AMRITSAR GROUP OF COLLEGES

UGC Autonomous College | Accredited by NAAC | Accredited by NBA

Project Report

On

"RAILWAY RESERVATION SYSTEM"

Submitted in the Partial fulfilment of the requirement for the Award of Degree of

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING

(2020-2024)

Submitted to:

Er. Ajay Sharma (Associate Professor)

Er. Neha (Assistant Prof.)

Er. Anamika Jain (Assistant Prof.)

Submitted by:

Rohit Kr (1900240)

Arif Anwer (2000070)

Anshu kumar (2000069)

Arshnoor Singh (2000072)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Amritsar Group of Colleges, Amritsar

ACKNOWLEDGEMENT

It is our proud privilege to release the feelings of our gratitude to several persons who helped us directly or indirectly to conduct this Analysis Project Work. We express our heart full thanks and owe a deep sense of gratitude to our teacher and my faculty guide ***Er. Ajay Sharma (Associate Professor, Department of CSE, Amritsar Group of Colleges, Amritsar)*** and ***Er. Neha Chadha (Assistant Professor, Department of CSE, Amritsar Group of Colleges, Amritsar)***, for his guidance and inspiration in completing this project.

We are extremely thankful to ***Dr. Sandeep Kad (Head of Department, Department of Computer Science & Engineering, Amritsar Group of Colleges)*** and all faculty members of CSE Department at Amritsar Group of Colleges, Amritsar for their co-ordination and co-operation and for their kind guidance and encouragement.

We also thank all our friends who have more or less contributed to the preparation of this project Report, we will be always indebted to them. This project completion has indeed helped us explore more knowledge avenues related to RDBMS/Python and we are sure it will help us in future too.

DECLARATION

We, hereby, declare that the project work entitled ***RAILWAY RESERVATION SYSTEM*** submitted to **Department of Computer Science & Engineering, AGC – Amritsar Group of Colleges**, is a record of an original work done by us under the guidance of **Er. Ajay Sharma**, **Er. Neha Chadha, Er. Anamika Jain** and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Rohit Kr
Univ. Roll No: 1900240

Arif Anwer
Univ Roll No: 2000070

Anshu Kumar
Univ Roll No: 2000069

Arshnoor Singh Sohi
Univ Roll No: 2000072

TABLE OF CONTENTS

INTRODUCTION TO RDBMS

Relational Data Model has an advantage that it is simple to implement and easy to understand as it uses table format. In this approach, a relation is only constructed by setting the association among the attributes of an entity as well the relationship among different entities. One of the main reasons for introducing this model was to increase the productivity of the application programmers by eliminating the need to change application program, when a change is made to the database. Data structure used in the data model is represented by both entities and relationship between them. Information is represented in the relational model in shape of tables. There are columns in a table which represent the attributes of an entity about which the table is constructed. The rows of a table are referred to as tuples.

1.1. STRUCTURED QUERY LANGUAGE

SQL tables, which consists of rows and columns, are used for storing data. The columns refer to the attributes. Each column in a table has a column name and a data type. A value's data type associates a fixed set of properties with a value. The data types available in Oracle fall under the following categories.

Category	**Available data types**
Character	CHAR, VARCHAR, VARCHAR, NCHAR, NVARCHAR2, LONG, RAW, LONGRAW
Number	Number
Date/Time	Date
LOB‟s	BFILE, BLOB CLOB, NCLOB

WHAT IS A TABLE ?

A table is a database object which is used to store data in relational databases. Each table consists of rows and columns. A column in the database table represents the table's attributes and a row represents a single set of column values in a database table. Each column of the table has a column name and a data type associated with it. The data types which can be used include VARCHAR2, NUMBER, DATE etc. A row of a table is also known as record. Within a table foreign keys are used to represent relationships.

CREATING A TABLE

In order to store and manage data it is necessary to create tables. In Oracle, tables are created using CREATE TABLE command which is one of the important DDL statement. The CREATE TABLE command specifies the name of the table, name of columns in the table as well as the data types and if required constraints associated with each column of the table.

INTEGRITY CONSTRAINTS IN CREATE TABLE

Oracle uses integrity constraints to prevent invalid data entry into the base tables of the database. One can define integrity constraints to enforce the business rules that one wants to associate with the information in the database. If the integrity constraints are violated due to the execution of any of DML statements (Insert, Update, Select) then Oracle rolls back the statement and returns an error. The different kinds of constraints are:

- Primary Key Constraint
- Unique Key Integrity Constraint
- NOT NULL Integrity Constraint
- Foreign Key Constraint
- Check Integrity Constraint

INTRODUCTION TO PYTHON

Python is a general purpose programming language that is often applied in scripting roles.

It is also called as Interpreted language.

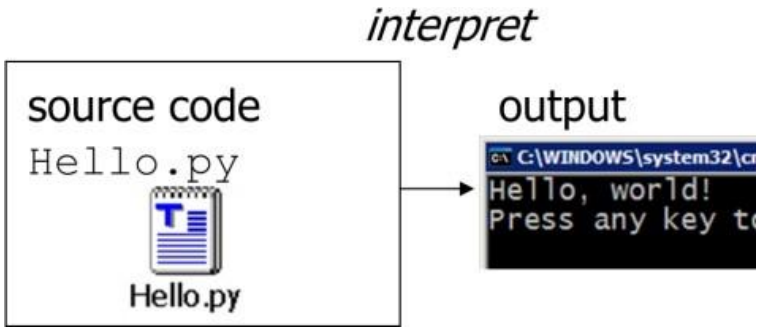
The Origin of Python:

It is invented in Netherlands, in early 90's and its implementation was started in December 1989. Guido Van Rossum is a fan of 'Monty Python's Flying Circus', famous TV show in Netherlands. So he named the language after Monty Python.

Features Of Python:

- It's Free: Downloading and installing Python is free and easy. Source code is easily accessible.
- Its Portable: Python runs virtually on every major platform used today. Programs runs exactly in the same manner irrespective of platform.
- It's Powerful:: Dynamic typing , Built-in types and tools , library utilities , third-party utilities(e.g. Numpy , Scipy)
- It's Mixable: Integration of python with other languages is widely used .
- It's Relatively Easy to Use: Python Programs are compiled automatically to an intermediate form called bytecode , which the interpreter then reads.
- It's Relatively Easy to Learn
- Its Object – Oriented and Functional

Python is instead directly interpreted into machine instructions.



INTRODUCTION OF THE PROJECT

This project aims at building an **Railway Reservation System**. It is divided into different modules to make it more user-friendly. The front –end is designed using Python3 – Tkinter Tool.

This project has been developed to override the problems of prevailing in the practicing manual system. This application is supported to eliminate and reduce the hardships faced by the existing system. Moreover this is designed for the particular need of the institute to carry out operations in a smooth and effective manner.

The purpose of the Railway Reservation System is to automate the existing manual system by the help of the computerized equipment and full-fledged computer software, fulfilling their requirements so that the valuable data/information can be stored for a long period with easy accessing and manipulation of the same.

The main objective of the Project on Railway Reseravation System is to manage the details of Train Routes, Customer details and Manage trains. It can manage all the information about Train Routes details along with their required information.

The project is built with administrative authorization where multiple users with different roles can access and manipulate the data/information according to their roles.

Functionalities provided by Railway Reservation System:

- Provides the searching facilities based on various factors. Such as Train, Booking, Customer, Payment
- Railway Reservation System also manage the Seat details online for Customer details, Payment details, Train.
- It tracks all the information of Ticket, Seat, Customer etc.
- Manage the information of Ticket
- Shows the information and description of the Train, Booking
- To increase efficiency of managing the Train, Ticket
- It deals with monitoring the information and transactions of Customer.
- Manage the information of Train
- Editing, adding and updating of Records is improved which results in proper resource management of Train data.
- Manage the information of Customer

OBJECTIVES OF THE PROJECT

- Authorized Log In can create his account and can reserve his or her ticket for train.
- Authorized Manager has full access of all the modules which again gives it full authorization to add, update or delete in train route details.
- If a user forgets his/her password he/she can reset the password any time by confirming the mobile number.
- If a user doesn't have an account he/she can simply register a new account.
- All the fields such as Train Booking , username and various entry fields are validated and does not take any invalid value.
- Each form for Train, Ticket, Seat can not accept blank value fields.
- It tracks all the information of Ticket, Seat, Customer etc.
- Manage the information of Ticket
- Shows the information and description of the Train, Booking
- To increase efficiency of managing the Train, Ticket
- It deals with monitoring the information and transactions of Customer.
- Manage the information of Train
- Editing, adding and updating of Records is improved which results in proper resource management of Train data.
- Manage the information of Customer
- Validations for all user inputs

TOOLS USED

Front-end: **Python Tkinter**

Back-end: **Oracle Database**

3.1 Introduction to Tkinter

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. All you need to do is perform the following steps −

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

```
import tkinter
```

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Tkinter is a good choice because of the following reasons:

- Easy to learn.
- Use very little code to make a functional desktop application.
- Layered design.
- Portable across all operating systems including Windows, macOS, and Linux.
- Pre-installed with the standard Python library.

3.2. Introduction to Oracle Database

Oracle database (Oracle DB) is a relational database management system (RDBMS) from the Oracle Corporation. Originally developed in 1977 by Lawrence Ellison and other developers, Oracle DB is one of the most trusted and widely-used relational database engines.

The system is built around a relational database framework in which data objects may be directly accessed by users (or an application front end) through structured query language (SQL). Oracle is fully scalable relational database architecture and is often used by global enterprises, which manage and process data across wide and local area networks. The Oracle database has its own network component to allow communications across networks.

A key feature of Oracle is that its architecture is split between the logical and the physical. This structure means that for large-scale distributed computing, also known as grid computing, the data location is irrelevant and transparent to the user, allowing for a more modular physical structure that can be added to and altered without affecting the activity of the database, its data or users. The sharing of resources in this way allows for very flexible data networks whose capacity can be adjusted up or down to suit demand, without degradation of service. It also allows for a robust system to be devised as there is no single point at which a failure can bring down the database, as the networked schema of the storage resources means that any failure would be local only.

MODULES USED

Following are the modules developed in the project:

- **Login Module**

  Login module is used when user or manager wants to login in the database. For manager a different window will open in which he can enter train route related data and for user a booking page will open in which the user can book a ticket for train. Another facility provided by login page is that if user doesn't have an account he can click to register button to register an account and use its features. One for facility is there if a user forgets his/her password he can reset the password with mobile configuration.

- **Manager Module**

  In Manager Module, Manager can insert all the details of the train. Details like From which station the train is going to start, To which station it is going along with train's number, the station code, the arrival time of the train and departure time of the train. The Manager can ADD new data, DELETE existing data, UPDATE existing with one more option of CLEAR data. The manager can search any train by Train number or station code. Tree-view structure is provided for this use.

- **Registration Module**

  The user can register a new account with the help of Registration Module. In Registration Module user have to enter his/her first name, last name, contact number, username, gender, age and password. At last he have to agree all terms and conditions to finally register an account.

- **Booking Module**

  The Booking Module Is designed for the user to book ticket for the train. In this user is provided with the search bar in which he/she will be provided two option From where he want to go and To what destination he want to reach. He can select any of the option to search train with providing name of the place . And then enter search button if he finds its train he/she can just click on it and train details will be placed in left view then he only have to click on the BOOK TICKET button and a message will be prompted providing ticket no. of that user and user will be successfully able to book his/her ticket.

SYSTEM ANALYSIS AND DATA TABLES

- **REGISTRATION TABLE**

```
SQL> desc registration;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                          VARCHAR2(20)
 CONTACT_NUMBER                                     VARCHAR2(20)
 EMAIL                                     NOT NULL VARCHAR2(20)
 GENDER                                             VARCHAR2(10)
 AGE                                                VARCHAR2(5)
 PASSWORD                                           VARCHAR2(20)
```

- **MANAGER TABLE**

```
SQL> desc manager;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FROM_                                              VARCHAR2(15)
 TO_                                                VARCHAR2(15)
 TRAIN_NO                                           VARCHAR2(15)
 STATION_CODE                                       VARCHAR2(15)
 ARRIVAL_TIME                                       VARCHAR2(15)
 DEPARTURE_TIME                                     VARCHAR2(15)
```

- **TICKET TABLE**

```
SQL> desc ticket;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TICKET_NO                                          VARCHAR2(10)
 FROM_                                              VARCHAR2(15)
 TO_                                                VARCHAR2(15)
 TRAIN_NO                                           VARCHAR2(15)
 STATION_CODE                                       VARCHAR2(15)
 ARRIVAL_TIME                                       VARCHAR2(15)
 DEPARTURE_TIME                                     VARCHAR2(15)
```

SOURCE CODE

LOGIN MODULE

```
from tkinter import *
from PIL import Image,ImageTk,ImageDraw
from tkinter import ttk,messagebox
from datetime import *
import time
from math import *
import cx_Oracle
class Login:
    def __init__(self, root):
        self.root = root
        self.root.title("Railway Login Page")
        self.root.geometry("1350x700+0+0")
        self.root.config(bg="#021e2f")

        #BACKGROUND COLORS----------------------
        left_label = Label(self.root, bg="#08A3D2", bd=0)
        left_label.place(x=0, y=0, relheight=1, width=600)

        right_label = Label(self.root, bg="#031F3C", bd=0)
        right_label.place(x=600, y=0, relheight=1, relwidth=1)

        #FRAMES---------------------------------
        login_frame = Frame(self.root, bg="white")
        login_frame.place(x=250,y=100,width=800,height=500)

        title=Label(login_frame,text="LOGIN HERE",font=("times new
roman",30,"bold"),bg="white",fg="#08A3D2").place(x=250,y=50)

        email=Label(login_frame,text="USERNAME",font=("times new
roman",18,"bold"),bg="white",fg="grey").place(x=250,y=150)
        self.txt_username=Entry(login_frame, font=("times new roman", 15, "bold"), bg="lightgrey")
        self.txt_username.place(x=250, y=180, width=350, height=35)

        pass_ = Label(login_frame, text="PASSWORD", font=("times new roman", 18, "bold"),
bg="white",fg="grey").place(x=250, y=250)
        self.txt_pass_ = Entry(login_frame, font=("times new roman", 15, "bold"), bg="lightgrey")
        self.txt_pass_.place(x=250, y=280, width=350, height=35)


        self.btn_reg = Button(login_frame,cursor="hand2",command=self.register_window,text="Register New
Account?",font=("times new roman",14),bg="white",bd=0,fg="#B00857").place(x=250,y=320)
        self.btn_forget =
Button(login_frame,cursor="hand2",command=self.forget_password_window,text="Forget
Password?",font=("times new roman",14),bg="white",bd=0,fg="red").place(x=450,y=320)

        self.btn_login = Button(login_frame,command=self.login, text="LOGIN", font=("times new roman", 20,
"bold"), fg="white", bg="#B00857",cursor="hand2").place(x=250, y=380, width=180, height=40)
```

```
        #CLOCK---------------------------------
        self.lbl = Label(self.root, text="CLOCK", font=("Book Antiqua", 25, "bold"), fg="white",
compound=BOTTOM, bg="#081923", bd=0)
        self.lbl.place(x=90, y=120, height=450, width=350)

        self.working()

    def reset(self):
        self.txt_new_password.delete(0, END)
        self.txt_contact.delete(0, END)
        self.txt_username.delete(0, END)
        self.txt_pass_.delete(0, END)

    def forget_password(self):
        if self.txt_contact.get() == "" or self.txt_new_password.get() == "":
            messagebox.showerror("Error", "All fields are required", parent=self.root2)
        else:
            try:
                con = cx_Oracle.connect("railway/railway007")
                cur = con.cursor()

                em = self.txt_username.get()
                cont = self.txt_contact.get()
                cur.execute("select * from registration where email = :email and contact_number = :contact", {'email':
em,'contact': cont})
                # To fetch email and password for user
                row = cur.fetchone()
                if row == None:
                    messagebox.showerror("Error", "Please enter the correct contact number.", parent=self.root2)
                else:
                    npass = self.txt_new_password.get()
                    cur.execute("update registration set password = :new_pass where email = :email ",{'new_pass':
npass,'email': em})
                    con.commit()
                    con.close()
                    messagebox.showinfo("Success","Your password has been reset,Please login with new
password",parent = self.root2)
                    self.reset()
                    self.root2.destroy()

            except Exception as es:
                messagebox.showerror("Error", f"Error Due to:{str(es)}", parent=self.root)


    def forget_password_window(self):
        if self.txt_username.get() == "":
            messagebox.showerror("Error","Please enter the username to rest your password.",parent=self.root)
        else:
            try:
                con = cx_Oracle.connect("railway/railway007")
                cur = con.cursor()

                em = self.txt_username.get()
                cur.execute("select * from registration where email = :email ",{'email':em })
                #To fetch email and password for user
                row = cur.fetchone()
```

```
            if row == None:
                messagebox.showerror("Error", "Please enter your valid username to rest your
password.",parent=self.root)
            else:
                con.close()
                self.root2 = Toplevel()
                self.root2.title("Forget Password")
                self.root2.geometry("400x350+440+150")
                self.root2.config(bg="white")
                self.root2.focus_force()  # focuses on the currently open window
                self.root2.grab_set()  # grab untile user closes it

                t = Label(self.root2, text="Forget Password", font=("times new roman", 20, "bold"),
bg="white",fg="red").place(x=0, y=20, relwidth=1)

                contact = Label(self.root2, text="CONTACT NO.", font=("times new roman", 15, "bold"),
bg="white",fg="#141466").place(x=50, y=100)
                self.txt_contact = Entry(self.root2, font=("times new roman", 15), bg="lightgrey")
                self.txt_contact.place(x=50, y=130, width=250)

                new_password = Label(self.root2, text="NEW PASSWORD", font=("times new roman", 15,
"bold"),bg="white", fg="#141466").place(x=50, y=170)
                self.txt_new_password = Entry(self.root2, font=("times new roman", 15), bg="lightgrey")
                self.txt_new_password.place(x=50, y=200, width=250)

                btn_change_password = Button(self.root2, text="Reset Password",command=self.forget_password,
bg="red", fg="white",font=("times new roman", 15, "bold")).place(x=120, y=270)


        except Exception as es:
            messagebox.showerror("Error", f"Error Due to:{str(es)}", parent=self.root)

    def register_window(self):
        self.root.destroy()
        import register

    def login(self):
        if self.txt_username.get() == "" or self.txt_pass_.get() == "":
            messagebox.showerror("Error","All fields are required",parent=self.root)
        else:
            try:
                con = cx_Oracle.connect("railway/railway007")
                cur = con.cursor()

                em = self.txt_username.get()
                ps = self.txt_pass_.get()
                cur.execute("select * from registration where email = :email and password = :password",{'email':em,
'password' :ps })
                #To fetch email and password for user
                row = cur.fetchone()
                if row == None:
                    messagebox.showerror("Error", "Invalid EMAIL or PASSWORD", parent=self.root)
                elif self.txt_username.get() == "Manager007@gmail.com" and self.txt_pass_.get() == "manager007":
                    messagebox.showinfo("Success", "Welcome", parent=self.root)
                    self.root.destroy()
                    import manager
```

```
            else:
                messagebox.showinfo("Success","Welcome",parent = self.root)
                self.root.destroy()
                import Booking
            con.close()
        except Exception as es:
            messagebox.showerror("Error", f"Error Due to:{str(es)}", parent=self.root)


    def clock_image(self, hr, min_, sec_):
        clock = Image.new("RGB", (400, 400), (8,25,35))
        draw = ImageDraw.Draw(clock)
        #For clock image
        bg = Image.open("images/darkclock.jpg")
        bg = bg.resize((300, 300), Image.ANTIALIAS)
        clock.paste(bg, (50, 50))
        #Formula To Rotate the Anti-Clock
        #angle_in_radians = angle_in_degrees * math.pi /180
        #line_lenght = 100
        #center_x = 250
        #center_y = 250
        #end_x = center_x + line_lenght * math.cos(angle_in_radians)
        #end_y = center_y + line_lenght * math.sin(angle_in_radians)

        #For Hour line image
        origin = 200, 200
        draw.line((origin, 200+50*sin(radians(hr)), 200-50*cos(radians(hr))), fill="#DF005E", width=4)
        #For Minute line image
        draw.line((origin, 200+80*sin(radians(min_)), 200-80*cos(radians(min_))), fill="white", width=3)
        #For Second line image
        draw.line((origin, 200+100*sin(radians(sec_)), 200-100*cos(radians(sec_))), fill="yellow", width=2)
        draw.ellipse((195, 195, 210, 210), fill="black")
        clock.save("images/clock_new.png")

    def working(self):
        h = datetime.now().time().hour
        m = datetime.now().time().minute
        s = datetime.now().time().second

        hr = (h/12)*360
        min_ = (m/60)*360
        sec_ = (s/60)*360

        self.clock_image(hr, min_, sec_)
        self.img = ImageTk.PhotoImage(file="images/clock_new.png")
        self.lbl.config(image=self.img)
        self.lbl.after(200,self.working)

root = Tk()
obj = Login(root)
root.mainloop()
```

REGISTER MODULE

```
from tkinter import *
from tkinter import ttk,messagebox
from PIL import Image,ImageTk
import cx_Oracle
import re
class Register:
    def __init__(self,root):
        self.root=root
        self.root.title("REGISTRATION TAB")
        self.root.geometry("1350x700+0+0")
        self.root.config(bg="white")
        # #To remove max tab issue
        # self.root.resizable(False,False)
        # BG IMAGE
        self.bg=ImageTk.PhotoImage(file="images/bg_pic1.png")
        bg=Label(self.root, image=self.bg).place(x=0, y=0, relwidth=1, relheight=1) #250


        #left imgae
        self.left=ImageTk.PhotoImage(file="images/railway_clipart1.png")
        left=Label(self.root, image=self.left).place(x=80, y=100, width=400, height=500)

        #Register Frame
        frame1=Frame(self.root,bg="white")
        frame1.place(x=480,y=100,width=700,height=500)

        title=Label(frame1,text="REGISTER HERE",font=("times new
roman",25,"bold"),bg="white",fg="red").place(x=50,y=30)

        #-------------------------------Row1-------------------------------------------

        f_name=Label(frame1, text="FIRST NAME", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=50, y=100)
        self.txt_fname=Entry(frame1,font=("times new roman",15),bg="lightgrey")
        self.txt_fname.place(x=50, y=130, width=250)

        l_name=Label(frame1, text="LAST NAME", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=370, y=100)
        self.txt_lname=Entry(frame1,font=("times new roman",15),bg="lightgrey")
        self.txt_lname.place(x=370, y=130, width=250)

        #-------------------------------Row2-------------------------------------------

        contact=Label(frame1, text="CONTACT NO.", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=50, y=170)
        self.txt_contact=Entry(frame1,font=("times new roman",15),bg="lightgrey")
        self.txt_contact.place(x=50,y=200,width=250)

        username=Label(frame1, text="USERNAME", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=370, y=170)
        self.txt_username=Entry(frame1, font=("times new roman", 15), bg="lightgrey")
```

```
        self.txt_username.place(x=370, y=200, width=250)

        #---------------------------------Row3-----------------------------------------------

        question=Label(frame1, text="GENDER", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=50, y=240)

        self.cmb_quest=ttk.Combobox(frame1,font=("times new roman",13),state='readonly',justify=CENTER)
        self.cmb_quest['values']=("Select","MALE","FEMALE","OTHER")
        self.cmb_quest.place(x=50, y=270, width=250)
        self.cmb_quest.current(0)

        age=Label(frame1, text="AGE", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=370, y=240)
        self.txt_age=Entry(frame1,font=("times new roman",15),bg="lightgrey")
        self.txt_age.place(x=370,y=270,width=250)

        #---------------------------------Row4-----------------------------------------------

        password=Label(frame1, text="PASSWORD.", font=("times new roman", 15, "bold"), bg="white",
fg="#141466").place(x=50, y=310)
        self.txt_password=Entry(frame1,font=("times new roman",15),bg="lightgrey")
        self.txt_password.place(x=50,y=340,width=250)

        cpassword=Label(frame1, text="CONFIRM PASSWORD", font=("times new roman", 15, "bold"),
bg="white", fg="#141466").place(x=370, y=310)
        self.txt_cpassword=Entry(frame1,font=("times new roman",15),bg="lightgrey")
        self.txt_cpassword.place(x=370,y=340,width=250)

        #-------------Terms and conditions-----------
        self.var_chk=IntVar()
        chk=Checkbutton(frame1,text="I AGREE THE TERMS AND
CONDITIONS",variable=self.var_chk,onvalue=1,offvalue=0,bg="white",font=("timws new
eoman",12)).place(x=50,y=380)

        self.btn_img=ImageTk.PhotoImage(file="images/register1.jpg")

btn_register=Button(frame1,image=self.btn_img,bd=0,cursor="hand2",command=self.register_data).place(x=50,
y=420)

        btn_signin=Button(self.root,text="SIGN IN",command=self.login_window,font=("times new
roman",20),bd=0,cursor="hand2").place(x=200,y=460,width=180)

    def login_window(self):
        self.root.destroy()
        import login

    def clear(self):
        self.txt_fname.delete(0,END)
        self.txt_lname.delete(0,END)
        self.txt_contact.delete(0,END)
        self.txt_username.delete(0, END)
        self.txt_age.delete(0,END)
        self.txt_password.delete(0,END)
        self.txt_cpassword.delete(0,END)
        self.cmb_quest.current(0)
```

```
    def register_data(self):

        if self.txt_fname.get() == "" or self.txt_contact.get() == "" or self.txt_username.get() == "" or
self.cmb_quest.get() == "Select" or self.txt_age.get() == "" or self.txt_password.get() == "" or
self.txt_cpassword.get() == "":
            messagebox.showerror("Error","All Fields Are Required",parent=self.root)

        elif self.txt_password.get() != self.txt_cpassword.get() :
            messagebox.showerror("Error","Password & Confirm Password should be same.",parent=self.root)
        elif self.var_chk.get() == 0:
            messagebox.showerror("Error","Please agree our terms & conditions",parent=self.root)
        else:
            try:
                con = cx_Oracle.connect("railway/railway007")
                cur = con.cursor()
                #To fetch email if already exists
                em = self.txt_username.get()
                stat = "select * from registration where email = :email"
                cur.execute(stat ,{'email':em})
                row = cur.fetchone()
                print(row)
                if row != None:
                    messagebox.showerror("Error", "User already exist,Please try with another email", parent=self.root)
                else:

                    cur.execute("insert into registration
(FIRST_NAME,LAST_NAME,CONTACT_NUMBER,EMAIL,GENDER,AGE,PASSWORD) values
(:1,:2,:3,:4,:5,:6,:7)",
                                (self.txt_fname.get(),
                                 self.txt_lname.get(),
                                 self.txt_contact.get(),
                                 self.txt_username.get(),
                                 self.cmb_quest.get(),
                                 self.txt_age.get(),
                                 self.txt_password.get()
                                ))
                    con.commit()
                    con.close()
                    messagebox.showinfo("SUCCESS", "Registered Successfull", parent=self.root)
                    self.clear()
            except Exception as es:
                messagebox.showerror("Error", f"Error due to: {str(es)}", parent=self.root)

root  = Tk()
obj=Register(root)
root.mainloop()
```

MANAGER MODULE

```
from tkinter import *
from tkinter import ttk, messagebox
import cx_Oracle
class Manager:
    def __init__(self):
        self.root = root
        self.root.title("RAILWAY MANAGEMENT SYSTEM")
        self.root.geometry("1350x700+0+0")
        self.root.config(bg="#021e2f")

        title = Label(self.root, text="RAILWAY MANAGEMENT SYSTEM", bd=10, relief=GROOVE,
font=("times new roman",40,"bold"), bg="#021e2f", fg="red")#fg=textcolor
        title.pack(side=TOP, fill=X)

        #------------ALL VARIABLES-------------
        self.From_var = StringVar()
        self.To_var = StringVar()
        self.TrainNo_var = StringVar()
        self.St_Code_var = StringVar()
        self.A_Time_var = StringVar()
        self.D_Time_var = StringVar()

        self.search_by = StringVar()
        self.search_txt = StringVar()


        #------------MANAGER FRAME-------------
        Manage_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="crimson")
        Manage_Frame.place(x=20,y=100,width=475,height=580)

        m_title = Label(Manage_Frame,text="Manage Train Routes",bg="crimson",fg="white",font=("times new
roman",30,"bold"))
        m_title.grid(row=0,columnspan=2,pady=20)
        #------From
        lbl_from = Label(Manage_Frame, text="From", bg="crimson", fg="white",font=("times new roman", 20,
"bold"))
        lbl_from.grid(row=1, column=0, pady=10, padx=20, sticky="W")

        txt_from = Entry(Manage_Frame,textvariable=self.From_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_from.grid(row=1, column=1, pady=10, sticky="W")
        #-------To
        lbl_to = Label(Manage_Frame, text="To", bg="crimson", fg="white",font=("times new roman", 20, "bold"))
        lbl_to.grid(row=2, column=0, pady=10, padx=20, sticky="W")

        txt_to = Entry(Manage_Frame,textvariable=self.To_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_to.grid(row=2, column=1, pady=10, sticky="W")
        #-------TrainNo
        lbl_train_no = Label(Manage_Frame, text="TrainNo.  ", bg="crimson", fg="white",font=("times new
roman", 20, "bold"))
        lbl_train_no.grid(row=3, column=0, pady=10, padx=20, sticky="W")
```

```
        txt_train_no = Entry(Manage_Frame,textvariable=self.TrainNo_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_train_no.grid(row=3, column=1, pady=10, sticky="W")
        #-------St-Code
        lbl_station_code = Label(Manage_Frame, text="St-Code", bg="crimson", fg="white",font=("times new
roman", 20, "bold"))
        lbl_station_code.grid(row=4, column=0, pady=10, padx=20, sticky="W")

        txt_station_code = Entry(Manage_Frame,textvariable=self.St_Code_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_station_code.grid(row=4, column=1, pady=10, sticky="W")
        #--------A-Time
        lbl_arrivaltime = Label(Manage_Frame, text="A-Time", bg="crimson", fg="white",font=("times new
roman", 20, "bold"))
        lbl_arrivaltime.grid(row=5, column=0, pady=10, padx=20, sticky="W")

        txt_arrivaltime = Entry(Manage_Frame,textvariable=self.A_Time_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_arrivaltime.grid(row=5, column=1, pady=10, sticky="W")
        #--------D-Time
        lbl_departuretime = Label(Manage_Frame, text="D-Time", bg="crimson", fg="white",font=("times new
roman", 20, "bold"))
        lbl_departuretime.grid(row=6, column=0, pady=10, padx=20, sticky="W")

        txt_departuretime = Entry(Manage_Frame,textvariable=self.D_Time_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_departuretime.grid(row=6, column=1, pady=10, sticky="W")

        #-------BUTTON FRAME----------------
        btn_Frame = Frame(Manage_Frame,bd=4,relief=RIDGE,bg="crimson")
        btn_Frame.place(x=30,y=500,width=410)

        Addbtn =
Button(btn_Frame,text="Add",width=10,command=self.add_train_info).grid(row=0,column=0,padx=10,pady=1
0)
        updatebtn =
Button(btn_Frame,text="Update",width=10,command=self.update_data).grid(row=0,column=1,padx=10,pady=1
0)
        deletebtn =
Button(btn_Frame,text="Delete",width=10,command=self.delete_data).grid(row=0,column=2,padx=10,pady=10)
        clearbtn =
Button(btn_Frame,text="Clear",width=10,command=self.clear).grid(row=0,column=3,padx=10,pady=10)


        # ------------DETAIL FRAME-------------
        Detail_Frame = Frame(self.root, bd=4, relief=RIDGE, bg="crimson")
        Detail_Frame.place(x=500, y=100, width=770, height=580)

        lbl_search = Label(Detail_Frame, text="Search By", bg="crimson", fg="white", font=("times new roman",
20, "bold"))
        lbl_search.grid(row=0, column=0, pady=10, padx=20, sticky="W")

        combo_search = ttk.Combobox(Detail_Frame,textvariable=self.search_by,width=15,font=("times new
roman",13,"bold"),state="readonly")
        combo_search['values'] = ('Train_No' , 'Station_Code')
```

```
        combo_search.grid(row=0,column=1,padx=20,pady=10)

        txt_search = Entry(Detail_Frame,textvariable=self.search_txt, font=("times new roman", 10, "bold"), bd=5,
relief=GROOVE)
        txt_search.grid(row=0, column=2, pady=10 ,padx=20 , sticky="w")

        searchbtn = Button(Detail_Frame, text="Search",
width=10,pady=5,command=self.search_data).grid(row=0, column=3, padx=10, pady=10)
        showallbtn = Button(Detail_Frame, text="Show All",
width=10,pady=5,command=self.fetch_data_to_searchby).grid(row=0, column=4, padx=10, pady=10)

        #---------Table Frame--------------
        Table_Frame = Frame(Detail_Frame, bd=4, relief=RIDGE, bg="crimson")
        Table_Frame.place(x=10, y=70, width=745, height=500)

        scroll_x = Scrollbar(Table_Frame,orient=HORIZONTAL)
        scroll_y = Scrollbar(Table_Frame,orient=VERTICAL)
        self.Train_table = ttk.Treeview(Table_Frame,columns=("From","To","TrainNo","St-Code","A-Time","D-
Time"),xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT,fill=Y)
        scroll_x.config(command=self.Train_table.xview)
        scroll_y.config(command=self.Train_table.yview)
        self.Train_table.heading("From",text="From")
        self.Train_table.heading("To",text="To")
        self.Train_table.heading("TrainNo",text="TrainNo")
        self.Train_table.heading("St-Code",text="St-Code")
        self.Train_table.heading("A-Time",text="A-Time")
        self.Train_table.heading("D-Time",text="D-Time")
        self.Train_table['show'] = 'headings'
        # Train_table.column("From",width=50) ---- IF YOU WANT TO ADJUST THE WIDTH
        self.Train_table.pack(fill=BOTH,expand=1)
        self.Train_table.bind("<ButtonRelease-1>", self.get_data_via_cursor) #event
        self.fetch_data_to_searchby()

    def add_train_info(self): #had to give two arguments
        if self.From_var.get() == "" or self.To_var.get() == "" or self.TrainNo_var.get() == "" or
self.St_Code_var.get() == "" or self.A_Time_var.get == "" or self.D_Time_var.get() == "":
            messagebox.showerror("Error","All fields are required")
        else:
            con = cx_Oracle.connect("railway/railway007")
            cur = con .cursor()
            cur.execute("insert into manager values (:1,:2,:3,:4,:5,:6)",
                    (
                        self.From_var.get(),
                        self.To_var.get(),
                        self.TrainNo_var.get(),
                        self.St_Code_var.get(),
                        self.A_Time_var.get(),
                        self.D_Time_var.get()
                    ))
            con.commit()
            self.fetch_data_to_searchby()
            self.clear()
            con.close()
            messagebox.showinfo("Success","Record has been inserted")
```

```
    #To show data from Train_table to horizontal section view ---- trieview
    def fetch_data_to_searchby(self):
        con = cx_Oracle.connect("railway/railway007")
        cur = con.cursor()
        cur.execute("select * from manager")
        rows = cur.fetchall()
        if len(rows) != 0:
            self.Train_table.delete(*self.Train_table.get_children())
            for row in rows:
                self.Train_table.insert('',END,values=row)
            con.commit()
        con.close()

    def clear(self):
        self.From_var.set("")
        self.To_var.set("")
        self.TrainNo_var.set("")
        self.St_Code_var.set("")
        self.A_Time_var.set("")
        self.D_Time_var.set("")

    #TO get data from horizontal to Train_table
    def get_data_via_cursor(self,event):#takes 1 positional argument
        cursor_row = self.Train_table.focus()
        contents = self.Train_table.item(cursor_row)
        row = contents['values'] #will return list
        self.From_var.set(row[0])
        self.To_var.set(row[1])
        self.TrainNo_var.set(row[2])
        self.St_Code_var.set(row[3])
        self.A_Time_var.set(row[4])
        self.D_Time_var.set(row[5])

    def update_data(self):
        con = cx_Oracle.connect("railway/railway007")
        cur = con.cursor()
        fr = self.From_var.get()
        to = self.To_var.get()
        tn = self.TrainNo_var.get()
        sc = self.St_Code_var.get()
        at = self.A_Time_var.get()
        dt = self.D_Time_var.get()
        cur.execute("update manager set From_ = :1, To_ = :2, Station_code = :3, Arrival_time = :4,
Departure_time = :5 where Train_no = :6",
                    {
                     '1' : fr,
                     '2' : to,
                     '3' : sc,
                     '4' : at,
                     '5' : dt,
                     '6' : tn
                    })
        con.commit()
        self.fetch_data_to_searchby()
        self.clear()
```

```
        con.close()

    def delete_data(self):
        con = cx_Oracle.connect("railway/railway007")
        cur = con.cursor()
        tn = self.TrainNo_var.get()
        cur.execute("delete from manager where Train_no = :1", {'1' : tn})
        con.commit()
        self.fetch_data_to_searchby()
        self.clear()
        con.close()

    def search_data(self):
        con = cx_Oracle.connect("railway/railway007")
        cur = con.cursor()
        cur.execute("select * from manager where " + str(self.search_by.get()) + " = " + str(self.search_txt.get()))
        rows = cur.fetchall()
        if len(rows) != 0:
            self.Train_table.delete(*self.Train_table.get_children())
            for row in rows:
                self.Train_table.insert(', END, values=row)
            con.commit()
        con.close()

root = Tk()
ob = Manager()
root.mainloop()
```

BOOKING MODULE

```
from tkinter import *
from tkinter import ttk, messagebox
import cx_Oracle
import random
class Booking:
    def __init__(self):
        self.root = root
        self.root.title("TICKET BOOKING SYSTEM")
        self.root.geometry("1350x700+0+0")
        self.root.config(bg="#021e2f")

        title = Label(self.root, text="WELCOME, PLEASE BOOK YOUR TICKET HERE", bd=10,
relief=GROOVE, font=("times new roman",40,"bold"), bg="#021e2f", fg="Orange")#fg=textcolor
        title.pack(side=TOP, fill=X)

        #------------ALL VARIABLES-------------
        self.From_var = StringVar()
        self.To_var = StringVar()
        self.TrainNo_var = StringVar()
        self.St_Code_var = StringVar()
        self.A_Time_var = StringVar()
        self.D_Time_var = StringVar()
```

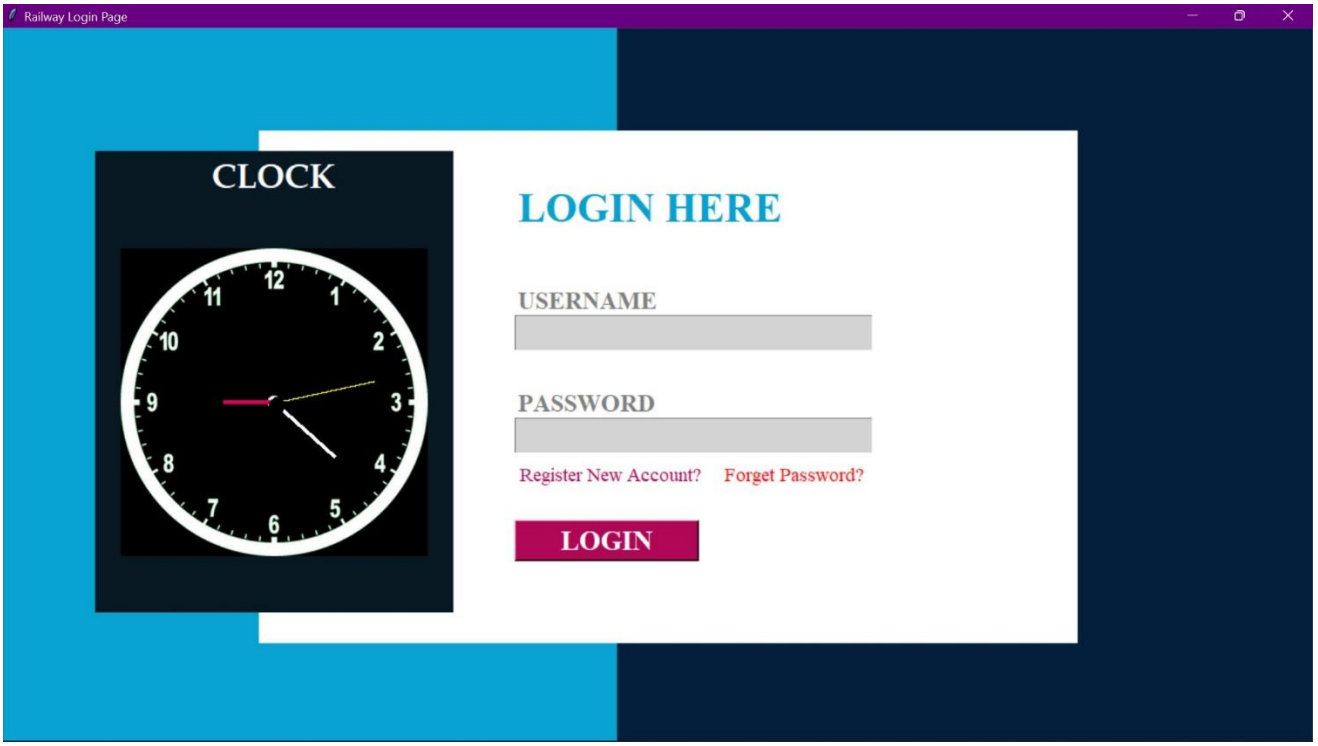
```
        self.search_by = StringVar()
        self.search_txt = StringVar()

        self.ticket_no = StringVar()
        #------------MANAGER FRAME-------------
        Manage_Frame = Frame(self.root,bd=4,relief=RIDGE,bg="#373E98")
        Manage_Frame.place(x=20,y=100,width=475,height=580)

        m_title = Label(Manage_Frame,text="Book Train ",bg="#373E98",fg="white",font=("times new
roman",30,"bold"))
        m_title.grid(row=0,columnspan=2,pady=20)
        #------From
        lbl_from = Label(Manage_Frame, text="From", bg="#373E98", fg="white",font=("times new roman", 20,
"bold"))
        lbl_from.grid(row=1, column=0, pady=10, padx=20, sticky="W")

        txt_from = Entry(Manage_Frame,textvariable=self.From_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_from.grid(row=1, column=1, pady=10, sticky="W")
        #-------To
        lbl_to = Label(Manage_Frame, text="To", bg="#373E98", fg="white",font=("times new roman", 20,
"bold"))
        lbl_to.grid(row=2, column=0, pady=10, padx=20, sticky="W")

        txt_to = Entry(Manage_Frame,textvariable=self.To_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_to.grid(row=2, column=1, pady=10, sticky="W")
        #-------TrainNo
        lbl_train_no = Label(Manage_Frame, text="TrainNo.  ", bg="#373E98", fg="white",font=("times new
roman", 20, "bold"))
        lbl_train_no.grid(row=3, column=0, pady=10, padx=20, sticky="W")

        txt_train_no = Entry(Manage_Frame,textvariable=self.TrainNo_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_train_no.grid(row=3, column=1, pady=10, sticky="W")
        #-------St-Code
        lbl_station_code = Label(Manage_Frame, text="St-Code", bg="#373E98", fg="white",font=("times new
roman", 20, "bold"))
        lbl_station_code.grid(row=4, column=0, pady=10, padx=20, sticky="W")

        txt_station_code = Entry(Manage_Frame,textvariable=self.St_Code_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_station_code.grid(row=4, column=1, pady=10, sticky="W")
        #--------A-Time
        lbl_arrivaltime = Label(Manage_Frame, text="A-Time", bg="#373E98", fg="white",font=("times new
roman", 20, "bold"))
        lbl_arrivaltime.grid(row=5, column=0, pady=10, padx=20, sticky="W")

        txt_arrivaltime = Entry(Manage_Frame,textvariable=self.A_Time_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_arrivaltime.grid(row=5, column=1, pady=10, sticky="W")
        #--------D-Time
        lbl_departuretime = Label(Manage_Frame, text="D-Time", bg="#373E98", fg="white",font=("times new
roman", 20, "bold"))
        lbl_departuretime.grid(row=6, column=0, pady=10, padx=20, sticky="W")
```

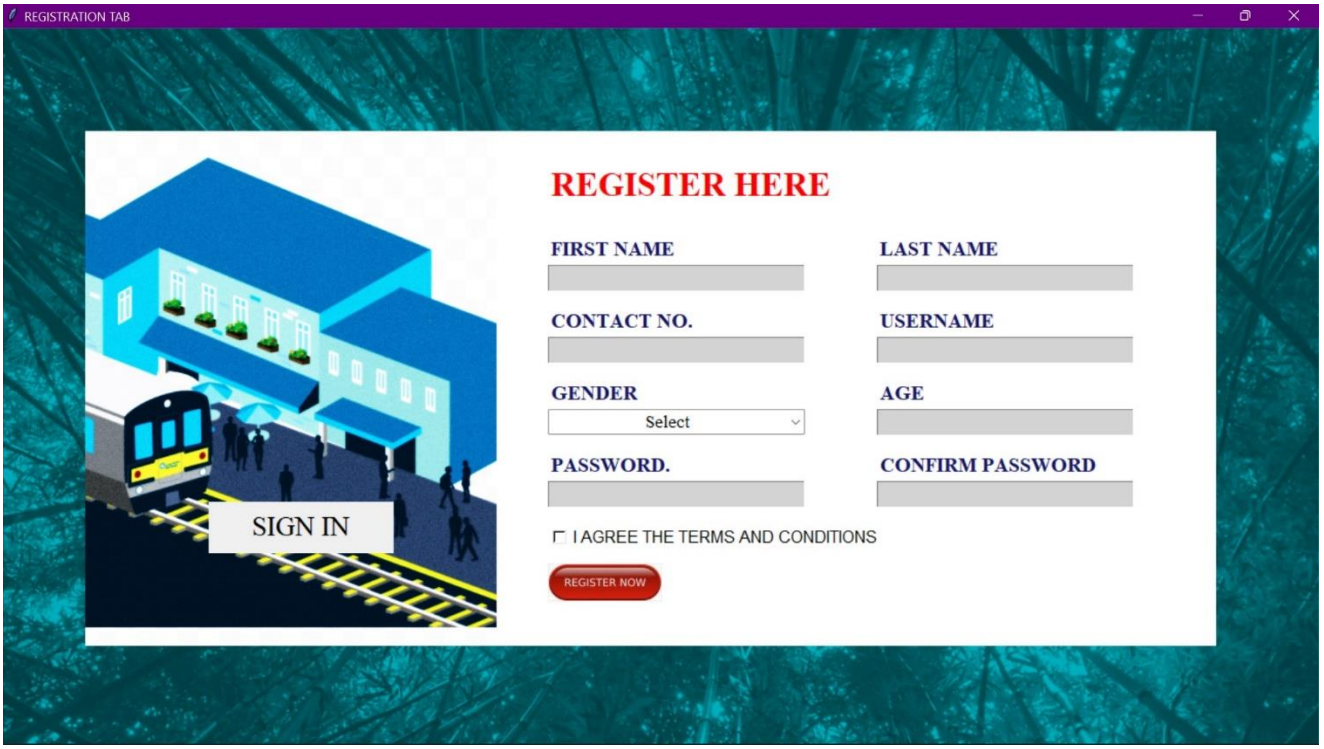
```
        txt_departuretime = Entry(Manage_Frame,textvariable=self.D_Time_var,font=("times new roman", 20,
"bold"),bd=5,relief=GROOVE)
        txt_departuretime.grid(row=6, column=1, pady=10, sticky="W")

        #-------BUTTON FRAME----------------
        btn_Frame = Frame(Manage_Frame,bd=4,relief=RIDGE,bg="orange")
        btn_Frame.place(x=30,y=500,width=400)

        bookbtn = Button(btn_Frame,text="BOOK
TICKET",width=20,height=2,command=self.add_train_info).grid(padx=120,pady=10)


        # ------------DETAIL FRAME-------------
        Detail_Frame = Frame(self.root, bd=4, relief=RIDGE, bg="#373E98")
        Detail_Frame.place(x=500, y=100, width=770, height=580)

        lbl_search = Label(Detail_Frame, text="Search By", bg="#373E98", fg="white", font=("times new roman",
20, "bold"))
        lbl_search.grid(row=0, column=0, pady=10, padx=20, sticky="W")

        combo_search = ttk.Combobox(Detail_Frame,textvariable=self.search_by,width=15,font=("times new
roman",13,"bold"),state="readonly")
        combo_search['values'] = ('From_', 'To_')
        combo_search.grid(row=0,column=1,padx=20,pady=10)

        txt_search = Entry(Detail_Frame,textvariable=self.search_txt, font=("times new roman", 10, "bold"), bd=5,
relief=GROOVE)
        txt_search.grid(row=0, column=2, pady=10 ,padx=20 , sticky="w")

        searchbtn = Button(Detail_Frame, text="Search",
width=10,pady=5,command=self.search_data).grid(row=0, column=3, padx=10, pady=10)
        showallbtn = Button(Detail_Frame, text="Show All",
width=10,pady=5,command=self.fetch_data_to_searchby).grid(row=0, column=4, padx=10, pady=10)

        #---------Table Frame--------------
        Table_Frame = Frame(Detail_Frame, bd=4, relief=RIDGE, bg="orange")
        Table_Frame.place(x=10, y=70, width=745, height=500)

        scroll_x = Scrollbar(Table_Frame,orient=HORIZONTAL)
        scroll_y = Scrollbar(Table_Frame,orient=VERTICAL)
        self.Train_table = ttk.Treeview(Table_Frame,columns=("From","To","TrainNo","St-Code","A-Time","D-
Time"),xscrollcommand=scroll_x.set,yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT,fill=Y)
        scroll_x.config(command=self.Train_table.xview)
        scroll_y.config(command=self.Train_table.yview)
        self.Train_table.heading("From",text="From")
        self.Train_table.heading("To",text="To")
        self.Train_table.heading("TrainNo",text="TrainNo")
        self.Train_table.heading("St-Code",text="St-Code")
        self.Train_table.heading("A-Time",text="A-Time")
        self.Train_table.heading("D-Time",text="D-Time")
        self.Train_table['show'] = 'headings'
        # Train_table.column("From",width=50) ---- IF YOU WANT TO ADJUST THE WIDTH
        self.Train_table.pack(fill=BOTH,expand=1)
        self.Train_table.bind("<ButtonRelease-1>", self.get_data_via_cursor) #event
```

```
        self.fetch_data_to_searchby()

    def add_train_info(self): #had to give two arguments
        if self.From_var.get() == "" or self.To_var.get() == "" or self.TrainNo_var.get() == "" or
self.St_Code_var.get() == "" or self.A_Time_var.get == "" or self.D_Time_var.get() == "":
            messagebox.showerror("Error","All fields are required")
        else:
            self.ticket_no = str(random.randint(100,10000)) +
str(random.choice("ABCDEFGHIJKLMNOPQRSTUVWXYZ"))
            con = cx_Oracle.connect("railway/railway007")
            self.ticket_no,
            self.From_var.get(),
            self.To_var.get(),
            self.TrainNo_var.get(),
            self.St_Code_var.get(),
            self.A_Time_var.get(),
            self.D_Time_var.get()
            cur = con .cursor()
            cur.execute("insert into ticket (ticket_no,from_,to_,train_no,station_code,arrival_time,departure_time)
values (:1,:2,:3,:4,:5,:6,:7)",
                        (       self.ticket_no,
                                self.From_var.get(),
                                self.To_var.get(),
                                self.TrainNo_var.get(),
                                self.St_Code_var.get(),
                                self.A_Time_var.get(),
                                self.D_Time_var.get()
                        ))
            con.commit()

            # self.clear()
            con.close()
            messagebox.showinfo("TICKET NUMBER", "Your Ticket Number is "+self.ticket_no)
            messagebox.showinfo("Success","Train has been booked")
            self.root.destroy()

    #To show data from Train_table to horizontal section view ---- trieview
    def fetch_data_to_searchby(self):
        con = cx_Oracle.connect("railway/railway007")
        cur = con.cursor()
        cur.execute("select * from manager")
        rows = cur.fetchall()
        if len(rows) != 0:
            self.Train_table.delete(*self.Train_table.get_children())
            for row in rows:
                self.Train_table.insert(',END,values=row)
            con.commit()
        con.close()

    #TO get data from horizontal to Train_table
    def get_data_via_cursor(self,event):#takes 1 positional argument
        cursor_row = self.Train_table.focus()
        contents = self.Train_table.item(cursor_row)
        row = contents['values'] #will return list
        self.From_var.set(row[0])
        self.To_var.set(row[1])
```

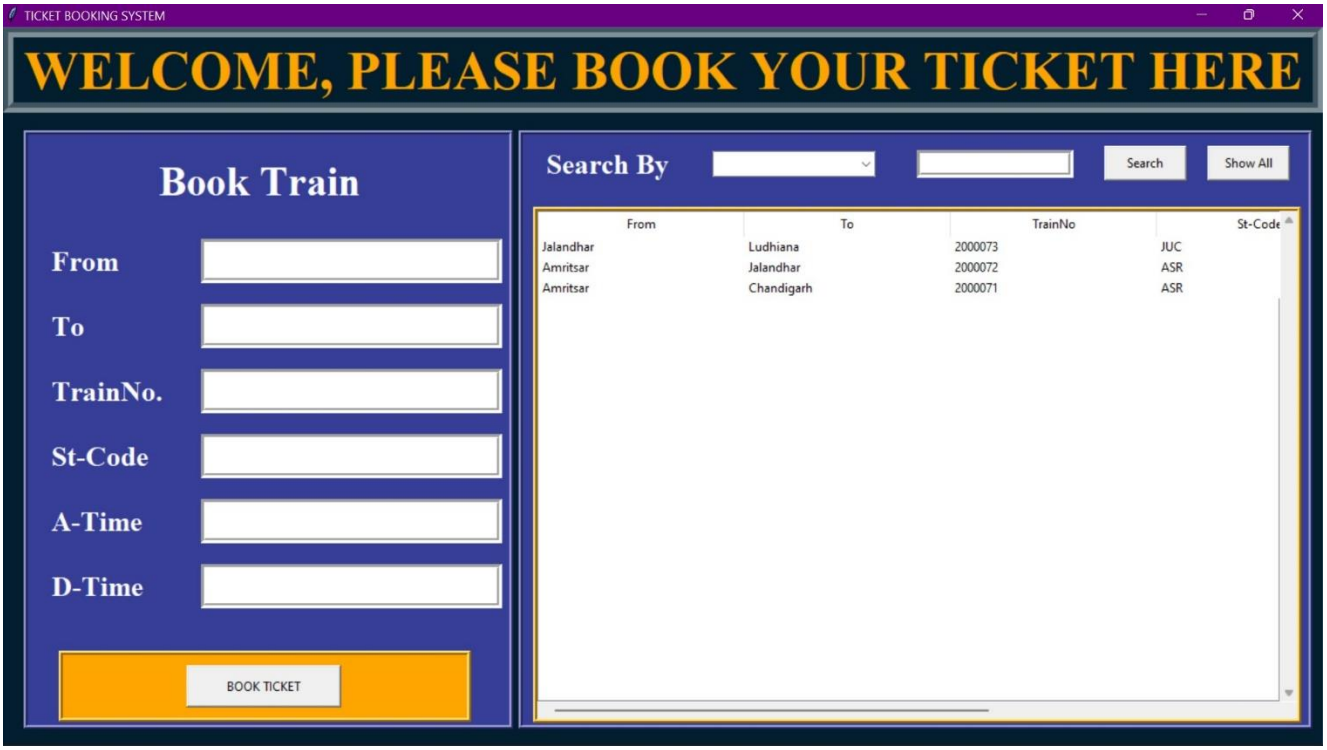
```
        self.TrainNo_var.set(row[2])
        self.St_Code_var.set(row[3])
        self.A_Time_var.set(row[4])
        self.D_Time_var.set(row[5])


    def search_data(self):
        con = cx_Oracle.connect("railway/railway007")
        cur = con.cursor()
        cur.execute("select * from manager where " + str(self.search_by.get()) + " Like '" + str(self.search_txt.get()
+"'"))
        rows = cur.fetchall()
        if len(rows) != 0:
            self.Train_table.delete(*self.Train_table.get_children())
            for row in rows:
                self.Train_table.insert('', END, values=row)
            con.commit()
        con.close()

root = Tk()
ob = Booking()
root.mainloop()
```

PROJECT PICTURES






RAILWAY MANAGEMENT SYSTEM
RAILWAY MANAGEMENT SYSTEM
Manage Train Routes
From
To
TrainNo.
St-Code
A-Time
D-Time
Add
Update
Delete
Clear
Search By
Search
Show All
From
To
TrainNo
St-Code
Jalandhar
Ludhiana
2000073
JUC
Amritsar
Jalandhar
2000072
ASR
Amritsar
Chandigarh
2000071
ASR


TICKET BOOKING SYSTEM
WELCOME, PLEASE BOOK YOUR TICKET HERE
Book Train
From
To
TrainNo.
St-Code
A-Time
D-Time
BOOK TICKET
Search By
Search
Show All
From
To
TrainNo
St-Code
Jalandhar
Ludhiana
2000073
JUC
Amritsar
Jalandhar
2000072
ASR
Amritsar
Chandigarh
2000071
ASR

REFERENCES:

- https://www.researchgate.net/publication/349265225_EXAMINATION_MANAGEMENT_SYS TEM
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3611554/A-Study-On-Web-Based-Online-Examination-System
- https://coderslegacy.com/python/getting-started-pyqt/
- https://zetcode.com/gui/pyqt5/firstprograms/
- https://ieeexplore.ieee.org/document/6011131
- https://stackoverflow.com/