str}) -> returns an int
'''
Creates the user_details table to store personal information like name,
password etc. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made
n : pass 1 to recreate table
'''
def create_user_details_table(n = 0, table_name = 'user_details'):

      r = 2
      if isTablePresent(table_name) == False or n == 1:

            try:

                  # delete old table if user selects to recreate data
                  tbldelete = "DROP TABLE IF EXISTS " + table_name
                  curs.execute(tbldelete)

                  # create table
                  tblcreate = "CREATE TABLE " + table_name + "(name text,
username text PRIMARY KEY, type char(5), password text, security_ques text,
security_answer text, payments text)"
                  curs.execute(tblcreate)
                  conn.commit()

                  # add a default administrator account
                  dtlins = "INSERT INTO '" + table_name + "'
values(?, ?, ?, ?, ?, ?, ?)"
                  curs.execute(dtlins, ('Administrator', 'admin', 'admin',
'admin', '', '', '')) # '' -> payment kept blank
                  conn.commit()

                  r = 1

            except: r = 2

      else:
            r = 0

      return r, table_name

# checkUsernamePresence(username {str}) -> returns bool
'''
```

```
Checks for the presence of a username in user_details table. All usernames must
be unique.
Returns True if present else False

Example:

mdb.checkUsernamePresence('ad')
True
'''
def checkUsernamePresence(username, table_name = ''):

      presence = False
      r1 = 0
      if table_name == '': r1, table_name = create_user_details_table(0)

      if r1 != 2 and init() != 2:
            try:
                  tblcmd = "SELECT username FROM '" + table_name + "'"
                  curs.execute(tblcmd)
                  usernames = curs.fetchall()
                  if (username,) in usernames:
                        presence = True
                  else: presence = False
            except:
                  pass

      return presence

def getNameFromUsername(username):

      r1, table_name = create_user_details_table(0)
      name = ''

      if r1 != 2 and checkUsernamePresence(username) == 1:
            try:
                  tblcmd = "SELECT name FROM '" + table_name + "' WHERE username
='" + username + "'"
                  curs.execute(tblcmd)
                  names = curs.fetchall()
                  name = names[0][0]
            except:
                  pass

      return name


# add_user(name {str}, username {str}, password {str}, security_ques {str},
security_answer {str}) -> returns an int
'''
Adds a user to user_details table. Returns:
      0 if username was present and user can't be added
      1 if username was not present and user successfully added
      2 in case of error

Example:

mdb.add_user('Dummy user', 'ag', 'dup', 'demo_q', 'demo_a') # 'ag' username is
already present
0
```

```
mdb.add_user('Dummy user', 'du', 'dup', 'demo_q', 'demo_a')
1
'''
def add_user(name, username, password, security_ques, security_answer):

      r = 2
      r1, userTName = create_user_details_table(0)
      r2, userActivityTable = create_user_activities_table(0)
      if r1 != 2 and r2 != 2:
            if checkUsernamePresence(username, userTName) == False:

                  try:
                        # insert into user_details
                        dtlins = "INSERT INTO '" + userTName + "'
values(?, ?, ?, ?, ?, ?, ?)"
                        curs.execute(dtlins, (name, username, 'cust', password,
security_ques, security_answer, ''))  # '' -> payment kept blank

                        # insert into user_activities
                        actins = "INSERT INTO '" + userActivityTable + "'
values(?, ?, ?, ?, ?)"
                        curs.execute(actins, (username, '', '', '', ''))

                        conn.commit()
                        r = 1
                  except:
                        r = 2
            else:
                  r = 0

      return r

# ************** internal use only ****************
'''
This method adds or removes a given element from an object returned by
curs.fetchall(). Returned is a string with line breaks.
source - data returned from curs.fetchall()
entry - the entry to be added to or removed from source
job - 1-> add entry to source, 2-> remove entry from source
Return:
2: error
1: success
-2: could not remove
'''
def entryAdditionRemoval(source, entry, job):
      source = source[0][0]
      r = 2
      if job == 1:
            if source == None or source == '':
                  source = entry
                  r = 1
            else:
                  source = source.split('\n')
                  if entry not in source:
                        source.append(entry)
                        r = 1
                  else:
```

```
                source = '\n'.join(source)
                source = source.strip()

        elif job == 0:
                if source == None:
                        pass
                elif entry == '':
                        source = ''
                elif entry == None:
                        source = None
                else:
                        source = source.split('\n')
                        try:
                                source.remove(entry)
                                r = 1
                        except: r = -2
                        source = '\n'.join(source)
                        source = source.strip()

        return r, source


# doesPasswordMatch(username {str}, password {str}) -> returns an int
'''
Used to verify if entered username matches with password. Returns:
1: match
-1: doesn't match
0: username not found
2: any other error
'''
def doesPasswordMatch(username, password):

        r = 2
        r1, userTName = create_user_details_table(0)
        if r1 != 2:
                if checkUsernamePresence(username, userTName):

                        try:
                                tblselect = "SELECT password FROM '" + userTName + "'
WHERE username='" + username + "'"
                                curs.execute(tblselect)
                                passwd = curs.fetchall()

                                if passwd[0][0] == password: r = 1
                                else: r = -1

                        except: r = 2

                else: r = 0
        return r


# verifyAdmin(username {str}, password {str}) -> returns bool
'''
Used to verify if entered username and password matches with administrator.
Returns True or False as the case may be.
'''
def verifyAdmin(username, password):
```

```
	v = False
	r1, table_name = create_user_details_table(0)

	if r1 != 2 and doesPasswordMatch(username, password) == 1:
		try:
			curs.execute("SELECT type FROM " + table_name + " WHERE
username='" + username + "'")
			t = curs.fetchall()
			t = t[0][0]
			if t == 'admin': v = True
			else: v = False
		except: pass
	return v


# change_user_payment(username {str}, password {str}, payment {str}, mode {0|1})
-> returns an int
'''
This method is used to add or remove payment options for a specified username.
Returns:
	0 if username is not found
	1 if payment method is successfully added or removed
	-1 if password is incorrect
	-2 if payment method was already present and no changes were made (only
for adding payment method)
	-3 payment removal error
	2 if there was any other error
mode = 1: add the payment method, 0: remove the payment method

Example:

mdb.change_user_payment('sr', 'srp', '4321-5678-1573-2389', 1)
1
'''
def change_user_payment(username, password, payment, mode = 1):

	r = 2
	r1, userTName = create_user_details_table(0)
	if r1 != 2:
		if checkUsernamePresence(username, userTName):

			try:
				tblselect = "SELECT payments FROM '" + userTName + "'
WHERE username='" + username + "'"
				curs.execute(tblselect)
				payments = curs.fetchall()

				if doesPasswordMatch(username, password) == 1:

					success, payments = entryAdditionRemoval(payments,
payment, mode)

					if success == 0: r = -2
					elif success == -2: r = -3
					elif success == 1:
						# update table
						tblupd = "UPDATE '" + userTName + "' SET
payments='" + payments + "' WHERE username='" + username + "'"
```

```
                                    conn.commit()
                                    r = 1
                              else: r = 2

                        else: r = -1

                  except: r = 2
            else: r = 0

      return r


# change_user_detail(username {str}, field {str}, fieldvalue {str}) -> returns
an int
'''
This method is used to change a data in user_details table. Returns:
      0 if username is not found
      1 if change was incorporated
      2 if there was any other error
field: name of field (or column) to be changed for the given username
fieldvalue: value to be placed in the given field

Example:

mdb.change_user_detail('src', 'security_ques', 'qqq')
1

mdb.change_user_detail('src', 'security_answer', 'aaa')
1
'''
def change_user_detail(username, field, fieldvalue):

      r = 2
      r1, userTName = create_user_details_table(0)
      if r1 != 2:
            if checkUsernamePresence(username, userTName):
                        try:
                              tblupd = "UPDATE '" + userTName + "' SET " + field +
"='" + fieldvalue + "' WHERE username='" + username + "'"
                              curs.execute(tblupd)
                              conn.commit()
                              r = 1

                        except:
                              r = 2
            else:
                  r = 0

      return r

# get_all_user_detail(username {str}) -> returns a tuple
'''
This method is used to get all information like name, password etc of a
username. Returns:
      a tuple with all information, if username is present
      () if username was not found

Example:
```

```
mdb.get_all_user_details('ag')
('Aayush', 'ag', 'cust', 'agp', '', '', '')
'''
def get_all_user_details(username):

      r1, userTName = create_user_details_table(0)
      data = ()
      if r1 != 2 and checkUsernamePresence(username, userTName):
            try:
                  curs.execute("SELECT * FROM " + userTName + " WHERE
username='" + username + "'")
                  data = curs.fetchall()[0]
            except: pass

      return data

# remove_user(username {str}, password {str}) -> returns a bool
'''
This method is used to remove a user and his/her information. Correct password
must be provided for this operation. Returns:
      True, if user was removed, False otherwise

Example: remove the previously created Dummy User

mdb.remove_user('du', 'dup')
True
'''
def remove_user(username, password):

      r1, userTName = create_user_details_table(0)
      r2, userActivityName = create_user_activities_table(0)
      success = False
      if r1 != 2 and r2 != 2 and checkUsernamePresence(username, userTName):
            try:
                  curs.execute("SELECT password FROM " + userTName + " WHERE
username='" + username + "'")
                  p = curs.fetchall()
                  if [(password,)] == p:
                        curs.execute("DELETE FROM " + userTName + " WHERE
username='" + username + "'")
                        curs.execute("DELETE FROM " + userActivityName + " WHERE
username='" + username + "'")
                        conn.commit()
                        success = True

            except: pass

      return success

# create_user_activities_table(n {0|1}, table_name {str}) -> returns an int
'''
Creates the user_activities table, which stores information like reservations,
cancellations etc. Returns:
      0 if table was present
      1 if table was not present and created
      2 if table could not be made
n : pass 1 to recreate table
'''
def create_user_a[illegible]ser_activities'):
```

```
    r = 2
    if isTablePresent(table_name) == False or n == 1:

        try:

            # delete old table if user selects to recreate data
            tbldelete = "DROP TABLE IF EXISTS " + table_name
            curs.execute(tbldelete)

            # create table
            tblcreate = "CREATE TABLE " + table_name + "(username text
PRIMARY KEY, time_tables text, buses_between_stops text, reservations text,
cancellations text)"
            curs.execute(tblcreate)
            conn.commit()

            r = 1

        except:
            r = 2

    else:
        r = 0

    return r, table_name


# get_user_activity(username {str}, field {str}) -> returns a str
'''
This method is used to get a particular activity information for a username
from user_activities table
    a string with the information from the given field, if username is
present
    None if username was not found

Example:

mdb.get_user_activity('sr', 'reservations')
'25072017SL2030AS2F1T2_17/07/2017_SL2_Kolkata_Bardhaman_25/07/2017_30_270\n2508
2017S5020MID1F2T3_17/07/2017_S5_Kolaghat_Kharagpur_25/08/2017_20_300'

'''
def get_user_activity(username, field):

    data = None

    r1, userTName = create_user_activities_table(0)

    if r1 != 2 and checkUsernamePresence(username, userTName):

        try:
            tblcmd = "SELECT " + field + " FROM " + userTName + " WHERE
username='" + username + "'"
            curs.execute(tblcmd)
            data = curs.fetchall()
            data = data[0][0] if data != None else None

        except
```

```
      return data

# change_user_activity(username {str}, field {str}, fieldvalue {str}, mode
{0|1}) -> returns an int
'''
Used to add or remove data (fieldvalue) from a given column (field) for a
username
      0 if username was not found
      1 if table was successfully updated
      2 for any other error
mode: 1 -> write fieldvalue to field, 2 -> remove fieldvalue from field

Example in add_reservation() and add_cancellation() methods.
'''
def change_user_activity(username, field, fieldvalue, mode = 1):

      r = 2
      r1, userTName = create_user_activities_table(0)
      if r1 != 2 and field != 'username':
            if checkUsernamePresence(username, userTName):

                  try:
                        tblselect = "SELECT " + field + " FROM '" + userTName +
"' WHERE username='" + username + "'"
                        curs.execute(tblselect)
                        values = curs.fetchall()

                        success, values = entryAdditionRemoval(values, fieldvalue,
mode)

                        # update table
                        tblupd = "UPDATE '" + userTName + "' SET " + field + "='"
+ values + "' WHERE username='" + username + "'"
                        curs.execute(tblupd)
                        conn.commit()
                        if success != 2: r = 1
                        else: r = 2

                  except: r = 2
            else:
                  r = 0

      return r

# buses_between_stops(source {str}, destination {str}, username {str}) ->
returns a list of bus_id
'''
Returns a list of buses running from source to destination

Example:

mdb.buses_between_stops('Kolkata', 'Bardhaman')
['S1', 'AC1', 'SL1', 'S2', 'SL2', 'S4', 'SL4']
'''
def buses_between_stops(source, destination, username = ''):

      r1, bus_table_name = create_bus_table(0)
      r2, route_ta
```

```
	r3, user_activities_table = create_user_activities_table(0)

	buses = []

	if r1 != 2 and r2 != 2 and r3 != 2:

		curs.execute("SELECT route_id FROM " + route_table_name)
		rids = curs.fetchall()
		rids = [rid for (rid,) in rids if validate_route(rid, source,
destination) != None]

		for rid in rids:
			curs.execute("SELECT bus_id FROM " + bus_table_name + " WHERE
route_id='" + rid + "'")
			bids = curs.fetchall()
			bids = [bid for (bid,) in bids]
			buses = buses + bids

			# push to user_activities, if username is available
			if username != '' and buses != [] and
checkUsernamePresence(username, user_activities_table):
				if change_user_activity(username, 'buses_between_stops',
str(source + '_' + destination), 1) != 1:
					print ('Error adding to user_activities.')


	return buses


# bus_timetable(bus_id {str}, username {str}) -> returns a list of tuples
'''
Returns a list of tuples with bus stop name and ETA at that stop

Example:

mdb.bus_timetable('S2', 'ag')
[('Kolkata', '05:00'), ('Bardhaman', '06:00'), ('Durgapur', '06:30'),
('Asansol', '08:00')]
'''

def bus_timetable(bus_id, username = ''):

	r1, fare_chart_name = create_fare_chart(0)
	r2, route_table_name = create_route_table(0)
	r3, user_activities_table = create_user_activities_table(0)
	route_id = getRouteFromBusID(bus_id)

	time_table = []

	if r1 != 2 and r2 != 2 and r3 != 2 and route_id != '':

		try:

			curs.execute("SELECT source, stop_1, stop_2, destination FROM
" + route_table_name + " WHERE route_id='" + route_id + "'")
			places = curs.fetchall()
			places = [place for place in places[0] if place != '']
```

```
                curs.execute("SELECT source_time, stop_1_time, stop_2_time,
destination_time FROM " + fare_chart_name + " WHERE bus_id='" + bus_id + "'")
                times = curs.fetchall()
                times = [time for time in times[0] if time != None]

                time_table = list(zip(places, times))

                # push to user_activities, if username is available
                if username != '' and time_table != [] and
checkUsernamePresence(username, user_activities_table):
                      if change_user_activity(username, 'time_tables', bus_id,
1) != 1:
                            print ('Error adding to user_activities.')

           except: pass

      return time_table


# availableSeats(bus_id {str}, starting {str}, ending {str}, date {str}) ->
returns a list of reservable seats
'''
Example: assuming seat 30 and 31 is booked in bus SL2 on 25/07/2017

mdb.availableSeats('SL2', 'Kolkata', 'Bardhaman', '25/07/2017')
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 32, 33, 34, 35, 36, 37, 38, 39, 40]
'''
def availableSeats(bus_id, starting, ending, date):

      route_id = getRouteFromBusID(bus_id)
      r1, bus_table_name = create_bus_table(0)
      seats = []

      if r1 != 2 and route_id != '':
           try:
                if validate_route(route_id, starting, ending) != None:

                      curs.execute("SELECT total_seats FROM " + bus_table_name
+ " WHERE bus_id='" + bus_id + "'")
                      total_seats = curs.fetchall()
                      total_seats = total_seats[0][0]

                      seats = [seat_no for seat_no in range(1, total_seats+1)
if isReservationPossible(bus_id, starting, ending, date, seat_no)]

           except: seats = []

      return seats

# adminDisplayTable(ch)
'''
Used to print tables by admin account
ch = 1: print reservation table
ch = 2: print cancellation table
ch = 3: print route table
ch = 4: print fare chart
ch = 5: print bus table
'''
```

```
def adminDisplayTable(ch):

      if ch == 1:
            r1, table_name = create_reservation_table(0)
            headers, data = get_table(table_name)
      elif ch == 2:
            r1, table_name = create_cancellation_table(0)
            headers, data = get_table(table_name)
      elif ch == 3:
            r1, table_name = create_route_table(0)
            headers, data = get_table(table_name)
      elif ch == 4:
            r1, table_name = create_fare_chart(0)
            headers, data = get_table(table_name)
      elif ch == 5:
            r1, table_name = create_bus_table(0)
            headers, data = get_table(table_name)

      ipop.print_table((headers, data))

# getRevenue(from_date {str}, to_date {str}) -> returns tuple
'''
Returns data from revenue_table filtering between from_date to to_date.
Returns: (revenue_table headers, revenue_table content, total revenue)

Example:

mdb.getRevenue('01/06/2017', '01/07/2017')
(['mode', 'username', 'ticket_no', 'date', 'amount', 'discount_or_penalty'],
[('reservation', 'mn', '23062017AC5023MID1F2T4', '19/06/2017', 400, 0),
('reservation', 'mn', '02072017AC5023MID1F2T4', '28/06/2017', 400, 0)], 800)
'''
def getRevenue(from_date = '', to_date = ''):

      dtfrom = None
      dtto = None
      final_revenue_data = []
      total_revenue = 0

      if from_date != '':
            from_date = from_date.split('/')
            dtfrom = datetime.datetime(int(from_date[2]), int(from_date[1]),
int(from_date[0]))    # datetime object 1

      if to_date != '':
            to_date = to_date.split('/')
            dtto = datetime.datetime(int(to_date[2]), int(to_date[1]),
int(to_date[0]))      # datetime object 2

      r1, table_name = create_revenue_table(0)
      headers, revenue_data = get_table(table_name)

      if r1 != 2:

            for rdata in revenue_data:

                  data = rdata
                  dt = data[3]
                  d
```

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

```
                dtobj = datetime.datetime(int(dt[2]), int(dt[1]), int(dt[0]))
            # making datetime object for each date of each row

                # comparing and filtering
                if dtfrom != None:
                      if dtobj < dtfrom: data = None
                if dtto != None:
                      if dtobj > dtto: data = None
                if data != None:
                      final_revenue_data.append(data)
                      total_revenue = total_revenue + data[4]

      return headers, final_revenue_data, total_revenue


# order_rc_by_catagories(cat {1|2|3}, from_date {str}, to_date {str}, t
{'r'|'c'}) -> returns tuple
'''
cat:
1 - bus_id
2 - route_id
3 - route (Example: Howrah_Haldia)
t:
'r' - reservations
'c' - cancellations
Returns in descending order, the number of reservations or cancellations (based
on t) on a particular type of field (based on cat). Date filtering can be
applied.
Returned tuple has - (headers, content, total reservation/cancellation)

Example:

mdb.order_rc_by_catagories(cat=2)      # no date filtering. Returns in
descending order the number of reservations per route_id
(['Reservations', 'Route ID'], [(7, 'MID1'), (2, 'DUR2'), (2, 'AS2'), (1,
'AS1')], 12)

mdb.order_rc_by_catagories(cat=1,to_date='01/08/2017') # all previous data is
considered upto to_date. Returns in descending order the number of reservations
per bus_id
(['Reservations', 'Bus ID'], [(3, 'AC5'), (2, 'V10'), (2, 'SL2'), (1, 'SL1'),
(1, 'S5')], 9)

mdb.order_rc_by_catagories(cat=3, from_date = '01/08/2017') # all next data is
considered from from_date. Returns in descending order the number of
reservations per route
(['Reservations', 'Route'], [(2, 'Howrah_Durgapur'), (1, 'Kolkata_Midnapore'),
(1, 'Kolkata_Kolaghat'), (1, 'Kolaghat_Kharagpur')], 5)

mdb.order_rc_by_catagories(cat=1, t = 'c')  # cancellation data is arranged on
bus_id
(['Cancellations', 'Bus ID'], [(3, 'SL2'), (2, 'S5'), (2, 'AC5'), (1, 'V10')],
8)
'''
def order_rc_by_catagories(cat, from_date = '', to_date = '', t = 'r'):

      dtfrom = None
      dtto = None
      final_reserv
```

```
    total_count = 0
    data_dict = dict()
    data_list = []
    h1 = ''

    if t == 'r':
          di = 9
          ri = 0
          si = 3
          ei = 4
          bi = 1
          h1 = 'Reservations'
          r1, table_name = create_reservation_table(0)
    elif t == 'c':
          di = 0
          ri = 2
          si = 4
          ei = 5
          bi = 3
          h1 = 'Cancellations'
          r1, table_name = create_cancellation_table(0)
    else: return None, None

    if from_date != '':
          from_date = from_date.split('/')
          dtfrom = datetime.datetime(int(from_date[2]), int(from_date[1]),
int(from_date[0]))

    if to_date != '':
          to_date = to_date.split('/')
          dtto = datetime.datetime(int(to_date[2]), int(to_date[1]),
int(to_date[0]))

    _, reservation_data = get_table(table_name)


    if cat == 1: h2 = 'Bus ID'
    elif cat == 2: h2 = 'Route ID'
    elif cat == 3: h2 = 'Route'
    else: return None, None

    if r1 != 2:

          for rdata in reservation_data:

                data = rdata
                dt = data[di]
                dt = dt.split('/')
                dtobj = datetime.datetime(int(dt[2]), int(dt[1]), int(dt[0]))

                if dtfrom != None:
                      if dtobj < dtfrom: data = None
                if dtto != None:
                      if dtobj > dtto: data = None
                if data != None:
                      final_reservation_data.append(data)
                      total_count = total_count + 1

          for rda
```

```
                data = rdata
                if cat == 1: key = data[bi]
                elif cat == 2: key = data[ri]
                elif cat == 3: key = data[si] + '_' + data[ei]
                data_dict[key] = data_dict.get(key, 0) + 1

            for k, v in data_dict.items(): data_list.append((v, k))

            data_list.sort(reverse = True)

        headers = [h1, h2]

        return headers, data_list, total_count


# order_rc_by_month(t {'r'|'c'}, rid {str}, bid {str}, starting {str}, ending
{str})
'''
t:
'r' - reservations
'c' - cancellations
Returns in descending order, the number of reservations or cancellations on a
per-month basis. The sorting is done in one or more of the parameters:
bid - bus_id
rid - route_id
starting - start of journey
ending - end of journey
Atleast one of the above 4 parameters MUST BE provided.

Example:

mdb.order_rc_by_month(rid='MID1')      # reservations per month on route_id
'MID1'
(['Number of reservations', 'For the month of'], [(3, '08/2017'), (3,
'07/2017'), (1, '06/2017')])

mdb.order_rc_by_month(starting='Kolkata')   # reservations per month starting
from 'Kolkata'
(['Number of reservations', 'For the month of'], [(4, '07/2017'), (2,
'08/2017')])

mdb.order_rc_by_month(ending='Midnapore', bid='AC5')   # reservations per month
on journeys on bus_id 'AC5' and ending at 'Midnapore'
(['Number of reservations', 'For the month of'], [(2, '07/2017'), (1,
'08/2017'), (1, '06/2017')])

mdb.order_rc_by_month(t='c', bid='AC5')           # cancellations per month on
bus_id 'AC5'
(['Number of cancellations', 'For the month of'], [(2, '07/2017')])

mdb.order_rc_by_month(rid='DUR2', bid='V10')      # the rid here is unnecessary
as bus_id 'V10' will always have route_id as 'DUR2'
(['Number of reservations', 'For the month of'], [(2, '08/2017')])

mdb.order_rc_by_month(rid='DUR1', starting='Kolkata')  # empty list returned as
'Kolkata' is not present in route_id 'DUR1', hence there was no reservations.
(['Number of reservations', 'For the month of'], [])
'''
```

```
def order_rc_by_month(t = 'r', rid = '', bid = '', starting = '', ending = ''):

      if rid == bid == starting == ending == '': return 0

      date_label = ''
      table_name = ''
      header = ['For the month of']
      dates = []
      rc_dict = dict()
      rc_list = []

      r1, rtn = create_reservation_table(0)
      r2, ctn = create_cancellation_table(0)

      if t == 'r' and r1 != 2:
            header = ['Number of reservations'] + header
            date_label = 'date'
            table_name = rtn
      elif t == 'c' and r2 != 2:
            header = ['Number of cancellations'] + header
            date_label = 'cancellation_date'
            table_name = ctn
      else: return [],[]

      s = lambda h, d: h+"='"+d+"' AND " if d != '' else ''

      tblselect = "SELECT " + date_label + " FROM " + table_name + " WHERE " +
s('route_id', rid) + s('bus_id', bid) + s('starting', starting) + s('ending',
ending)
      tblselect = tblselect[:len(tblselect)-5]
      curs.execute(tblselect)
      dates = curs.fetchall()

      for (rdate,) in dates:

            date = rdate
            date = date.split('/')
            m = date[1]
            y = date[2]
            k = m + '/' + y

            rc_dict[k] = rc_dict.get(k, 0) + 1


      for k, v in rc_dict.items(): rc_list.append((v, k))

      rc_list.sort(reverse = True)

      return header, rc_list


# initialise()
'''
Creates all tables needed for execution of the program. Should be run at first.
'''
def initialise():
      create_bus_table(0)
      create_route_table(0)
      create_fare_
```

```
	create_reservation_table(0)
	create_cancellation_table(0)
	create_revenue_table(0)
	create_user_details_table(0)
	create_user_activities_table(0)
```

Busses Between Stops

```
import os
import sys
import manageDB as mdb
import ipop

def main():

	ipop.cls()

	username = ''

	try:
		username = sys.argv[1]
		if mdb.checkUsernamePresence(username) == False: username = ''
	except: pass

	screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tEnter source and destination to get avilable
buses\t|\n" + "|\tExample: Howrah to Asansol etc...\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

	print (screen)

	bbslist = []

	if username != '':
		try:
			storedbbs = mdb.get_user_activity(username,
'buses_between_stops')
			if storedbbs != '':
				storedbbs = storedbbs.split('\n')
				bbslist = list(enumerate(storedbbs, start=1))
				print ("||\tYour previous searches:\n")
				h = ['Choice', 'Buses between stops']
				tup = (h, bbslist)
				ipop.print_table(tup)
				print ("||\tYou can enter these choices in starting
field.\n")
		except: pass


	while True:
		r = disp_bbs(username, bbslist)
		if r == 0:
			exit1(username)
			break
		else:
```

```
                ch = ipop.getUserData([int, str], "||\tEnter 1 for another
search or !q for main menu.\n\t", "Wrong data.")
                if ch == 1: continue
                else:
                      exit1(username)
                      break


def exit1(un):
      if un == '':
            os.system("python3 main_menu.py")
      else:
            os.system("python3 logged_in_main_menu.py '" + un + "'")

def disp_bbs(un, bbslist):

      buses = []
      source = ''
      destination = ''
      chflag = 0

      source = ipop.getUserData([str, int], "||\tEnter source or previous
search option (if available)\n||\tor !q to cancel: ", "Wrong data")

      if source == None:
            return 0
      for (i, d) in bbslist:
            if source == i:
                  source = d.split('_')[0]
                  destination = d.split('_')[1]
                  chflag = 1
                  break

      if chflag == 0:
            destination = ipop.getUserData([str, int], "||\tEnter destination of
journey or !q to cancel: ", "Wrong data")
            if destination == None:
                  return 0

      buses = mdb.buses_between_stops(source, destination, un)
      if buses == []:
            print ("This route does not exist.")
            return 1
      else:
            l = []
            print('\nFollowing buses are available:')
            print('From ' + source + ' to ' + destination)
            for b in buses:
                  l.append((b, mdb.getBusType(b), mdb.getTime(b, source,
destination), mdb.getFare(b, source, destination)))
            h = ['Bus ID', 'Bus type', 'Journey time', 'Journey fare']
            ipop.print_table((h, l))
            return 1

if __name__ == '__main__':
      main()
```

Payments

```
import sys
import os
import manageDB as mdb
import ipop

def main():

      ipop.cls()

      username = sys.argv[1]

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t\t|\n" + "|\tThis page lists all your payment methods.\t\t|\n"
+ "|\tYou can add or delete some of them\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print (screen)

      data = mdb.get_user_detail(username, 'payments')
      if data == None or data == '':
            print ("No payment methods available in this account.")
      else:
            lines = data.split('\n')
            m = list(enumerate(lines, start=1))

            h = ['Option', 'Payment method']

            ipop.print_table((h, m))


def exit1(un): os.system("python3 logged_in_main_menu.py '" + un + "'")

if __name__ == '__main__':
      main()
```

Main Menu

```
#!usr/bin/env python3

import ipop
import os
import sys

import manageDB as mdb

def main():

      ipop.cls()
      mdb.initialise()

      try:
            print(sys.argv[1])
      except: pass
```

```
	screen =
"_____________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tWelcome\t\t\t\t\t\t\t|\n" + "|\t\t\t\t\t\t\t\t|\n"
+ "|\tPlease select the appropriate option:\t\t\t|\n" + "|\t1) Bus time
table\t\t\t\t\t|\n" + "|\t2) Buses between stops\t\t\t\t\t|\n" + "|\t3) Seat
availability\t\t\t\t\t|\n" + "|\t\t\t\t\t\t\t\t|\n" + "|\tLogin and get access
to\t\t\t\t\t|\n" + "|\tSeat booking\t\t\t\t\t\t|\n" + "|\tand much
more...\t\t\t\t\t|\n" + "|\t\t\t\t\t\t\t\t|\n" + "|\tL) LOGIN\t\t\t\t\t\t|\n" +
"|\tNot registered yet? Getting an account is easy\t\t|\n" + "|\tS) SIGN UP
>>>>\t\t\t\t\t\t|\n" + "|\t\t\t\t\t\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

	print(screen)

	p = "Enter option (1,2,3,L,S): "
	ip = ipop.getUserData([int, str], p, "Wrong data entered!", True, ["x in
[1, 2, 3]", "x in ['L', 'S']"], True)
	if ip == 'L':
		os.system("python3 login_screen.py")
	elif ip == 'S':
		os.system("python3 sign-up_screen.py")
	elif ip == 1:
		os.system("python3 time_table_screen.py")
	elif ip == 2:
		os.system("python3 buses_between_stops.py")
	elif ip == 3:
		os.system("python3 seat_availability.py")

if __name__ == '__main__':
	main()
```

Login Screen

```
import manageDB as mdb
import ipop
import os
import sys

def main():

	ipop.cls()

	back = "python3 main_menu.py"
	uok = False

	while True:
		username = ipop.getUserData([str], "||\tEnter username:\n||\t",
"Wrong data!")
		if username == None:
			os.system(back)
			break
		else:
			i
```

```
                        uok = True
                        break
                else: print ("Username not present.")

      while uok:
            password = ipop.getUserData([str], "||\tEnter password:\n||\t",
"Wrong data!")
            if password == None:
                  os.system(back)
                  break
            else:
                  r = mdb.doesPasswordMatch(username, password)
                  if r == 1:
                        if mdb.verifyAdmin(username, password):
os.system("python3 admin_page.py '" + username + "'")
                        else: os.system("python3 logged_in_main_menu.py '" +
username + "'")
                        break
                  elif r == -1: print ("Wrong password.")
                  else:
                        print("Error!")
                        break

if __name__ == '__main__':
      main()
```

Sign Up Screen

```
import manageDB as mdb
import os
import ipop


def main():

      ipop.cls()

      uok = False

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t\t|\n" + "|\tSign up to get exclusive benefits!\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print(screen)

      name = ipop.getUserData([str], "||\tEnter name: ", "Wrong data!", True,
['x != ""'])

      while True:
            username = ipop.getUserData([str], "||\tEnter username: ", "Wrong
data!", True, ['x != ""'])
            if mdb.checkUsernamePresence(username) == False:
                  break
            else: p                                    e another username.")
```

```
	password = ipop.getUserData([str, int, float], "||\tEnter password: ",
"Wrong data!", True, ['x != ""'])

	sec_ques = "||\tEnter security question (in case you forget your
password)\n" + "||\tChoose an option:\n" + "||\t1) What was your first mobile
number?\n" + "||\t2) What was your first car number?\n" + "||\t3) Favourite
dish of your pet?\n" + "||\tE) Define your own question.\n"

	print(sec_ques)
	sq = ipop.getUserData([int, str], "||\tEnter option (1,2,3,E)\n||\t",
"Wrong data", True, ['x in [1,2,3]', 'x == "E"'], True)
	if sq == 1: sq = "What was your first mobile number?"
	elif sq == 2: sq = "What was your first car number?"
	elif sq == 3: sq = "Favourite dish of your pet?"
	else: sq = ipop.getUserData([str], "\tEnter your question: ", 'Wrong
data', True, ['x != "" or x != None'])

	sa = ipop.getUserData([str, int, float], "||\tEnter answer to security
question:\n||\t", "Wrong data", True, ['x != ""'])

	confirm = "Enter any character to confirm OR '!q' to cancel
registration....\n"
	print(confirm)
	c = ipop.getUserData([str], '', 'Enter any character.')

	if c != None:
		try:
			mdb.add_user(name, username, password, sq, sa)
			alert = "'\nRegistration successful! Please login again..\n'"
			os.system("python3 main_menu.py " + alert)
		except:
			print('Registration unsuccessful..')
			exit(0)
	else:
		alert = "\nRegistration cancelled..\n"
		os.system("python3 main_menu.py")

if __name__ == '__main__':
	main()
```

Main Menu Creation

```
import ipop
import os
import sys
import manageDB as mdb

def main():

	ipop.cls()
	username = sys.argv[1]
	name = mdb.getNameFromUsername(username)

	screen =
"_________________________________________________________________\n" +
```

```
"|\t\t\t\t\t\t\t\t|\n" + "|\tHello,\t\t\t\t\t\t\t|\n" + "|\t" + name + "\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tPlease select the appropriate option:\t\t\t|\n" +
"|\t1) Bus time table\t\t\t\t\t|\n" + "|\t2) Buses between stops\t\t\t\t\t|\n"
+ "|\t3) Seat availability\t\t\t\t\t|\n" + "|\t\t\t\t\t\t\t\t|\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tStart your journey...\t\t\t\t\t|\n" + "|\t4) Book
a seat\t\t\t\t\t\t|\n" + "|\t5) Cancel a seat\t\t\t\t\t|\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tYour activities\t\t\t\t\t\t|\n" + "|\t6) Your
reservations\t\t\t\t\t|\n" + "|\t7) Your cancellations\t\t\t\t\t|\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tNot " + name + "?\n" + "|\tL)
Logout\t\t\t\t\t\t|\n" + "|\t\t\t\t\t\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

	print(screen)

	p = "Enter option (1-7,L): "
	ip = ipop.getUserData([int, str], p, "Wrong data entered!", True, ["x in
list(range(1,10))", "x == 'L'"], True)

	if ip == 'L':
		os.system("python3 main_menu.py")
	elif ip == 1:
		os.system("python3 time_table_screen.py '" + username + "'")
	elif ip == 2:
		os.system("python3 buses_between_stops.py '" + username + "'")
	elif ip == 3:
		os.system("python3 seat_availability.py '" + username + "'")
	elif ip == 4:
		os.system("python3 add_reservation.py '" + username + "'")
	elif ip == 5:
		os.system("python3 add_cancellation.py '" + username + "'")
	elif ip == 6:
		os.system("python3 your_reservations.py '" + username + "'")
	elif ip == 7:
		os.system("python3 your_cancellations.py '" + username + "'")

if __name__ == '__main__':
	main()
```

Display Revenue

```
import ipop
import os
import sys
import manageDB as mdb
import calc

def main():

	ipop.cls()
	username = sys.argv[1]
	name = mdb.getNameFromUsername(username)

	fdate = ''
	tdate = ''
```

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

```
      screen =
"_____________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tHello administrator,\t\t\t\t\t\t\t|\n" + "|\t" +
name + "\n" + "|\t\t\t\t\t\t\t\t|\n" + "|\tPlease select the appropriate
option:\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print(screen)
      print("||\t1. Print entire table")
      print("||\t2. Print revenue between two dates")
      print("||")

      ip = ipop.getUserData([int], "||\tEnter option: ", "Wrong data entered!",
True, ["x in [1,2]"], True)
      print()

      if ip != None:
            ipop.cls()
            if ip == 1:
                  h, c, t = mdb.getRevenue()
                  ipop.print_table((h, c))
                  print('\nTotal revenue: ' + str(t))
                  p = input("\nPress any key to go to main menu....")
                  exit1(username)
            elif ip == 2:
                  while True:

                        print("||\tEnter a 'from' date (DD/MM/YYYY): ")
                        fdate = reqdate()
                        if fdate == '':
                              exit1(username)
                              break

                        print("||\tEnter a 'to' date (DD/MM/YYYY): ")
                        tdate = reqdate()
                        if tdate == '':
                              exit1(username)
                              break

                        if calc.compareDates(fdate, tdate):
                              h, c, t = mdb.getRevenue(fdate, tdate)
                              ipop.print_table((h, c))
                              print('\nTotal revenue: ' + str(t))
                              p = input("\nPress any key to go to main menu....")
                              exit1(username)
                              break
                        else:
                              print("The 'from' date is later than 'to'
date.\nEnter 1 to re-enter the dates !q to quit")
                              p = ipop.getUserData([int, str], "", "")
                              if p == 1: continue
                              else: break

def reqdate():
      d = ''
      while True:
            d = ipop.getUserData([str], '||\t', "Wrong data entered!")
            if d == None:
                  r
```

```
            else:
                  if calc.isPreviousDate(d) == False:
                        print("Wrong date format or future date entered. Re-enter
date or !q to cancel.")
                        continue
                  else: return d

def exit1(un): os.system("python3 admin_page.py '" + un + "'")

if __name__ == '__main__':
      main()
```

Display Tables

```
import ipop
import os
import sys
import manageDB as mdb

def main():

      ipop.cls()
      username = sys.argv[1]
      name = mdb.getNameFromUsername(username)

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tHello administrator,\t\t\t\t\t\t\t|\n" + "|\t" +
name + "\n" + "|\t\t\t\t\t\t\t\t|\n" + "|\tPlease select the appropriate
table:\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print(screen)
      print("||\t1. Reservation table")
      print("||\t2. Cancellation table")
      print("||\t3. Route table")
      print("||\t4. Fare chart")
      print("||\t5. Bus table")
      print("||")

      ip = ipop.getUserData([int], "||\tEnter table number: ", "Wrong data
entered!", True, ["x in list(range(1,6))"], True)
      print()

      if ip != None:
            ipop.cls()
            mdb.adminDisplayTable(ip)
            p = input("\nPress any key to go to main menu....")

      exit1(username)

def exit1(un): os.system("python3 admin_page.py '" + un + "'")

if __name__ == '__main__':
      main()
```

Bus Time Table

```
import os
import sys
import manageDB as mdb
import ipop

def main():

      ipop.cls()

      username = ''

      try:
            username = sys.argv[1]
            if mdb.checkUsernamePresence(username) == False: username = ''
      except: pass

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t\t|\n" + "|\tEnter bus ID to see time-table\t\t\t\t|\n" +
"|\tExample: S1, V10, AC5, SL2 etc...\t\t\t|\n" + "|\t\t\t\t\t\t\t\t\t|\n" +
"|\tDon't know bus id? Enter R to see buses between stops.\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print (screen)

      ttlist = []

      if username != '':
            try:
                  storedtt = mdb.get_user_activity(username, 'time_tables')
                  if storedtt != '':
                        storedtt = storedtt.split('\n')
                        ttlist = list(enumerate(storedtt, start=1))
                        print ("||\tYour previous searches:\n")
                        h = ['Choice', 'Bus ID']
                        tup = (h, ttlist)
                        ipop.print_table(tup)
                        print ("||\tYou can enter these choices in place of bus
ID.\n")
            except: pass


      while True:
            r = disp_tt(username, ttlist)
            if r == 0:
                  exit1(username)
                  break
            elif r == -1:
                  buses_bet_stops(username)
                  break
            else:
                  ch = ipop.getUserData([int, str], "||\tEnter 1 for another
search or !q for main menu.\n\t", "Wrong data.")
                  if ch == 1: continue
                  e
```

```
                        exit1(username)
                        break


def exit1(un):
      if un == '':
            os.system("python3 main_menu.py")
      else:
            os.system("python3 logged_in_main_menu.py '" + un + "'")

def buses_bet_stops(un):
      if un == '':
            os.system("python3 buses_between_stops.py")
      else:
            os.system("python3 buses_between_stops.py '" + un + "'")


def disp_tt(un, ttlist):

      bus_id = ipop.getUserData([str, int], "||\tEnter bus ID or !q to cancel:
", "Wrong data")

      if bus_id == 'R':
            return -1
      elif bus_id == None:
            return 0
      for (i, b) in ttlist:
            if bus_id == i:
                  bus_id = b
                  break

      tt = mdb.bus_timetable(bus_id, un)

      if tt == []:
            print ("Bus with this ID does not exist.")
            return 1
      else:
            h = ['Bus stop', 'Time']
            c = tt
            tup = (h,c)
            print ("\nTime table for: " + bus_id)
            ipop.print_table(tup)
            return 1

if __name__ == '__main__':
      main()
```

Seat Availability

```
import os
import sys
import manageDB as mdb
import ipop
import sys
import os
import calc
```

```
def main():

      ipop.cls()

      username = ''

      try:
            username = sys.argv[1]
            if mdb.checkUsernamePresence(username) == False: username = ''
      except: pass

      screen =
"_____________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tEnter source, destination, date and bus ID of
journey\t|\n" + "|\tto get available seats before planning reservation\t|\n" +
"|\tExample: Howrah to Haldia on 20/08/2017 in AC8\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print (screen)
      print("\nEnter !q to cancel anytime.\n")

      while True:
            r = disp_seats()
            if r == 0:
                  exit1(username)
                  break
            else:
                  ch = ipop.getUserData([int, str], "||\tEnter 1 to see more
seat availabilities or !q for main menu.\n\t", "Wrong data.")
                  if ch == 1: continue
                  else:
                        exit1(username)
                        break


def exit1(un):
      if un == '':
            os.system("python3 main_menu.py")
      else:
            os.system("python3 logged_in_main_menu.py '" + un + "'")


def disp_seats():

      source = ipop.getUserData([str], "||\tEnter source of journey:\n||\t",
"Wrong data")
      if source == None:
            return 0

      destination = ipop.getUserData([str], "||\tEnter destination of
journey:\n||\t", "Wrong data")
      if destination == None:
            return 0


      while True:
            date = ipop.getUserData([str], "||\tEnter date of journey: (as
DD/MM/YYYY format):\n||\t", "Wrong data")
            if date
```

```
                return 0
            else:
                if calc.isValidTransactionDate(date):
                    break
                else: print("Invalid date or wrong date format given.")

        buses = mdb.buses_between_stops(source, destination)
        if buses != []:
            bus_details = []
            for b in buses:
                n_seats = len(mdb.availableSeats(b, source, destination, date))
                bus_details.append((b, n_seats))
            bh = ['Bus ID', 'Seats available']
            print ("\nFollowing buses are available:")
            ipop.print_table((bh, bus_details))
            print()

        bus_id = ipop.getUserData([str], "||\tEnter bus ID:\n||\t", "Wrong data")
        if bus_id == None:
            return 0

        buses = mdb.buses_between_stops(source, destination)
        if bus_id not in buses:
            print ("This journey does not exist!!")
            return 1

        seats = mdb.availableSeats(bus_id, source, destination, date)
        if seats == []: print ("No more seats are available in this bus for the
given date!")
        else:
            print("||\tAvailable seats are:")
            print(seats)
            return 1

if __name__ == '__main__':
    main()
```

Reservation Table

```
import os
import sys
import manageDB as mdb
import ipop
import calc

def main():

    ipop.cls()

    username = sys.argv[1]

    screen =
"_____________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tWelcome to seat reservation.\t\t\t\t|\n" + "|\tYou
will be guided through the reservation process.\t|\n" + "|\tPlease enter the
relevant information as asked.\t\t|\n" +
"~~~~~~~~~~~~~~~~                                          ~~~~~~~~~~~\n"
```

```
      print (screen)

      print ("\nOn entering the start and end of your journey\nwe will
automatically list the available buses for you to choose.\nYou can quit the
process anytime by entering !q\n")

      routeFlag = False
      dateFlag = False
      seatFlag = False
      seatReserveFlag = False

      source = ''
      destination = ''
      bus_id = ''
      date = ''
      n = 0
      seats = []
      bookedSeats = []
      fare = 0
      amount = 0

      ts = '00:00'
      te = '00:00'

      tickets = []

      while True:
            date = ipop.getUserData([str], "||\tEnter date of journey: (as
DD/MM/YYYY format):\n||\t", "Wrong data")
            if date == None:
                  return 0
            else:
                  if calc.isValidTransactionDate(date):
                        dateFlag = True
                        break
                  else: print("Invalid date or wrong date format given.")


      while dateFlag:

            source = ipop.getUserData([str], "||\tEnter source:\n||\t", "Wrong
data")
            if source == None:
                  exit1(username)
                  break
            destination = ipop.getUserData([str], "||\tEnter destination:\n||\t",
"Wrong data")
            if destination == None:
                  exit1(username)
                  break
            buses = mdb.buses_between_stops(source, destination)
            if buses == []:
                  print ("Sorry, this route does not exist. Press 1 to re-enter
start and destination, !q to cancel and go to main menu.")
                  p = ipop.getUserData([str, int], "", "Wrong data")
                  if p == 1: continue
                  else:
```

```
                        break
            else:
                  bus_details = []
                  for b in buses:
                        bus_type = mdb.getBusType(b)
                        bus_fare = mdb.getFare(b, source, destination)
                        n_seats = len(mdb.availableSeats(b, source, destination,
date))
                        bus_details.append((b, bus_type, bus_fare, n_seats))
                  bh = ['Bus ID', 'Type', 'Fare', 'Seats available']
                  print ("\nFollowing buses are available:")
                  ipop.print_table((bh, bus_details))
                  print()
                  bus_id = ipop.getUserData([str], "||\tEnter bus id:\n||\t",
"Choose among the available buses.", True, ['x in ' + str(buses)], True)
                  if bus_id == None:
                        exit1(username)
                  else: routeFlag = True
                  break


      if routeFlag:
            print ("Almost there....")
            while True:
                  n = ipop.getUserData([int], "||\tEnter number of seats to be
reserved:\n||\t", "Wrong data")
                  if n == None:
                        exit1(username)
                  else:
                        seats = mdb.availableSeats(bus_id, source, destination,
date)
                        if n > len(seats):
                              print("The requested number of seats is currently
not available in " + bus_id + " on " + date)
                              p = ipop.getUserData([int, str], "||\tEnter 1 to
decrease number of seats or !q to cancel:\n||\t", "Wrong data")
                              if p == 1: continue
                              else:
                                    exit1(username)
                                    break
                        else:
                              seatFlag = True
                              break


      if seatFlag:
            p = ipop.getUserData([str], "||\tEnter S to manually select seats or
any other key for automatic selection:\n||\t", "Wrong data")
            if p == None:
                  exit1(username)
            elif p == 'S' or p == 's':
                  print ("||\tAvailable seats are:")
                  print (seats)
                  print ("||\tEnter seat numbers for " + str(n) + "
reservations")
                  for i in range(n):
                        s = ipop.getUserData([int], "", "Wrong data", True, ['x
in ' + str(seats)], True)
```

```
                                exit1(username)
                            bookedSeats.append(s)
            else:
                  for i in range(n):
                        s = seats[i]
                        bookedSeats.append(s)
            seatReserveFlag = True
            print ("||\tBooked seats are: " + str(bookedSeats) + '\n')

      if seatReserveFlag:
            fare = mdb.getFare(bus_id, source, destination)
            amount = n*fare
            print("||\tAll set! Total Fare = " + str(n) + " * " + str(fare) + "
= " + str(amount))
            print("||\tPress any key to book. !q to cancel")
            p = input()
            if p == None:
                  exit1(username)
            else:
                  for bs in bookedSeats:
                        tickets.append(mdb.add_reservation(bus_id, username,
source, destination, date, bs, fare) + '\t' + str(bs))

                  timetable = mdb.bus_timetable(bus_id)
                  for place, time in timetable:
                        if source == place: ts = time
                        if destination == place: te = time

                  ticket_print(username, tickets, bus_id,
mdb.getRouteFromBusID(bus_id), source, destination, date, n, amount, ts, te)


def ticket_print(un, tickets, bus_id, route_id, source, destination, date, n,
amount, ts, te):

      ticketsConcat = ''
      for t in tickets:
            ticketsConcat = ticketsConcat + "||\t" + t + '\n'

      tickString =
"_____________________________________________________________\n" + "||\n"
+ "||\tReservation in bus ID: " + bus_id + "\n" + "||\tOn route ID: " +
route_id + "\n" + "||\tJourney starting from: " + source + "\n" + "||\tJourney
ending at: " + destination + "\n" + "||\tOn date: " + date + '\n' +
"||\tJourney from: " + ts + " to " + te + '\n' + "||\tNumber of reservations =
" + str(n) + '\n||\n' + "||\tTicket numbers:\t\tSeat no:\n" + ticketsConcat +
'||\n' + "||\tTotal amounting to: " + str(amount) + '\n' +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      ipop.cls()
      print (tickString + '\n')
      print ("Please note down the ticket numbers.\n")
      p = input("Press any key to go to main menu....")
      exit1(un)

def exit1(un): os.system("python3 logged_in_main_menu.py '" + un + "'")

if __name__ == '__main__':
      main()
```

Cancellation Table

```
import sys
import manageDB as mdb
import os
import ipop
import calc

def main():

      ipop.cls()

      username = sys.argv[1]

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t\t|\n" + "|\tWelcome to seat cancellation.\t\t\t\t|\n" +
"|\tYou will need to provide the ticket number\t\t|\n" + "|\tof every
reservation you wish cancel\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print (screen)

      while True:
            ticket_no = ipop.getUserData([str], "||\tEnter ticket number\n||\t",
"Wrong data")
            details = mdb.ticketDetails(ticket_no, 1)
            if details == None:
                  print("Ticket number does not exist.")

            elif username != details[2]: print("This ticket belongs to a
different customer.")

            elif calc.isValidTransactionDate(details[5]):

                  username = details[2]
                  route_id = details[0]
                  bus_id = details[1]
                  starting = details[3]
                  ending = details[4]
                  reservation_date = details[5]
                  amount = details[7]

                  tickString =
"_____________________________________________________________________\n" + "||\n"
+ "||\tReservation in bus ID: " + bus_id + "\n" + "||\tOn route ID: " +
route_id + "\n" + "||\tJourney starting from: " + starting + "\n" +
"||\tJourney ending at: " + ending + "\n" + "||\tOn date: " + reservation_date
+ '\n' + "||\tNumber of reservations = 1" + '\n||\n' + "||\tTicket numbers:\n"
+ '||\t' + ticket_no + '\n' + "||\tTotal amounting to: " + str(amount) + '\n' +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

                  print(tickString+'\n')

                  p = input("To cancel this ticket, press any key or !q to go
back....")
```

```
                if p != '!q':
                     if (mdb.add_cancellation(ticket_no) != 1):
                          print ("Ticket could not be cancelled.")
                     else: print ("Ticket cancelled.")

           else:
                print("This ticket cannot be cancelled.")

           ip = ipop.getUserData([int, str], "Enter 1 to enter another ticket
number, !q to go to main menu.", "Wrong data")
           if ip == 1:
                continue
           else:
                exit1(username)
                break

def exit1(un): os.system("python3 logged_in_main_menu.py '" + un + "'")

if __name__ == '__main__':
     main()
```

Reservation & Cancellation Details for User

```
import sys
import os
import manageDB as mdb
import ipop

def main():

     ipop.cls()

     username = sys.argv[1]

     screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tThis page lists all the cancellations done\t\t|\n"
+ "|\tfrom this account.\t\t\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

     print (screen)

     data = mdb.get_user_activity(username, 'cancellations')
     if data == None or data == '':
          print ("No cancellation data available in this account.")
          p = input("Press any key to go to main menu...")
          exit1(username)
     else:
          lines = data.split('\n')
          m = []
          for line in lines:
               m.append(tuple(line.split('_')))

          h = ['Ticket no.', 'Cancelled on', 'Bus ID', 'Source', 'Destination',
'Journey date', 'Seat no.', 'Amount']

          ipop.print_table((h, m))
```

```
            p = input("\nPress any key to go to main menu...")
            exit1(username)


def exit1(un): os.system("python3 logged_in_main_menu.py '" + un + "'")

if __name__ == '__main__':
      main()



import sys
import os
import manageDB as mdb
import ipop

def main():

      ipop.cls()

      username = sys.argv[1]

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tThis page lists all the reservations done\t\t|\n"
+ "|\tfrom this account.\t\t\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print (screen)

      data = mdb.get_user_activity(username, 'reservations')
      if data == None or data == '':
            print ("No reservation data available in this account.")
            p = input("Press any key to go to main menu...")
            exit1(username)
      else:
            lines = data.split('\n')
            m = []
            for line in lines:
                  m.append(tuple(line.split('_')))

            h = ['Ticket no.', 'Reserved on', 'Bus ID', 'Source', 'Destination',
'Journey date', 'Seat no.', 'Amount']

            ipop.print_table((h, m))

            p = input("\nPress any key to go to main menu...")
            exit1(username)


def exit1(un): os.system("python3 logged_in_main_menu.py '" + un + "'")

if __name__ == '__main__':
      main()
```

Reservation Ar istration

```
import ipop
import os
import sys
import manageDB as mdb
import calc

def main():

      ipop.cls()
      username = sys.argv[1]
      name = mdb.getNameFromUsername(username)

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t\t|\n" + "|\tHello administrator,\t\t\t\t\t|\n" + "|\t" + name
+ "\n" + "|\t\t\t\t\t\t\t\t\t|\n" + "|\tSee report on
reservation/cancellation\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print(screen)

      fdate = ''
      tdate = ''
      cat = 0
      t = ''


      while True:
            print("\tCatagories:\n\t1. Order according to bus IDs\n\t2.
According to route IDs\n\t3. According to actual routes\n")
            cat = ipop.getUserData([int], "||\tEnter category: ", "Wrong data
entered!", True, ["x in [1,2,3]"], True)
            if cat == None:
                  exit1(username)
                  break

            print("||\tEnter a 'from' date (DD/MM/YYYY): ")
            fdate = reqdate()
            if fdate == '':
                  exit1(username)
                  break

            print("||\tEnter a 'to' date (DD/MM/YYYY): ")
            tdate = reqdate()
            if tdate == '':
                  exit1(username)
                  break

            print("\tType:\n\tr. See reservations\n\tc. See cancellations\n")
            t = ipop.getUserData([str], "||\tEnter type: ", "Wrong data
entered!", True, ["x in ['r', 'c']"], True)
            if t == None:
                  exit1(username)
                  break

            print()
```

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

```
            h, c, t = mdb.order_rc_by_catagories(cat, fdate, tdate, t)
            ipop.print_table((h,c))
            print("\nTotal count: " + str(t) + '\n')

            c = ipop.getUserData([int, str], "||\tEnter 1 for re-search, !q to
cancel: ", "Wrong data entered!")
            if c != 1:
                  exit1(username)
                  break


def reqdate():
      d = ''
      while True:
            d = ipop.getUserData([str], '||\t', "Wrong data entered!")
            if d == None:
                  return ''
            else:
                  if calc.isPreviousDate(d) == False:
                        print("Wrong date format or future date entered. Re-enter
date or !q to cancel.")
                        continue
                  else: return d

def exit1(un): os.system("python3 admin_page.py '" + un + "'")

if __name__ == '__main__':
      main()




import ipop
import os
import sys
import manageDB as mdb
import calc

def main():

      ipop.cls()
      username = sys.argv[1]
      name = mdb.getNameFromUsername(username)

      screen =
"_____________________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tHello administrator,\t\t\t\t\t|\n" + "|\t" + name
+ "\n" + "|\t\t\t\t\t\t\t\t|\n" + "|\tSee report on
reservation/cancellation\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

      print(screen)

      bid = ''
      rid = ''
      source = ''
      destination = ''
      t = ''
```

```
    print("\tEnter ATLEAST bus ID, or route ID or source or destination or
their combination.\n")

    while True:
        bid = input("||\tEnter bus ID (leave blank to include all buses): ")
        if bid == '!q':
            exit1(username)
            break
        elif bid == '':
            rid = input("||\tEnter route ID (leave blank to include all
routes): ")
            if rid == '!q':
                exit1(username)
                break

        source = input("||\tEnter source (leave blank to include all
sources): ")
        if source == '!q':
            exit1(username)
            break

        destination = input("||\tEnter destination (leave blank to include
all destinations): ")
        if destination == '!q':
            exit1(username)
            break

        t = input("||\tEnter 'r' for reservation or 'c' for cancellation: ")
        if t == '!q':
            exit1(username)
            break

        print()
        h, c = mdb.order_rc_by_month(t, rid, bid, source, destination)
        if c == []:
            print("\nData unavailable.")
        else:
            ipop.print_table((h,c))

        print()
        c = ipop.getUserData([int, str], "||\tEnter 1 for re-search, !q to
cancel: ", "Wrong data entered!")
        if c != 1:
            exit1(username)
            break


def reqdate():
    d = ''
    while True:
        d = ipop.getUserData([str], '||\t', "Wrong data entered!")
        if d == None:
            return ''
        else:
            if calc.isPreviousDate(d) == False:
                print("Wrong date format or future date entered. Re-enter
date or !q to cancel.")
```

```
                else: return d

def exit1(un): os.system("python3 admin_page.py '" + un + "'")

if __name__ == '__main__':
     main()
```

Administration Details

```
import ipop
import os
import sys
import manageDB as mdb

def main():

     ipop.cls()
     username = sys.argv[1]
     name = mdb.getNameFromUsername(username)

     screen =
"_____________________________________________________________\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tHello administrator,\t\t\t\t\t|\n" + "|\t" + name
+ "\n" + "|\t\t\t\t\t\t\t\t|\n" + "|\tPlease select the appropriate
option:\t\t\t|\n" + "|\t1) Print tables\t\t\t\t\t\t|\n" + "|\t2) Print
revenue\t\t\t\t\t|\n" + "|\t3) Sort reservation/cancellation catagory-
wise\t\t|\n" + "|\t4) Sort reservation/cancellation month-wise\t\t|\n" +
"|\t\t\t\t\t\t\t\t|\n" + "|\tL) Logout\t\t\t\t\t\t|\n" +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

     print(screen)

     p = "Enter option (1-4,L): "
     ip = ipop.getUserData([int, str], p, "Wrong data entered!", True, ["x in
list(range(1,5))", "x == 'L'"], True)

     if ip == 1:
          os.system("python3 display_tables.py '" + username + "'")
     if ip == 2:
          os.system("python3 display_revenue.py '" + username + "'")
     if ip == 3:
          os.system("python3 sort_rc_cat.py '" + username + "'")
     if ip == 4:
          os.system("python3 sort_rc_month.py '" + username + "'")
     elif ip == 'L':
          os.system("python3 main_menu.py")

if __name__ == '__main__':
     main()
```

CERTIFICATE

This is to certify that:

AAYUSH GADIA	100436	UNIVERSITY OF KALYANI
AVIK DUTTA	1001410899	TECHNO INDIA UNIVERSITY
MOINAK NANDI	304201500900605	UEM
SAYANTAN ROYCHOWDHURY	151040110481	IEM
SHUBHAM OMKAR	151150110101	BPPIMT
SUMIT RAY	151260110166	HITK
VIKASH KUMAR CHOUDHARY	151150110119	BPPIMT

have successfully completed a project on Bus Ticket Reservation System using Python under the guidance of Mr. Prasun Neogy.

Mr. Prasun Neogy

Globsyn Finishing School

Prasun Neogy

Globsyn Knowledge Foundation Kolkata

Document sign date :Jul 26, 2017