UNIVERSITI TEKNOLOGI MARA

UNIVERSITI TEKNOLOGI

MARA, KEDAH BRANCH

SCHOOL OF INFORMATIC SCIENCE

COLLEGE OF COMPUTING, INFORMATION SECURITY AND MATHEMATHICS

DIPLOMA IN LIBRARY INFORMATICS

(CDIM144) IML208 : PROGRAMMING FOR LIBRARIES

INDIVIDUAL PROJECT : CAB BOOKING SYSTEM

PREPARED BY :

NURUL ALIEYA NATASA BINTI ALIAS SANI

(200487572)

PREPARED FOR :

SIR MOHD FIRDAUS BIN MOHD HELMI

SUBMISSION DATE :

PROJECT NAME: cab booking sytem

FILE NAME:booking.py

PROMPT DATA:

1. Name
2. Address
3. Postcode
4. Telephone number
5. Email
6. Pick up and drop location

FUNCTION:

1. Creating data
2. Read data
3. Update data
4. Delete data

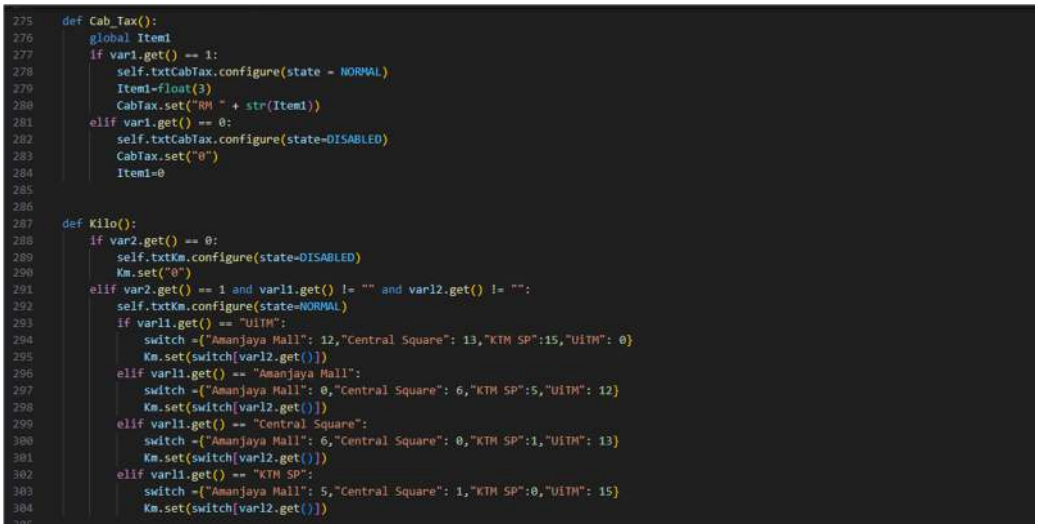
CONDITIONAL STATEMENT: yes

```
def Receiptt():
    if reset_counter == 0 and Firstname.get()!="" and Surname.get()!="" and Address.get()!="" and Postcode.get()!="" and Mobile.get()!="" and Te
        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")
```

Figure 1. screenshort coditional statement

```
C: > Users > User > Downloads > cabbooking.py > ...
            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, var11.get() + "\n")
            self.txtReceipt1.insert(END,'To:\n')
            self.txtReceipt2.insert(END, var12.get() + "\n")
            self.txtReceipt1.insert(END,'Pooling:\n')
            self.txtReceipt2.insert(END, var13.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")
```

Figure 2. screenshort conditional statement

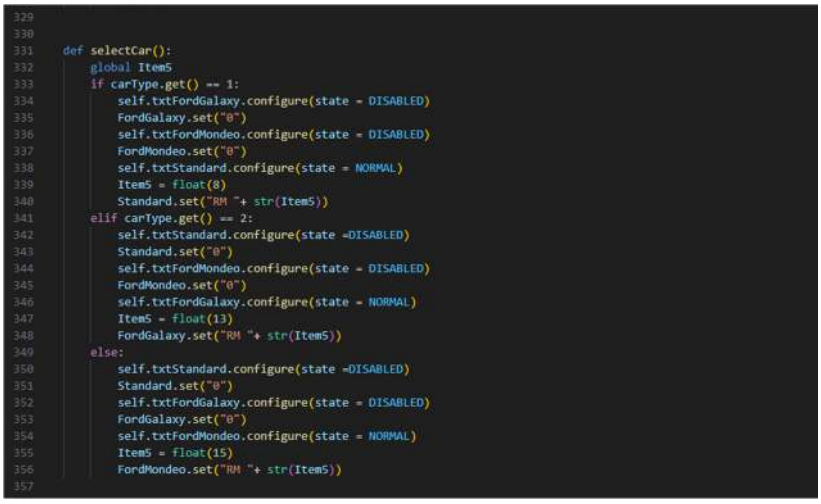


```
def Cab_Tax():
    global Item1
    if var1.get() == 1:
        self.txtCabTax.configure(state = NORMAL)
        Item1=float(3)
        CabTax.set("RM " + str(Item1))
    elif var1.get() == 0:
        self.txtCabTax.configure(state=DISABLED)
        CabTax.set("0")
        Item1=0


def Kilo():
    if var2.get() == 0:
        self.txtKm.configure(state=DISABLED)
        Km.set("0")
    elif var2.get() == 1 and var11.get() != "" and var12.get() != "":
        self.txtKm.configure(state=NORMAL)
        if var11.get() == "UiTM":
            switch ={"Amanjaya Mall": 12,"Central Square": 13,"KTM SP":15,"UiTM": 0}
            Km.set(switch[var12.get()])
        elif var11.get() == "Amanjaya Mall":
            switch ={"Amanjaya Mall": 0,"Central Square": 6,"KTM SP":5,"UiTM": 12}
            Km.set(switch[var12.get()])
        elif var11.get() == "Central Square":
            switch ={"Amanjaya Mall": 6,"Central Square": 0,"KTM SP":1,"UiTM": 13}
            Km.set(switch[var12.get()])
        elif var11.get() == "KTM SP":
            switch ={"Amanjaya Mall": 5,"Central Square": 1,"KTM SP":0,"UiTM": 15}
            Km.set(switch[var12.get()])
```

Figure 3. screenshort conditional statement

```
306
307  def Travelling():
308      global Item3
309      if var3.get() == 1:
310          self.txtTravel_Ins.configure(state = NORMAL)
311          Item3=float(2)
312          Travel_Ins.set("RM " + str(Item3))
313      elif var3.get() == 0:
314          self.txtTravel_Ins.configure(state = DISABLED)
315          Travel_Ins.set("0")
316          Item3=0
317
318
319  def Lug():
320      global Item4
321      if (var4.get()==1):
322          self.txtLuggage.configure(state = NORMAL)
323          Item4=float(10)
324          Luggage.set("RM "+ str(Item4))
325      elif var4.get()== 0:
326          self.txtLuggage.configure(state = DISABLED)
327          Luggage.set("0")
328          Item4=0
329
330
```

Figure 4. screenshort conditional statement



```
329
330
331  def selectCar():
332      global Item5
333      if carType.get() == 1:
334          self.txtFordGalaxy.configure(state = DISABLED)
335          FordGalaxy.set("0")
336          self.txtFordMondeo.configure(state = DISABLED)
337          FordMondeo.set("0")
338          self.txtStandard.configure(state = NORMAL)
339          Item5 = float(8)
340          Standard.set("RM "+ str(Item5))
341      elif carType.get() == 2:
342          self.txtStandard.configure(state =DISABLED)
343          Standard.set("0")
344          self.txtFordMondeo.configure(state = DISABLED)
345          FordMondeo.set("0")
346          self.txtFordGalaxy.configure(state = NORMAL)
347          Item5 = float(13)
348          FordGalaxy.set("RM "+ str(Item5))
349      else:
350          self.txtStandard.configure(state =DISABLED)
351          Standard.set("0")
352          self.txtFordGalaxy.configure(state = DISABLED)
353          FordGalaxy.set("0")
354          self.txtFordMondeo.configure(state = NORMAL)
355          Item5 = float(15)
356          FordMondeo.set("RM "+ str(Item5))
357
```

Figure 5. screenshort conditional statement

```
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 (var1
        if journeyType.get()==1:
            Item2=Km.get()
            Cost_of_fare = (Item1+(float(Item2)*Item5)+Item3+Item4)

            Tax = "RM " + str('%.1f'%((Cost_of_fare) *0.03))
            ST = "RM " + str('%.1f'%((Cost_of_fare)))
            TT = "RM " + str('%.1f'%(Cost_of_fare+((Cost_of_fare)*0.2)))
        elif journeyType.get()==2:
            Item2=Km.get()
            Cost_of_fare = (Item1+(float(Item2)*Item5)*1.1+Item3+Item4)

            Tax = "RM " + str('%.1f'%((Cost_of_fare) *0.06))
            ST = "RM " + str('%.1f'%((Cost_of_fare)))
            TT = "RM " + str('%.1f'%(Cost_of_fare+((Cost_of_fare)*0.2)))
        else:
            Item2=Km.get()
            Cost_of_fare = (Item1+(float(Item2)*Item5)*2+Item3+Item4)

            Tax = "RM " + str('%.1f'%((Cost_of_fare) *0.09))
            ST = "RM " + str('%.1f'%((Cost_of_fare)))
            TT = "RM " + str('%.1f'%(Cost_of_fare+((Cost_of_fare)*0.2)))

        PaidTax.set(Tax)
        SubTotal.set(ST)
        TotalCost.set(TT)
    else:
        w = ms.showwarning("Error !","Invalid Input\nPlease try again !!!")
```

Figure 6. screenshort conditional statement

GUI : YES

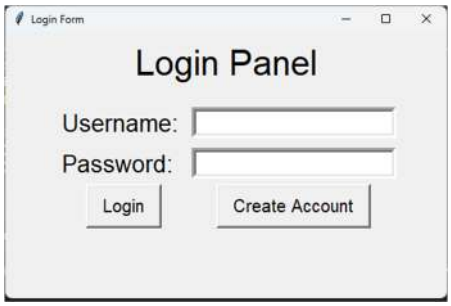


Figure 7. screenshort of GUI

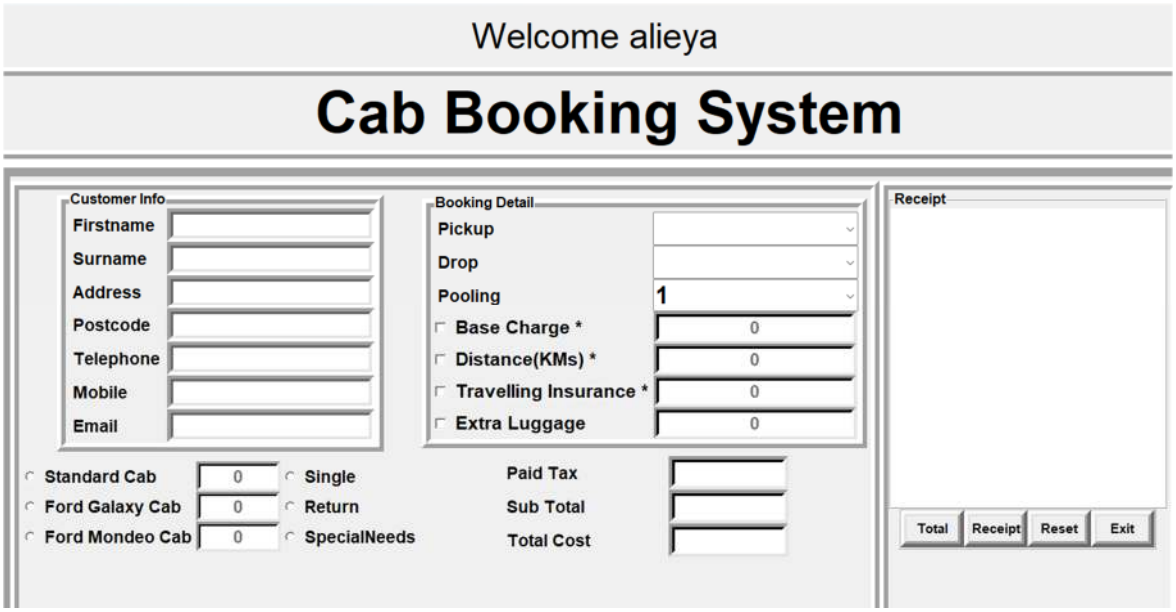


Figure 8. screenshort of GUI

RESULTS:

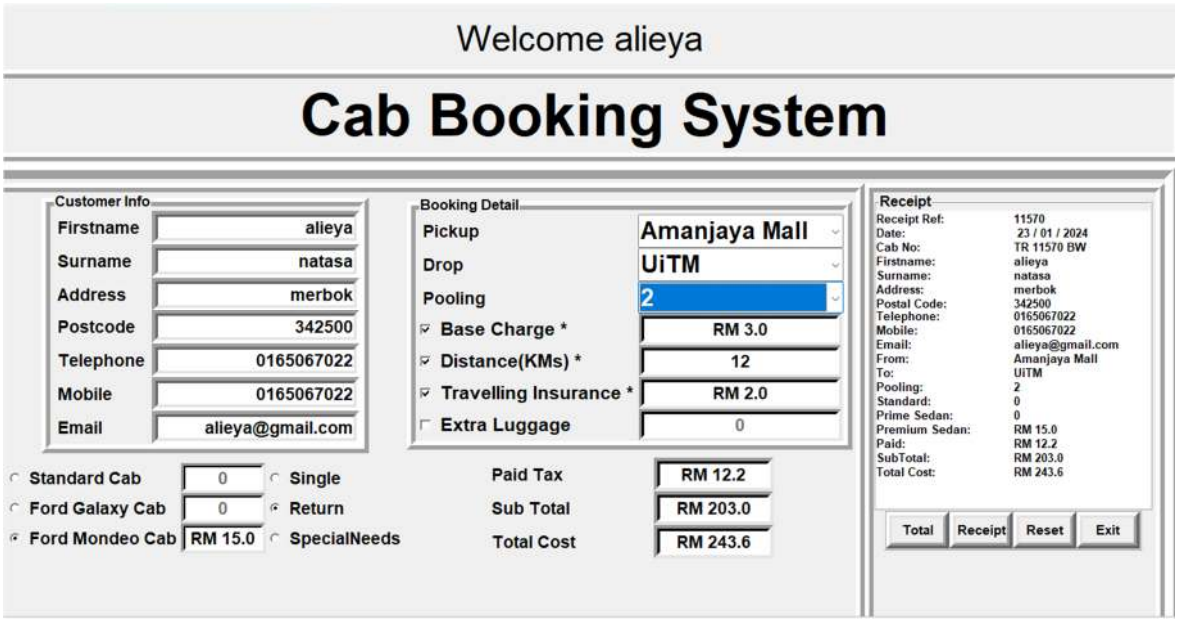


Figure 9. screenshort of GUI result

STRENGHT:

1. Convenience and Accessibility:
    - Enables users to book a cab conveniently using a mobile app or website.
    - 24/7 availability allows users to schedule rides at any time.
2. Real-Time Tracking:
    - Provides real-time tracking of the cab's location, allowing users to know the exact arrival time.
    - Enhances safety and security for passengers.
3. User-Friendly Interface:
    - Easy-to-use interface for both passengers and drivers.
    - Seamless booking process with minimal steps.
4. Multiple Payment Options:
    - Supports various payment methods, including credit/debit cards, digital wallets, and cash.
    - Enhances flexibility for users.
5. Integration with Maps:
    - Integration with mapping services for efficient route planning.
    - Reduces the chances of getting lost and optimizes travel time.

6. Automated Fare Calculation:
    - Transparent fare calculation based on distance, time, and other relevant factors.
    - Reduces disputes and provides clarity for users.
7. Feedback and Rating System:
    - Allows passengers to provide feedback and rate drivers.
    - Improves the overall quality of service and helps in maintaining high standards.

KAIZEN (ROOM FOR IMPROVEMENT)

1. Notification System:
    - Sends timely notifications to users regarding booking confirmation, driver details, and arrival updates.
    - Improves communication and keeps users informed.
2. Driver Management:
    - Efficient management of driver profiles, including background checks and verification.
    - Ensures the reliability and safety of drivers.
3. Scalability:
    - Ability to handle a large number of users and transactions simultaneously.
    - Can adapt to the growing demands of the user base.
4. Promotions and Discounts:
    - Integration of promotional offers, discounts, and loyalty programs.
    - Attracts more users and retains existing ones.
5. Analytics and Reporting:
    - Provides data analytics and reporting tools for operators to analyze performance.
    - Enables data-driven decision-making and optimization.
6. Security and Privacy:
    - Implements robust security measures to protect user data and payment information.
    - Ensures the privacy and safety of both passengers and drivers.