**1. INTRODUCTION**

Online Cab Booking/Hiring is a method that may be utilized for a price for a limited length of time. People who do not have access to their own personal automobile or do not own one at all can travel around by renting a cab. Individuals who wish to hire or booking an automobile must first contact the desired vehicle's cab booking business. the online cab booking system in python with source code project based on cab booking system which uses python language with Tkinter library for GUI. In this project, all systems allow the only user panel can make the bill and insert the customer details the user has to fill up the details such as name, contact, details, address information, select cabs, pick up location destination, pooling etc. This cab booking system project in python mainly focus on dealing with customer’s booking details with their respective bill amounts may be done over the internet. This individual must now provide certain information, such as the booked dates and automobile type. Following the completion of these data, the person booking the automobile must produce a valid identification card. The majority of enterprises in the sector earn based on the sort of automobiles they sell. Customers can use an online booking system to booked cabs. Customers may use this online system to find available cabs, register cabs, see profiles, and book cabs. Cab booking is a typical kind of transportation that is offered by a number of different transportation forms in a particular city. The bulk of people rely on cab services for their daily transportation needs. The company must be registered and fulfill all of the transportation department's requirements and security requirements. The Online Cab Booking System is a tkinter library for GUI based that allows your customers to order cab and executive cabs from their own home or office .

* 1. **SYSTEM SPECIFICATION**

1.1.1 HARDWARE SPECIFICATION

The section of hardware configuration is an important task related to the software Development insufficient random Access memory may affect adversely on the speed and efficiency of the entire system. the process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and application.

* System : Intel(R)Core(TM)i57500cpu3.40GHz
* Hard Disk : 500 GB
* Ram : 4GB
  + 1. SOFTWARE SPECIFICATION

This section gives the details of the software that are used for the development

* Operating System : Windows 10 pro
* Coding Language : python
* Application : Python Idel
  + 1. SOFTWARE DESCRIPTION

PYTHON

Python is an object-oriented high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is relatively simple, so it’s easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate python code much easier than other languages.

Additionally, python supports the use of modules and Packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. as Python and all the necessary tools are available on all major platforms.

READABLE AND MAINTAINABLE CODE

While writing software application, we must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of python allow we to express concepts without writing additional code. At the same time in Python we can use python to build custom applications without writing additional code. The readable and clean code base will help we to maintain and update the software without putting extra time and effort.

PYTHON IDLE

IDLE (Integrated Development And Learning Environment) is an integrated development environment (IDE) for python. The python installer for windows contains the IDLE module by default.

IDLE is not available by default in python distributions for Linux. It needs to be installed using the respective package managers.

IDLE can be used to execute a single statement just like python shell and also to create, modify, and execute python scripts. Its provides a fully featured text editor to create, python scripts that includes features like syntax highlighting, autocompletion, and smart indent. It also has a debugger with stepping and breakpoints features.

The feature makes it easier for you to make changes to the code without increasing development time.

MANY OPEN SOURCE FRAMEWORKS AND TOOLS

As an open source programming languages, python helps we to curtail software development cost significantly for instance, you can simplify and speedup web application development by using robust python web frameworks like Django , flask, pyramid, bottle and cherrypy, likewise, you can accelerate desktop in python.

BENEFITS OF LEARNING PYTHON

They are many benefits of learning python, especially as your first language, which we will discuss.

Python is widely used, including by a number of big companies

1)python can be used to develop prototypes,and quickly because it is so easy to work with and read

2)Most automation, datamining ,and big data platforms rely on python.this is because it is the ideal language to work with for general tasks.

3) python allows for a more productive coding environment than massive languages like c# and java. Experienced coders tend to stay more organized and productive when working with python.

**2. SYSTEM STUDY**

2.1 EXSITING SYSTEM

2.1.1 DESCRIPTION

This system developed under four lead entities. This Project is based on MEAN stack Technologies. The strategy in web development has changed dramatically in the recent years, thanks to the improvement of information technology. The browser is no longer used to serve static information. Python is used heavily to serve dynamic elements in the browser. The browser is becoming a kind of mini-operating system . The modern web application, which is called web app, connects to the server to retrieve data dynamically. It only exists in the browser. When users approach the app for the first time, the app is rendered by the server and supports its services such as database access or transaction. Deriving from modern web application, Wire delta’s internal project is designed to build application service provider called an APIs (Application Programming Interfaces) and use python as formatting language. This project is based on MEAN stack. Mongo DB and My SQL both Databases are used to shape this project. The Project consists of several ideas and functionality. it is very powerful and helpful for this project.

SIGNUP PROCESS

The basic design architecture of your application is to keep everything simple and user friendly. The same should hold true for sign-up page. Before using the application, it is mandatory to have a registered account with the platform. The overall process should be minimalistic keeping in mind the mobile nature, and requests only most critical information such as:

* Name
* Email
* Password
* Mobile number
* Date of Birth
* Identity Number
* Car details (In the case of driver)

USER CENTRIC BOOKING PROCESS

After the completion of registration process, the users can be taken to the main screen where they can easily pick a ride for themselves. Here are the most noticeable points in the cab booking process

* Search option to enter a desired pickup location.
* GPS feature to detect current location.
* A detailed map to see their selected location.

Locations saving feature User can also select and save their favourite location which can be accessed easily.

2.1.2 DRAWBACKS

* The main drawback to app-based cab services is that they're destroying the cab industry.
* The major way which these apps complete with each other is through lowering their prices, which means the original black cab and self-employed car drivers are losing out.

2.2 PROPOSED SYSTEM

2.2.1 DESCRIPTION

The online **Cab Booking System Project In Python** is a fully functional desktop application developed in Python that covers all of the features that IT students and computer-related courses will require for their college projects or assignments. This Car Booking is one of the important systems for organizations who have been having trouble booking every passenger because it offers features that allow all users to interact in the same way that cab drivers connect with their clients about their cab or car reservations. This online **Cab Booking System In Python**is quite useful, and the concept and logic of the project are simple to grasp. all the user has to do is fill in their name, contact information, and address. They can then choose cabs, pick-up and drop-off locations, pooling, etc. Also, the system figures out the total cost, tax, and subtotal based on the information the user gives. The user can also see an over all the receipt with their information, including the date, cab number, total cost, and other details.

2.2.2 FEATURE

* Login system
* Set customer information
* Set booking details
* Select cabs
* Generates total cost with tax
* Booking Receipt

**\**

**3. SYSTEM DEVELOPMENT**

3.1 INTRODUCTION

The online cab booking system is a web based project that has been created on python language. in this project Is the first administrator part and the second Is user part. Login here with admin id and password in admin part.

Admin Id= “admin”

Password= ”admin”

After the login. the administrator can add cabs for booking with All the information for the cab. In the user part user can view different types of cabs for booking after this user can check availability with cab, and booking with fill all the Mandatory details.

3.2 DESCRIPTION OF MODULES

The general theme behind a database is to handle information as an integrated whole, a database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and effectively after designing input output. The general objective is to make information access , easy quick, inexpensive and flexible for other users.

* Tkinter Gui module
* Python SQLITE
* Time module

TKINTER GUI MODULE

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task.

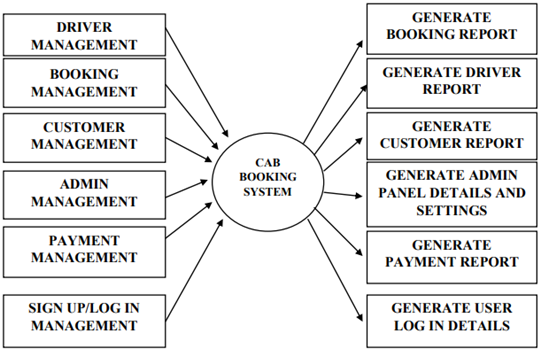
PYTHON SQLITE

SQLite is a self-contained, file-based SQL database. SQLite comes bundled with Python and can be used in any of your Python applications without having to install any additional software.

TIME MODULE

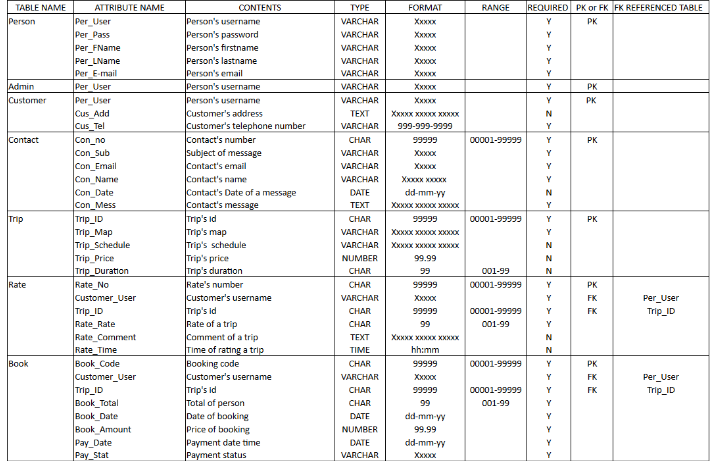
All the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module we need to import it.

3.3 SYSTEM ARCHITECTURE



**4. IMPLEMENTATION**

4.1 DATASET



4.2. SAMPLE CODING

from tkinter import \*

from tkinter import ttk

import random

import time

import datetils

from tkinter import messagebox as ms

import sqlite3

Item4 = 0

# make database and users (if not exists already) table at programme start up

with sqlite3.connect('Users.db') as db:

c = db.cursor()

c.execute('CREATE TABLE IF NOT EXISTS user (username TEXT NOT NULL ,password TEXT NOT NULL)')

db.commit()

db.close()

#main Class

class user:

def \_\_init\_\_(self,master):

# Window

self.master = master

# Some Usefull variable

self.username = StringVar()

self.password = StringVar()

self.n\_username = StringVar()

self.n\_password = StringVar()

#Create Widgets

self.widgets()

#Login Function

def login(self):

#Establish Connection

with sqlite3.connect('Users.db') as db:

c = db.cursor()

#Find user If there is any take proper action

find\_user = ('SELECT \* FROM user WHERE username = ? and password = ?')

c.execute(find\_user,[(self.username.get()),(self.password.get())])

result = c.fetchall()

if result:

self.logf.pack\_forget()

self.head['text'] = "Welcome " + self.username.get()

self.head.configure(fg="black")

self.head.pack(fill=X)

application = travel(root)

else:

ms.showerror('Oops!','Username Not Found.')

def new\_user(self):

#Establish Connection

with sqlite3.connect('Users.db') as db:

c = db.cursor()

#Find Existing username if any take proper action

find\_user = ('SELECT \* FROM user WHERE username = ?')

c.execute(find\_user,[(self.username.get())])

if c.fetchall():

ms.showerror('Error!','Username Already Taken!')

else:

ms.showinfo('Success!','Account Created!')

self.log()

#Create New Account

insert = 'INSERT INTO user(username,password) VALUES(?,?)'

c.execute(insert,[(self.n\_username.get()),(self.n\_password.get())])

db.commit()

#Frame Packing Methords

def log(self):

self.username.set('')

self.password.set('')

self.crf.pack\_forget()

self.head['text'] = 'Login'

self.logf.pack()

def cr(self):

self.n\_username.set('')

self.n\_password.set('')

self.logf.pack\_forget()

self.head['text'] = 'Create Account'

self.crf.pack()

#Draw Widgets

def widgets(self):

self.head = Label(self.master,text = 'Login Panel',font = ('',30),pady = 10)

self.head.pack()

self.logf = Frame(self.master,padx =10,pady = 10)

Label(self.logf,text = 'Username: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

Entry(self.logf,textvariable = self.username,bd = 5,font = ('',15)).grid(row=0,column=1)

Label(self.logf,text = 'Password: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

Entry(self.logf,textvariable=self.password,bd=5,font=('',15),show= '\*').grid(row=1,column=1)

Button(self.logf,text='Login',bd=3,font=('',15),padx=5,pady=5,command=self.login).grid()

Button(self.logf,text='CreateAccount',bd=3,font=('',15),padx=5,pady=5,command=self.cr).grid(row=2,column=1)

self.logf.pack()

self.crf = Frame(self.master,padx =10,pady = 10)

Label(self.crf,text = 'Username: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

Entry(self.crf,textvariable= self.n\_username,bd = 5,font = ('',15)).grid(row=0,column=1)

Label(self.crf,text = 'Password: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

Entry(self.crf,textvariable=self.n\_password,bd=5,font=('',15),show='\*').grid(row=1,column=1)

Button(self.crf,text='CreateAccount',bd=3,font=('',15),padx=5,pady=5,command=self.new\_user).grid()

Button(self.crf,text='GotoLogin',bd=3,font=('',15),padx=5,pady=5,command=self.log).grid(row=2,column=1)

class travel:

def \_\_init\_\_(self,root):

self.root = root

self.root.title("Cab Booking System")

self.root.geometry(geometry)

self.root.configure(background='black')

DateofOrder=StringVar()

DateofOrder.set(time.strftime(" %d / %m / %Y "))

Receipt\_Ref=StringVar()

PaidTax=StringVar()

SubTotal=StringVar()

TotalCost=StringVar()

var1=IntVar()

var2=IntVar()

var3=IntVar()

var4=IntVar()

journeyType=IntVar()

carType=IntVar()

varl1=StringVar()

varl2=StringVar()

varl3=StringVar()

reset\_counter=0

Firstname=StringVar()

Surname=StringVar()

Address=StringVar()

Postcode=StringVar()

Mobile=StringVar()

Telephone=StringVar()

Email=StringVar()

CabTax=StringVar()

Km=StringVar()

Travel\_Ins=StringVar()

Luggage=StringVar()

Receipt=StringVar()

Standard=StringVar()

FordGalaxy=StringVar()

FordMondeo=StringVar()

CabTax.set("0")

Km.set("0")

Travel\_Ins.set("0")

Luggage.set("0")

Standard.set("0")

FordGalaxy.set("0")

FordMondeo.set("0")

def iExit():

iExit= ms.askyesno("Prompt!","Do you want to exit?")

if iExit > 0:

root.destroy()

return

def Reset():

CabTax.set("0")

Km.set("0")

Travel\_Ins.set("0")

Luggage.set("0")

Standard.set("0")

FordGalaxy.set("0")

FordMondeo.set("0")

Firstname.set("")

Surname.set("")

Address.set("")

Postcode.set("")

Mobile.set("")

Telephone.set("")

Email.set("")

PaidTax.set("")

SubTotal.set("")

TotalCost.set("")

self.txtReceipt1.delete("1.0",END)

self.txtReceipt2.delete("1.0",END)

var1.set(0)

var2.set(0)

var3.set(0)

var4.set(0)

journeyType.set(0)

carType.set(0)

varl1.set("0")

varl2.set("0")

varl3.set("0")

self.cboPickup.current(0)

self.cboDrop.current(0)

self.cboPooling.current(0)

self.txtCabTax.configure(state=DISABLED)

self.txtKm.configure(state=DISABLED)

self.txtTravel\_Ins.configure(state=DISABLED)

self.txtLuggage.configure(state=DISABLED)

self.txtStandard.configure(state=DISABLED)

self.txtFordGalaxy.configure(state=DISABLED)

self.txtFordMondeo.configure(state=DISABLED)

self.reset\_counter=1

def Receiptt():

if reset\_counter == 0 and Firstname.get()!="" and Surname.get()!="" and Address.get()!="" and Postcode.get()!="" and Mobile.get()!="" and Telephone.get()!="" and Email.get()!="":

self.txtReceipt1.delete("1.0",END)

self.txtReceipt2.delete("1.0",END)

x=random.randint(10853,500831)

randomRef = str(x)

Receipt\_Ref.set(randomRef)

self.txtReceipt1.insert(END,"Receipt Ref:\n")

self.txtReceipt2.insert(END, Receipt\_Ref.get() + "\n")

self.txtReceipt1.insert(END,'Date:\n')

self.txtReceipt2.insert(END, DateofOrder.get() + "\n")

self.txtReceipt1.insert(END,'Cab No:\n')

self.txtReceipt2.insert(END, 'TR ' + Receipt\_Ref.get() + " BW\n")

self.txtReceipt1.insert(END,'Firstname:\n')

self.txtReceipt2.insert(END, Firstname.get() + "\n")

self.txtReceipt1.insert(END,'Surname:\n')

self.txtReceipt2.insert(END, Surname.get() + "\n")

self.txtReceipt1.insert(END,'Address:\n')

self.txtReceipt2.insert(END, Address.get() + "\n")

self.txtReceipt1.insert(END,'Postal Code:\n')

self.txtReceipt2.insert(END, Postcode.get() + "\n")

self.txtReceipt1.insert(END,'Telephone:\n')

self.txtReceipt2.insert(END, Telephone.get() + "\n")

self.txtReceipt1.insert(END,'Mobile:\n')

self.txtReceipt2.insert(END, Mobile.get() + "\n")

self.txtReceipt1.insert(END,'Email:\n')

self.txtReceipt2.insert(END, Email.get() + "\n")

self.txtReceipt1.insert(END,'From:\n')

self.txtReceipt2.insert(END, varl1.get() + "\n")

self.txtReceipt1.insert(END,'To:\n')

self.txtReceipt2.insert(END, varl2.get() + "\n")

self.txtReceipt1.insert(END,'Pooling:\n')

self.txtReceipt2.insert(END, varl3.get() + "\n")

self.txtReceipt1.insert(END,'Standard:\n')

self.txtReceipt2.insert(END, Standard.get() + "\n")

self.txtReceipt1.insert(END,'Prime Sedan:\n')

self.txtReceipt2.insert(END, FordGalaxy.get() + "\n")

self.txtReceipt1.insert(END,'Premium Sedan:\n')

self.txtReceipt2.insert(END, FordMondeo.get() + "\n")

self.txtReceipt1.insert(END,'Paid:\n')

self.txtReceipt2.insert(END, PaidTax.get() + "\n")

self.txtReceipt1.insert(END,'SubTotal:\n')

self.txtReceipt2.insert(END, str(SubTotal.get()) + "\n")

self.txtReceipt1.insert(END,'Total Cost:\n')

self.txtReceipt2.insert(END, str(TotalCost.get()))

else:

self.txtReceipt1.delete("1.0",END)

self.txtReceipt2.delete("1.0",END)

self.txtReceipt1.insert(END,"\nNo Input")

def Cab\_Tax():

global Item1

if var1.get() == 1:

self.txtCabTax.configure(state = NORMAL)

Item1=float(50

CabTax.set("Rs " + str(Item1))

elif var1.get() == 0:

self.txtCabTax.configure(state=DISABLED)

CabTax.set("0")

Item1=0

def Kilo():

if var2.get() == 0:

self.txtKm.configure(state=DISABLED)

Km.set("0")

elif var2.get() == 1 and varl1.get() != "" and varl2.get() != "":

self.txtKm.configure(state=NORMAL)

if varl1.get() == "BleckerStreet":

switch={"BrownAvenue":10,"NorthAvenue":8,"BoggessStreet":6,"BleckerStreet":0 Km.set(switch[varl2.get()])

elif varl1.get() == "BrownAvenue":

switch={"BrownAvenue":0,"NorthAvenue":2,"BoggessStreet":5,"BleckerStreet":1 Km.set(switch[varl2.get()])

elif varl1.get() == "NorthAvenue":

switch ={"BrownAvenue": 2,"NorthAvenue": 0,"BoggessStreet":3,"BleckerStreet":8}

Km.set(switch[varl2.get()])

elif varl1.get() == "BoggessStreet":

switch ={"BrownAvenue": 5,"NorthAvenue": 3,"BoggessStreet":0,"BleckerStreet": 6}

Km.set(switch[varl2.get()])

def Travelling():

global Item3

if var3.get() == 1:

self.txtTravel\_Ins.configure(state = NORMAL)

Item3=float(10)

Travel\_Ins.set("Rs " + str(Item3))

elif var3.get() == 0:

self.txtTravel\_Ins.configure(state = DISABLED)

Travel\_Ins.set("0")

Item3=0

def Lug():

global Item4

if (var4.get()==1):

self.txtLuggage.configure(state = NORMAL)

Item4=float(30)

Luggage.set("Rs "+ str(Item4))

elif var4.get()== 0:

self.txtLuggage.configure(state = DISABLED)

Luggage.set("0")

Item4=0

def selectCar():

global Item5

if carType.get() == 1:

self.txtFordGalaxy.configure(state = DISABLED)

FordGalaxy.set("0")

self.txtFordMondeo.configure(state = DISABLED)

FordMondeo.set("0")

self.txtStandard.configure(state = NORMAL)

Item5 = float(8)

Standard.set("Rs "+ str(Item5))

elif carType.get() == 2:

self.txtStandard.configure(state =DISABLED)

Standard.set("0")

self.txtFordMondeo.configure(state = DISABLED)

FordMondeo.set("0")

self.txtFordGalaxy.configure(state = NORMAL)

Item5 = float(15)

FordGalaxy.set("Rs "+ str(Item5))

else:

self.txtStandard.configure(state =DISABLED)

Standard.set("0")

self.txtFordGalaxy.configure(state = DISABLED)

FordGalaxy.set("0")

self.txtFordMondeo.configure(state = NORMAL)

Item5 = float(22)

FordMondeo.set("Rs "+ str(Item5))

def Total\_Paid():

if ((var1.get() == 1 and var2.get() == 1 and var3.get() == 1 or var4.get() == 1) and carType.get() != 0 and journeyType.get() != 0 and (varl1.get() != "" and varl2.get() !="")):

if journeyType.get()==1:

Item2=Km.get()

Cost\_of\_fare = (Item1+(float(Item2)\*Item5)+Item3+Item4)

Tax = "Rs " + str('%.2f'%((Cost\_of\_fare) \*0.09))

ST = "Rs " + str('%.2f'%((Cost\_of\_fare)))

TT = "Rs " + str('%.2f'%(Cost\_of\_fare+((Cost\_of\_fare)\*0.9)))

elif journeyType.get()==2:

Item2=Km.get()

Cost\_of\_fare = (Item1+(float(Item2)\*Item5)\*1.5+Item3+Item4)

Tax = "Rs " + str('%.2f'%((Cost\_of\_fare) \*0.09))

ST = "Rs " + str('%.2f'%((Cost\_of\_fare)))

TT = "Rs " + str('%.2f'%(Cost\_of\_fare+((Cost\_of\_fare)\*0.9)))

else:

Item2=Km.get()

Cost\_of\_fare = (Item1+(float(Item2)\*Item5)\*2+Item3+Item4)

Tax = "Rs " + str('%.2f'%((Cost\_of\_fare) \*0.09))

ST = "Rs " + str('%.2f'%((Cost\_of\_fare)))

TT = "Rs " + str('%.2f'%(Cost\_of\_fare+((Cost\_of\_fare)\*0.9)))

PaidTax.set(Tax)

SubTotal.set(ST)

TotalCost.set(TT)

else:

w = ms.showwarning("Error !","Invalid Input\nPlease try again !!!") MainFrame=Frame(self.root)

MainFrame.pack(fill=BOTH,expand=True)

Tops = Frame(MainFrame, bd=10, width=1350,relief=RIDGE)

Tops.pack(side=TOP,fill=BOTH)

self.lblTitle=Label(Tops,font=('arial',50,'bold'),text="\t Cab Booking System ")

self.lblTitle.grid()

CustomerDetailsFrame=LabelFrame(MainFrame,width=1350,height=500,bd=20, pady=5, relief=RIDGE)

CustomerDetailsFrame.pack(side=BOTTOM,fill=BOTH,expand=True

FrameDetails=Frame(CustomerDetailsFrame,width=880,height=400,bd=10, relief=RIDGE)

FrameDetails.pack(side=LEFT,fill=BOTH,expand=True)

CustomerName=LabelFrame(FrameDetails,width=150,height=250,bd=10, font=('arial',12,'bold'),text="Customer Info", relief=RIDGE)

CustomerName.grid(row=0,column=0)

TravelFrame=LabelFrame(FrameDetails,bd=10,width=300,height=250, font=('arial',12,'bold'),text="Booking Detail", relief=RIDGE)

TravelFrame.grid(row=0,column=1)

Book\_Frame=LabelFrame(FrameDetails,width=300,height=150,relief=FLAT)

Book\_Frame.grid(row=1,column=0)

CostFrame = LabelFrame(FrameDetails,width=150,height=150,bd=5,relief=FLAT)

CostFrame.grid(row=1,column=1)

Receipt\_BottonFrame=LabelFrame(CustomerDetailsFrame,bd=10,width=450,height=400, relief=RIDGE)

Receipt\_BottonFrame.pack(side=RIGHT,fill=BOTH,expand=True)

ReceiptFrame=LabelFrame(Receipt\_BottonFrame,width=350,height=300, font=('arial',12,'bold'),text="Receipt", relief=RIDGE)

ReceiptFrame.grid(row=0,column=0)

ButtonFrame=LabelFrame(Receipt\_BottonFrame, width=350,height=100, relief=RIDGE)

ButtonFrame.grid(row=1,column=0)

self.lblFirstname=Label(CustomerName,font=('arial',14,'bold'),text="Firstname",bd=7)

self.lblFirstname.grid(row=0,column=0,sticky=W)

self.txtFirstname=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Firstname,bd=7,insertwidth=2,justify=RIGHT)

self.txtFirstname.grid(row=0,column=1)

self.lblSurname=Label(CustomerName,font=('arial',14,'bold'),text="Surname",bd=7)

self.lblSurname.grid(row=1,column=0,sticky=W)

self.txtSurname=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Surname,bd=7,insertwidth=2,justify=RIGHT)

self.txtSurname.grid(row=1,column=1,sticky=W)

self.lblAddress=Label(CustomerName,font=('arial',14,'bold'),text="Address",bd=7)

self.lblAddress.grid(row=2,column=0,sticky=W)

self.txtAddress=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Address,bd=7,insertwidth=2,justify=RIGHT)

self.txtAddress.grid(row=2,column=1)

self.lblPostcode=Label(CustomerName,font=('arial',14,'bold'),text="Postcode",bd=7)

self.lblPostcode.grid(row=3,column=0,sticky=W)

self.txtPostcode=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Postcode,bd=7,insertwidth=2,justify=RIGHT)

self.txtPostcode.grid(row=3,column=1)

self.lblTelephone=Label(CustomerName,font=('arial',14,'bold'),text="Telephone",bd=7)

self.lblTelephone.grid(row=4,column=0,sticky=W)

self.txtTelephone=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Telephone,bd=7,insertwidth=2,justify=RIGHT)

self.txtTelephone.grid(row=4,column=1)

self.lblMobile=Label(CustomerName,font=('arial',14,'bold'),text="Mobile",bd=7)

self.lblMobile.grid(row=5,column=0,sticky=W)

self.txtMobile=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Mobile,bd=7,insertwidth=2,justify=RIGHT)

self.txtMobile.grid(row=5,column=1)

self.lblEmail=Label(CustomerName,font=('arial',14,'bold'),text="Email",bd=7)

self.lblEmail.grid(row=6,column=0,sticky=W)

self.txtEmail=Entry(CustomerName,font=('arial',14,'bold'),textvariable=Email,bd=7,insertwidth=2,justify=RIGHT)

self.txtEmail.grid(row=6,column=1)

self.lblPickup=Label(TravelFrame,font=('arial',14,'bold'),text="Pickup",bd=7)

self.lblPickup.grid(row=0,column=0,sticky=W)

self.cboPickup=ttk.Combobox(TravelFrame,textvariable=varl1,state='readonly', font=('arial',20,'bold'), width=14)

self.cboPickup['value']=('','BleckerStreet','BoggessStreet','NorthAvenue','BrownAvenue')

self.cboPickup.current(0)

self.cboPickup.grid(row=0,column=1)

self.lblDrop=Label(TravelFrame,font=('arial',14,'bold'),text="Drop",bd=7)

self.lblDrop.grid(row=1,column=0,sticky=W)

self.cboDrop =ttk.Combobox(TravelFrame, textvariable = varl2 , state='readonly', font=('arial',20,'bold'), width=14)

self.cboDrop['value']=('','BrownAvenue','NorthAvenue','BleckerStreet','BoggessStreet')

self.cboDrop.current(0)

self.cboDrop.grid(row=1,column=1)

self.lblPooling=Label(TravelFrame,font=('arial',14,'bold'),text="Pooling",bd=7)

self.lblPooling.grid(row=2,column=0,sticky=W)

self.cboPooling=ttk.Combobox(TravelFrame,textvariable=varl3,state='readonly', font=('arial',20,'bold'), width=14)

self.cboPooling['value']=('','1','2','3','4')

self.cboPooling.current(1)

self.cboPooling.grid(row=2,column=1)

self.chkCabTax=Checkbutton(TravelFrame,text="BaseCharge\*",variable=var1, onvalue=1,offvalue=0,font=('arial',16,'bold'),command=Cab\_Tax).grid(row=3,column=0, sticky=W)

self.txtCabTax=Label(TravelFrame,font=('arial',14,'bold'),textvariable=CabTax,bd=6,width=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN)

self.txtCabTax.grid(row=3,column=1)

self.chkKm=Checkbutton(TravelFrame,text="Distance(KMs)\*",variable=var2, onvalue=1,offvalue=0,font=('arial',16,'bold'),command=Kilo).grid(row=4,column=0, sticky=)

self.txtKm=Label(TravelFrame,font=('arial',14,'bold'),textvariable=Km,bd=6,width=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN,highlightthickness=0)

self.txtKm.grid(row=4,column=1)

self.chkTravel\_Ins=Checkbutton(TravelFrame,text="Travelling Insurance \*",variable = var3, onvalue=1, offvalue=0,font=('arial',16,'bold'),command=Travelling).grid(row=5, column=0, sticky=W)

self.txtTravel\_Ins=Label(TravelFrame,font=('arial',14,'bold'),textvariable=Travel\_Ins,bd=6,width=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN)

self.txtTravel\_Ins.grid(row=5,column=1)

self.chkLuggage=Checkbutton(TravelFrame,text="ExtraLuggage",variable=var4, onvalue=1,offvalue=0,font=('arial',16,'bold'),command=Lug).grid(row=6,column=0, sticky=W)

self.txtLuggage=Label(TravelFrame,font=('arial',14,'bold'),textvariable=Luggage,bd=6,width=18,bg="white",state= DISABLED,justify=RIGHT,relief=SUNKEN)

self.txtLuggage.grid(row=6,column=1)

self.lblPaidTax=Label(CostFrame,font=('arial',14,'bold'),text="Paid Tax\t\t",bd=7)

self.lblPaidTax.grid(row=0,column=2,sticky=W)

self.txtPaidTax = Label(CostFrame,font=('arial',14,'bold'),textvariable=PaidTax,bd=7, width=10, justify=RIGHT,bg="white",relief=SUNKEN)

self.txtPaidTax.grid(row=0,column=3)

self.lblSubTotal=Label(CostFrame,font=('arial',14,'bold'),text="Sub Total",bd=7)

self.lblSubTotal.grid(row=1,column=2,sticky=W)

self.txtSubTotal = Label(CostFrame,font=('arial',14,'bold'),textvariable=SubTotal,bd=7, width=10, justify=RIGHT,bg="white",relief=SUNKEN

self.txtSubTotal.grid(row=1,column=3)

self.lblTotalCost=Label(CostFrame,font=('arial',14,'bold'),text="Total Cost",bd=7)

self.lblTotalCost.grid(row=2,column=2,sticky=W)

self.txtTotalCost=Label(CostFrame,font=('arial',14,'bold'),textvariable=TotalCost,bd=7,width=10, justify=RIGHT,bg="white",relief=SUNKEN)

self.txtTotalCost.grid(row=2,column=3)

self.chkStandard=Radiobutton(Book\_Frame,text="Standard Cab",value=1,variable = carType,font=('arial',14,'bold'),command=selectCar).grid(row=0, column=0, sticky=W)

self.txtStandard=Label(Book\_Frame,font=('arial',14,'bold'),width=7,textvariable=Standard,b,state=DISABLED, justify=RIGHT,bg="white",relief=SUNKEN)

self.txtStandard.grid(row=0,column=1)

self.chkFordGalaxyd=Radiobutton(Book\_Frame,text="FordGalaxyCab",value=2,variable=carType,font=('arial',14,'bold'),command=selectCar).grid(row=1, column=0, sticky=W)

self.txtFordGalaxy=Label(Book\_Frame,font=('arial',14,'bold'),width=7,textvariable=FordGalaxy,bd=5,state=DISABLED,justify=RIGHT,bg="white",relief=SUNKEN)

self.txtFordGalaxy.grid(row=1,column=1)

self.chkFordMondeo=Radiobutton(Book\_Frame,text="FordMondeo Cab",value=3,variable=carType,font=('arial',14,'bold'),command=selectCar).grid(row=2, column=0)

self.txtFordMondeo=Label(Book\_Frame,font=('arial',14,'bold'),width=7,textvariable=FordMondeo,bd=5,state=DISABLED,justify=RIGHT,bg="white",relief=SUNKEN)

self.txtFordMondeo.grid(row=2,column=1)

self.chkSingl=Radiobutton(Book\_Frame,text="Single",value=1,variable=journeyType,font=('arial',14,'bold')).grid(row=0, column=2, sticky=W)

self.chkReturn=Radiobutton(Book\_Frame,text="Return",value=2,variable=journeyType,font=('arial',14,'bold')).grid(row=1, column=2, sticky=W)

self.chkSpecialsNeeds =Radiobutton(Book\_Frame,text="SpecialNeeds",value=3,variable = journeyType,font=('arial',14,'bold')).grid(row=2, column=2, sticky=W)

self.txtReceipt1=Text(ReceiptFrame,width=22,height=21,font=('arial',10,'bold'),borderwidth=0)

self.txtReceipt1.grid(row=0,column=0,columnspan=2)

self.txtReceipt2=Text(ReceiptFrame,width=22,height=21,font=('arial',10,'bold'),borderwidth=0)

self.txtReceipt2.grid(row=0,column=2,columnspan=2)

if \_\_name\_\_=='\_\_main\_\_':

root = Tk()

w = root.winfo\_screenwidth()

h = root.winfo\_screenheight()

geometry="%dx%d+%d+%d"%(w,h,0,0)

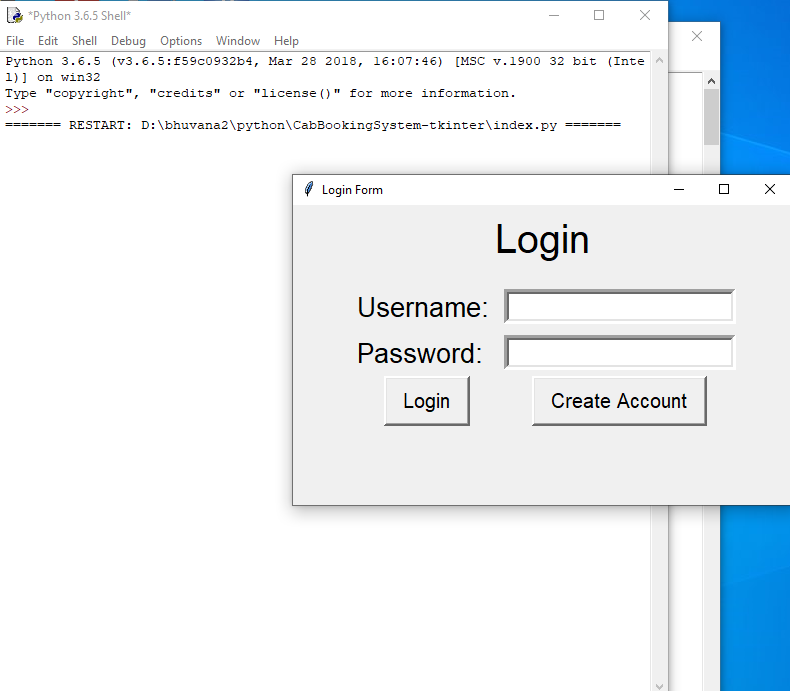
root.geometry("500x300+320+200")

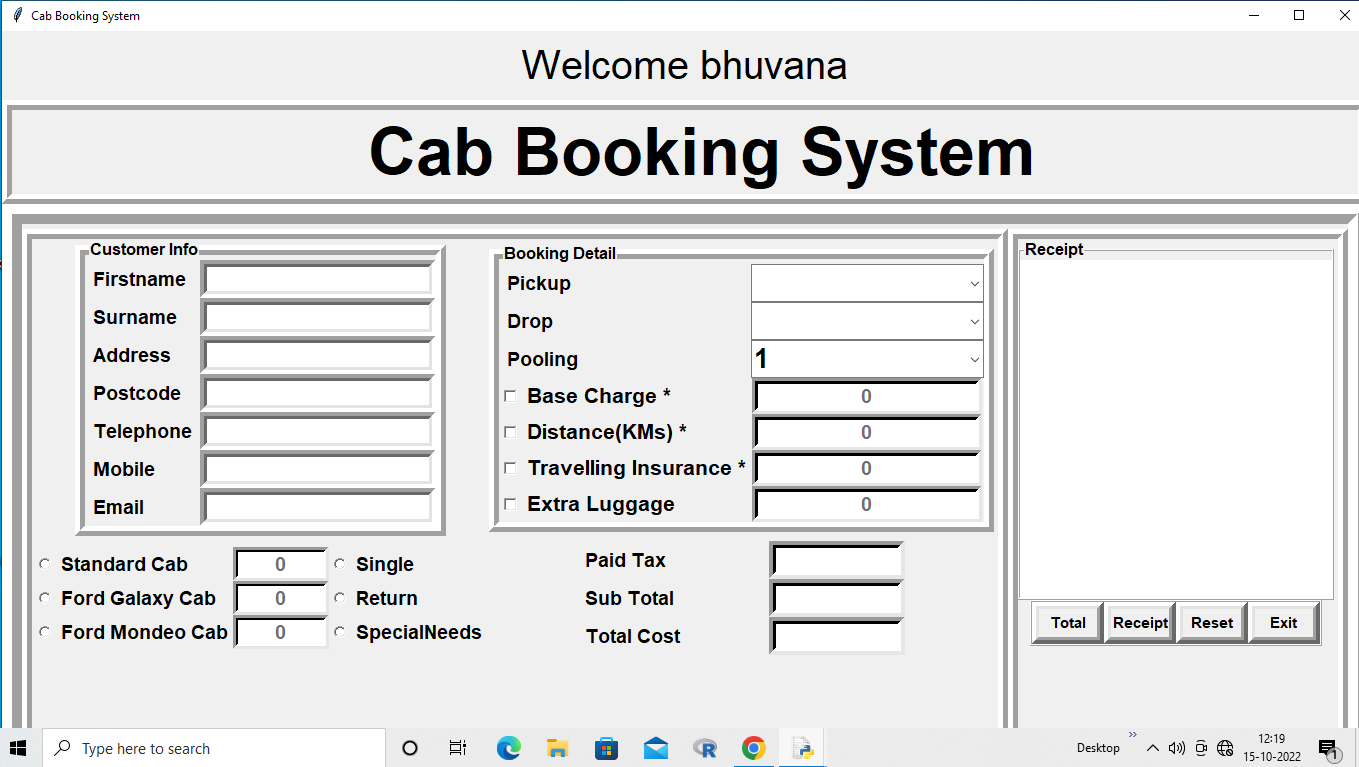
root.title('Login Form')

application = user(root)

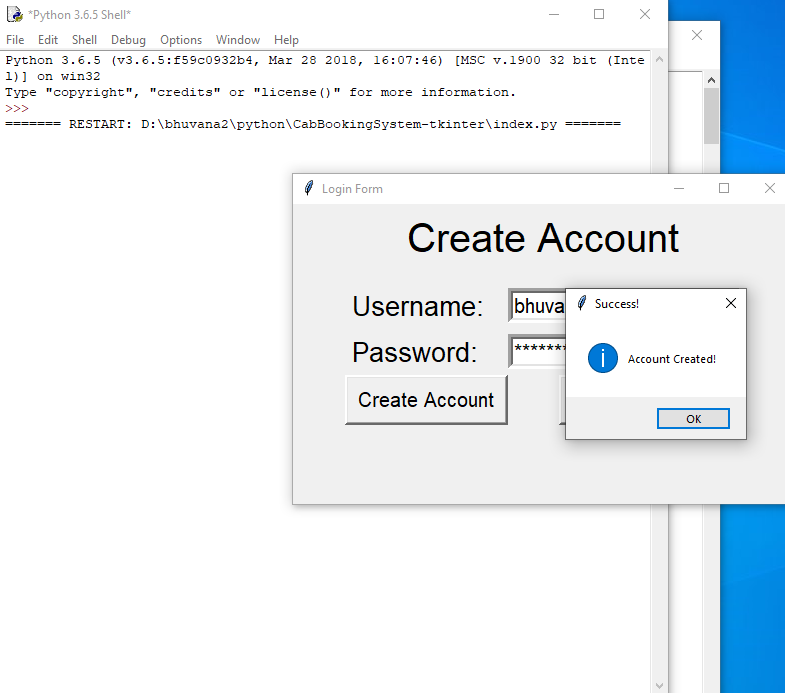
root.mainloop()

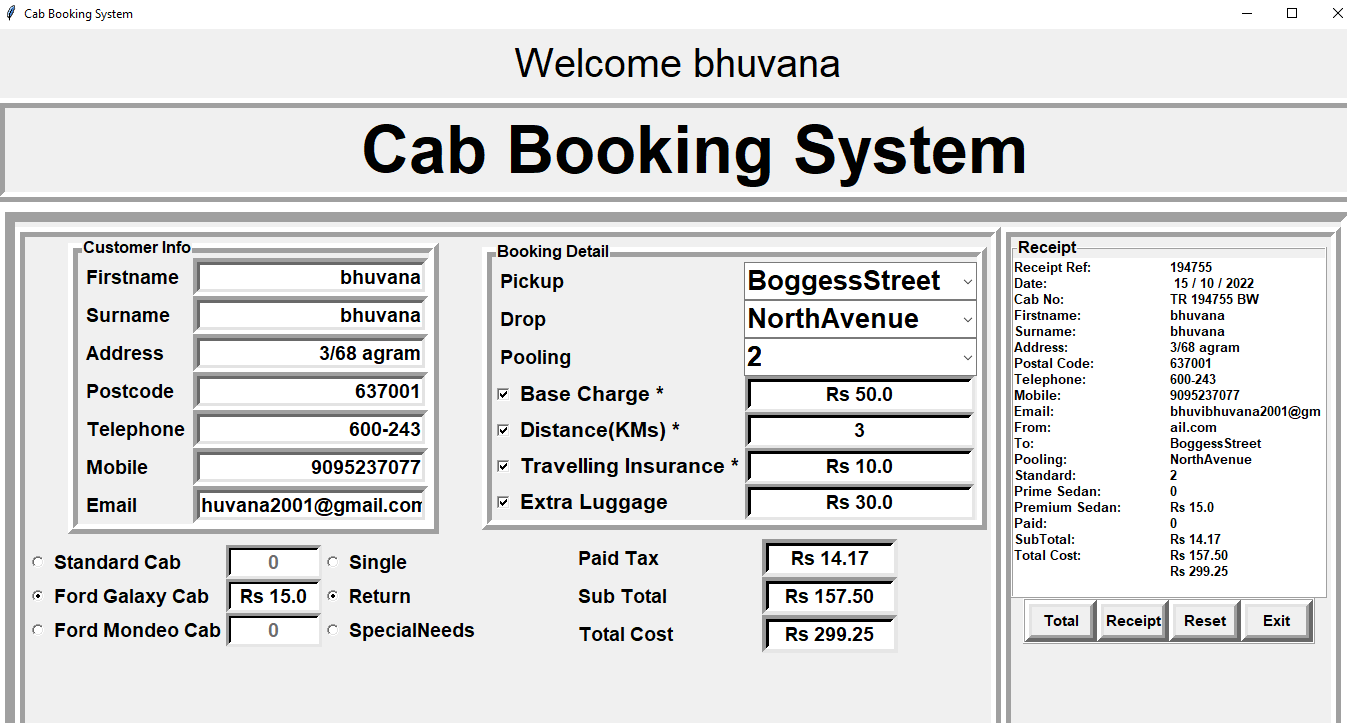
4.2.SAMPLE INPUT





4.3 SAMPLE OUTPUT





**CONCLUSION**

Customers can use an online booking system to book cabs. Customers may use this online system to browse available cabs, view profiles, and book cabs. Online cab booking is a typical kind of transportation that is offered by a number of different transportation firms in a particular city. The bulk of people rely on cab services for their daily transportation needs. The company must be © 2019 JETIR February 2019, Volume 6, Issue 2 www.jetir.org (ISSN-2349-5162) JETIRFE06047 Journal of Emerging Technologies and Innovative Research (JETIR) www.jetir.org 255 registered and fulfil all of the transportation department's requirements and security requirements. This paper demonstrates an effective cab booking system. This project included a wide variety of topics, from corporate principles to computer science, and required the completion of a number of courses in order to reach the deadlines.

**BIBLIOGRAPHY AND REFERENCES**

1. B. Oluwafemi, Uber Lagos Is Slashing Uber X Pricing By Up To 25 Percent [Online], Available from: http://techcabal.com/2015/05/01/uberlagos-is-slashing-uber-x-pricingby-up-to-25-percent/ Date accessed: 25th August, 2015.

2. Techloy, Android Is Officially The Most Widely Used Mobile OS In Nigeria [STATS] [Online], Available from: http://techloy.com/2014/09/27/android-is-officially-the-mostwidely used-mobile-os-in-nigeria-stats/ Date accessed: 12th October, 2015

3. G. Porter, Transport planning in sub-Saharan Africa, Progress in development studies., vol. 7, pp. 251-257, 2007.

4. J. Kujenya, How technology boosts enterprise [Online], The Nation, Available from: http://thenationonlineng.net/how-technology-boostsenterprise/ Date accessed: 25th August, 2015.

5. G. Porter, Mobility constraints and their implications for rural women and girls in subSaharan Africa, Gender, place and culture, vol. 18, pp. 65-81, 2011.

6. G. Porter, K. Blaufuss, and F. Owusu Acheampong, Youth, mobility and rural livelihoods in sub-Saharan Africa: perspectives from Ghana and Nigeria. Africa insight., vol. 37, pp. 420-431, 2007.

7. J. C. Aker, and I. M. Mbiti, Mobile phones and economic development in Africa, Center for Global Development Working Paper, (211), 2010.

8. G. Porter, Mobile phones, livelihoods and the poor in Sub‐Saharan Africa: Review and prospect.Geography Compass, vol. 6, pp. 241-259, 2012.

9. R. Sietchiping, M. J. Permezel, and C. Ngomsi, Transport and mobility in sub-Saharan African cities: An overview of practices, lessons and options for improvements. Cities, vol. 29, pp. 183-189, 2012