

**National University of ScienceS & Technology**

**Fundamentals of Programming**

**cs-114**

**Semester Project Report**

**Instructor:**

**Dr. Zuhair Zafar**

**Submitted by:**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Name** | **Reg. No.** |
| **1** | Muhammad Bilal | 389994 |
| **2** | Abdul Arham | 374696 |
| **3** | Muhammad Ashhub Ali | 380078 |

**Train Reservation System**

**Modules:**

1. **PyQT5**

With the help of the ‘*PyQt5*’ module, we set up the GUI of our project, using its vast libraries of in-built classes and widgets, such as QLabel, QTab, QLineEdit, and many more.

1. **CSV**

The CSV module is used to store the data inputted by the customer into an excel file in the format of csv, i.e., Comma-Separated Values. We have specifically imported this csv module to store customer information so that it is easier to read and access the data of each customer. This aids in the View Booking as well as Cancel Booking features, because the customer’s input directly alludes to the required row in the Excel file.

1. **JSON**

The JSON module is used to read and write the data of each train, i.e., Train ID, Departure, Destination and seat availability in the form of dictionaries directly into a text file. This module allows the strings saved in the format of dictionaries to be directly read as a dictionary.

Saving each train separately as a dictionary helps ease of access, and makes it easier to know which seats in which trains have been booked and which are free.

1. **Calendar**

When taking user inputs such as Date of Birth and Date of Departure, we used the ‘*calendar’* module to access this information and convert it to the required data (Day of the week, etc.) using functions and classes designed for the Gregorian Calendar.

1. **Datetime**

The ‘*datetime’* module allowed us to work with time objects such as suggesting the nearest time slot available according to the user’s preferred time; as well as storing and accessing time in any particular format.

1. **Pandas**

The Pandas module has been used for the generation of csv files. The program generates a txt file for saving the customer information. Pandas module helps to convert the txt file into a csv to assist in storing data in the csv file or accessing the data.

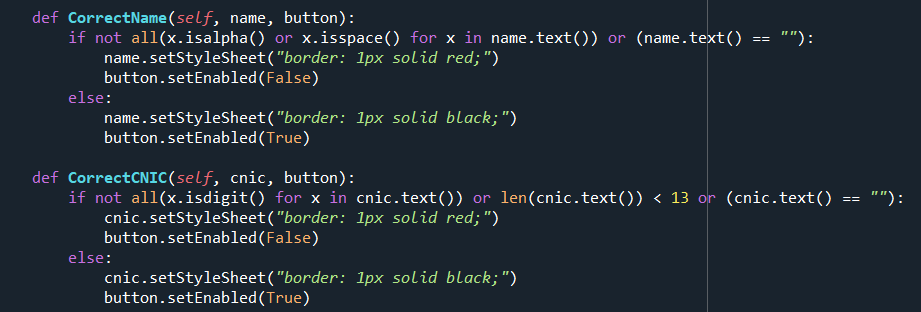
1. **OS**

The OS module is used to access the name of directory which is used for the conversion of the txt file into csv.

**Features & Working:**

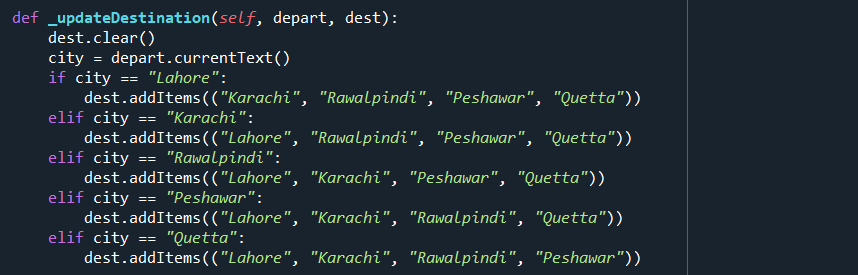
1. **Checking for Correct Input**

The code includes two functions *CorrectName* and *CorrectCNIC* that take two parameters, namely the *Input Box*, and the *Button* associated with it. Whenever the user types in the *Name Input Box*, the program calls for the *CorrectName* function. The function checks for each index of the text entered and changes the border color to red as well as disables the Submit button associated with it only if the input type is incorrect, i.e., it is not a String. Similarly, the code checks whether the text entered in the *CNIC Box* contains digits or not.



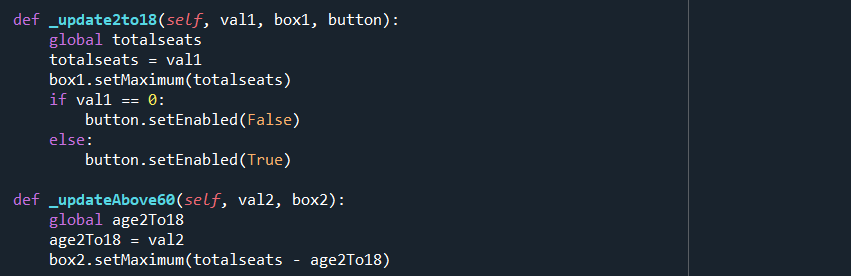
1. **Updating Destination**

The code defines a function that takes the *Departure City Combo Box* and *Destination City Combo Box* as parameters. After the user chooses the desired city of departure, the function removes it from the *Destination City Box*, allowing the user to only choose from the other cities. This avoids the user to choose a particular city as both Departure and Destination.



1. **Updating & Limiting Seat Count**

While selecting the number of seats during booking, the maximum number of available seats are updated every time the user changes their preference. The max number of Elderly seats and those for children also update constantly and their sum cannot be more than the total number of seats selected by user. The following functions are called to set the maximum value of *Spin Boxes:*



1. **File Maker**

The code includes three separate functions: *schedule\_maker*, *trains\_maker*, and *customer\_information*. These 3 functions create three separate files that save data entered by the user.

**Schedule\_maker:**

Creates a .txt file that includes the weekly schedule of commute from one city to another. We have included 5 cities: Peshawar, Karachi, Rawalpindi, Lahore and Quetta.

Each day, two trains leave each station and hence there are 2 timings per day per one city to another. The .txt file is created and saved as Schedule.txt

**Trains\_maker:**

Creates a .txt file that includes the data of each and every train that commutes from city to city. Each train is represented and saved as two separate dictionaries with a unique Train ID. One dictionary is for Business class seats and the other is for Economy class seats. The availability of each seat is represented by “true” as the value in its key-value pair. The value changes to “false” when the seat is booked, and so is not displayed to the next user that tries to book a seat. Two trains travel each day, hence there are 14 trains for the weekly schedule. As there are 2 dictionaries for each train, and 20 combinations of departure-destination city, there are 560 dictionaries that store the data of seat availability of 280 trains.

For the seat, the key represents the seat number while the value represents its availability, e.g.,

*“1”: True means seat 1 is available*

*“B1”: False means Seat 1 in berth B is booked.*

The text is saved as a string in the format of a dictionary. We have imported ‘*json*’ module to read each line directly as a dictionary, so we can make appropriate changes to the seat availability whenever the seat is booked.

To update each seat’s availability, the dictionaries are read into a list. The dictionary that contains the user’s booked seats is taken as a separate dictionary, and each seat booked by the user has its value changed to False. The original dictionary is removed from the list of dictionaries, and the new dictionary is appended into the list. The .txt file is opened in write format, and the list of dictionaries is updated into the .txt file.

**Customer\_information:**

This function creates a .csv file, i.e., comma-separated values. This file stores all the credentials of the customer, i.e., Name, CNIC, DOB etc. The csv file is saved in Excel format where the first row are the Headers while the next rows are each customer entry.

When the user enters their information in the GUI, the input is passed onto the corresponding function that takes the information into a dictionary. In this dictionary, the keys represent each header of the csv while the values represent the User’s corresponding information. This dictionary is then appended into the csv file, to store the user’s information.