**PROJECT DESCRIPTION**

1. AIM OF THE PROJECT:

The primary aim of this project is to develop a bus ticket reservation system where the passenger can book a number of tickets. This system offer a more convenient experience than visiting counter in person and also it including a features such as

* View bus information
* Checking seats availability
* Booking number of tickets
* Receive confirmation

By offering real-time updates on seat availability, it reduces the bother associated with travel planning. Additionally it incorporates the strategies that effectively manage unexpected situations or incorrect user inputs without causing the program to crash. Allowing the user to retry actions after an error rather than terminating the program. This ensures more user– friendly experience. Moreover this bus booking system making it easier for passengers to secure their desired journey with minimal effort.

2. BUSINESS PROBLEM OR PROBLEM STATEMENT :

In the rapidly evolving transportation sector, the demand for efficient and user-friendly bus booking solutions has become increasingly critical. Currently, most of the booking system lack error handling and validation mechanism which results in further complications when user enters a incorrect information. This not only frustrates passengers but also leads to confusion and long-wait.

To address these challenges, there is a need of bus ticket reservation system that enhances the ticket booking experience for passengers. Such a system offer real-time updates on seat availability, a straightforward booking process, and guiding users in clear manner through any potential issues. Inorder to simplify the booking process, it provides accurate availability information and enhances customer satisfaction. The need for an effective Bus Booking System is clear. Such a system should allow the passengers to check bus availability, make reservations, and receive timely confirmations. By integrating booking functionalities, the system can significantly enhance the overall user experience.

By addressing the challenges through the development this proposed system includes an error handling to guide users through any issues they may encounter during the booking process and easy-to-follow prompts, the system can create a more pleasant user experience while minimizing operational burdens.

3. PROJECT DESCRIPTION:

All the manual systems can be digitized just by using digital solution like online or bus reservation system. This project involves developing a Bus Ticket Reservation system using OOPS, error and exception handling in python, where the code containing class inheritance, method overriding, user interaction and file handling. Here the methodologies used are

1. Viewing Bus details
2. Checking seat availability
3. Booking seats
4. Receive confirmation
5. Error handling

The main thing is it is a user-friendly system and manage their bus travel efficient and also scheduling of bus is a major addition . It help in providing adequate data to the passenger. Safely managing file operations like reading bus details to handle situations where the file might not exist or is inaccessible. This includes using try and except blocks to catch and respond to potential errors. Providing clear and informative messages when an error occurs, guiding users on how to correct their inputs (e.g., specifying the correct format for dates or integers). The system checks for seat availability and, if available, collects further details such as passenger name, A/C preference, slot number, and booking date. After confirming the booking, the system displays the booking details to the user. By delivering clear and concise confirmation, the system enhances user confidence, ensuring that travelers feel secure in their arrangements and prepared for their upcoming journey. Additionally, by ensuring that every step of the booking process is communicated clearly, the system fosters a sense of security among users. Passengers can feel confident in their arrangements, knowing they have reliable access to their booking information and support if needed. This emphasis on clarity and user support ultimately contributes to a more enjoyable travel experience, making the system a valuable tool for anyone planning a bus journey.

4. FUNCTIONALITIES:

The bus ticket reservation system is designed to streamline the functionalities which are,

* View Bus Information:

This functionality allows users to easily access information about available buses for their travel they can access details about available buses, including bus numbers and the number of available seats. This feature is designed to be user-friendly, ensuring that travelers can quickly find the information they need without unnecessary complications. By presenting clear and concise bus details, the system facilitates a seamless planning process, ultimately improving the overall efficiency of travel arrangements.

* Checking seats availability:

It is a crucial component of the system that enables users to determine whether their desired number of seats is available. Once a user selects a bus, they can check seat availability. If the required number of seats are available, users are prompted to proceed with the booking. The system verifies by comparing the requested seat count against the bus available seats.

If the seats are available, users receive a confirmation message indicating that they can proceed with the booking. Conversely, if the requested number exceeds the available seats, the system alerts users, suggesting them to reduce the number of seats or consider alternative buses. By ensuring that users can quickly ascertain availability without unnecessary delays and seamlessly integrating seat availability checks into the booking process, the system minimizes the risk of disappointment.

* Booking Seats:

Once users have confirmed seat availability for their chosen bus, they can proceed to book the desired number of seats.

During this process, users are prompted to enter essential information, including:

**Passenger Name:** The name of the individual travelling.

**Seat Preference:** Users can specify whether they want an A/C or Non A/C coach with either sleeper or semi sleeper depending on their comfort needs.

**Slot Number:** This refers to the type morning or evening that users wish to reserve.

**Travel Date:** The date on which they intend to travel, ensuring the booking is correctly aligned with their travel plans

* Receive confirmation:

Once a booking is successfully completed, the system generates a confirmation message summarizing the key details of the transaction. This message serves detailing their booking information, including the number of seats reserved, the travel date for the purpose of reassuring users that their seats are secured and that all necessary information has been captured accurately. By delivering clear and concise confirmation, the system enhances user confidence, ensuring that travelers feel secure in their arrangements and prepared for their upcoming journey. This feature plays a significant role in creating a smooth and reliable booking experience, ultimately contributing to customer satisfaction.

* Error Handling:

It includes mechanisms for handling invalid inputs such as incorrect date formats and seat requests that exceed availability. Ensuring the user inputs, such as bus numbers and seat counts, are valid integers. This prevents type errors when the program attempts to process these inputs.

Checking the date entered by the user is in the correct format and is not a past date. If the input is invalid, the system prompts the user to enter a valid date.

Safely managing file operations using try and except blocks to catch and respond to potential errors.

Allowing users to retry by providing clear and informative messages after an error rather than terminating the program, thereby enhancing the overall user experience.

5. INPUT VERSATALITY AND ERROR HANDLING:

In a bus ticket reservation system, input versatility refers to the system's ability to accept a variety of inputs from users while maintaining accuracy and efficiency.

Managing file operations like reading bus details there is a need to handle the situations like file may not exist or asking permission to read a file where these cases are overcome by proving try and except block.

Users should be able to input data in specified formats. (e.g.,DD-MM-YYYY) to reduce confusion. This flexibility minimizes the chances of input errors related to date formats.

Checking the date entered by the user is in the correct format and is not a past date. If the input is past date, the system prompts the user to enter a valid date. This includes using try and except blocks to catch and respond to different kind of errors.

Instead of vague prompts like "Invalid input," more descriptive messages such as "Please enter a valid date," guide users in the right direction. When errors occur, the system should provide clear and informative messages that help users understand what went wrong and how to rectify it.

The system should offer users a way to easily recover from errors. For instance, if a user fails to enter a date correctly, the system can guide them back to the date input section, highlighting the error, rather than requiring them to start the entire booking process over again.

6. CODE IMPLEMENTATION:

In this project, we implement various modules using basic Python programming concepts. Each module is designed to handle specific functionalities of the bus ticket reservation system.

from datetime import datetime

from abc import ABC, abstractmethod

class Bus: “””Parent class “””

def \_\_init\_\_(self, bus\_no):

self.bus\_no = bus\_no

self.available\_seats = 15 “”” Fixed available seats”””

def bus\_info(self):

return f"Bus\_No: {self.bus\_no}\tAvailable Seats: {self.available\_seats}"

class IsAvailable(Bus): “””Inheriting from parent class(Bus)”””

def \_\_init\_\_(self, bus\_no, seat\_confirm):

super().\_\_init\_\_(bus\_no)“””super() allows us to access methods of Bus“””

self.seat\_confirm = seat\_confirm

‘’’Checking available seats ‘’’

def is\_available\_info(self):

if self.available\_seats >= self.seat\_confirm:

print("Seats are available. You may proceed.")

return True

else:

print("There are not enough available seats.")

return False

class Booking(ABC, IsAvailable): “””Inherit from ABC”””

def \_\_init\_\_(self, bus\_no, seat\_confirm):

super().\_\_init\_\_(bus\_no, seat\_confirm)

Args – Passenger name, AC, Slot, Date getting from user

self.passenger\_name = input("Enter passenger's name: ")

self.slot = int(input('Enter a slot number:'))

self.AC, self.seat\_comfort = self.get\_AC() # Get both AC preference and seat comfort

self.date = self.get\_date()

@abstractmethod

def get\_AC(self):

"""Abstract method to get the booking date."""

pass

@abstractmethod

def get\_date(self): “””Used to hide the complex instruction of get\_date”””

"""Abstract method to get the booking date."""

pass

class ConcreteBooking(Booking):

“””Concrete implementation of Booking”””

def get\_AC(self):

while True:

try:

seat\_detail = input('Do you want AC or Non-AC? ').strip().upper()

if seat\_detail not in ['AC', 'NON-AC']:

raise ValueError("Invalid option. Please enter 'AC' or 'Non-AC'.")

seat = input('You want sleeper or semi-sleeper: ')

print(f'{seat\_detail} {seat} is available for you')

return seat\_detail, seat # Return both comfort type and seat type

except ValueError as ve:

print(ve)

def get\_date(self): “””For valid booking date from user”””

while True:

date\_str = input("Enter date in (DD-MM-YYYY): ")

try: “””Converting string date into datetime object”””

date = datetime.strptime(date\_str, '%d-%m-%Y')

if date < datetime.now():

print("The date cannot be in the past. Please enter a valid date.")

continue

return date

except ValueError:

print('Please enter the date in correct format (DD-MM-YYYY)')

def booking\_info(self): “””display confirmation message”””

return (f"\nHi {self.passenger\_name}! You have booked "

f"{self.seat\_confirm} seats with {self.AC}, slot {self.slot} on "

f"{self.date.strftime('%d-%m-%Y')}.")

def main():

while True:

try:

“”” Attempt to open and read bus details from a file”””

with open(r'C:\Users\Lenovo\Documents\bus\_details.txt', 'r', encoding='UTF-8') as fileobj:

details = fileobj.read()

print(details)

except FileNotFoundError:

print('Bus details file not found.')

return

except IOError: “””Handle other I/O errors (e.g., permission issues)”””

print("Error: Unable to read the bus details file.")

return

while True:

bus\_check = int(input("\nEnter 1 for booking or 2 for exit: "))

if bus\_check == 1:

bus\_no = int(input("Enter bus number: "))

seat\_confirm = int(input("Enter number of seats to book: "))

bus = IsAvailable(bus\_no, seat\_confirm) “”” Create instance of IsAvailable”””

print(bus.bus\_info()) “”” Display bus information”””

if bus.is\_available\_info():

booking = ConcreteBooking(bus\_no, seat\_confirm)

print(booking.booking\_info())

else:

print("Sorry,You may check with less number of seats...")

elif bus\_check == 2:

print('Thank you for visiting.')

return “”” Exit the program”””

else:

print("Invalid input. Please enter 1 or 2.")

if \_\_name\_\_ == "\_\_main\_\_":

main() “”” Run the main function”””

The code implements a bus booking system using object-oriented programming. It defines classes for Bus, IsAvailable, Booking, and Concrete Booking.

7. RESULTS AND OUTCOMES:

We achieved a user-friendly bus booking system that checks seat availability, facilitates reservations, and includes error handling and input validation. We improved user experience by implementing real-time seat availability checks, clear booking confirmations and input validations to prevent errors, such as ensuring the booking date isn’t in the past. Additionally, robust error handling for file access issues enhances reliability, providing users with informative feedback throughout the booking process. The incorporation of error handling ensures that users receive immediate response, reducing frustration. Overall, the system promotes efficiency, reliability, and a positive booking experience.

8. CONCLUSION:

The bus booking system potrays a seamless and efficient approach to managing travel reservations . It is a transformative tool that significantly enhances the travel experience for passengers while improving operational efficiency.

It emphasizes the importance of user experience by providing intuitive interactions, real-time seat availability checks. The system highlights the necessity of validating user input to prevent errors, ensuring reliability and trustworthiness. By combining essential features like error handling and guiding mechanisms, the system demonstrates a commitment to enhancing customer satisfaction. Overall, it reflects a modern solution to travel management, aiming to meet user needs with clarity and efficiency. By prioritizing user experience, it effectively addresses common pain points in reservations.