Project Title: Railway Management System (RMS)

Objective

Design and implement a **command-line based Railway Management System** in Python that simulates core railway booking operations such as managing trains, booking tickets, canceling tickets, viewing passenger details, and checking PNR status. This project should demonstrate the use of:

- Object-Oriented Programming (OOP)
- Exception handling
- File handling
- Appropriate data structures for in-memory operations

Functional Requirements

---

1. Train Management

Class: Train

Attributes:

- train_id (int): Unique integer identifier
- name (str): Name of the train
- seats (int): Total available seats

Methods:

- __init__(self, train_id, name, seats): Constructor to initialize train details.
- get_available_seats(self): Returns the number of available seats.
- book_seat(self): Decrements seat count by one when a seat is booked.
- cancel_seat(self): Increments seat count by one when a booking is canceled.
- Display_train_info(self): String representation of train info for display.

Data Structure:

- Store trains in a list:
  self.trains = [] (List of Train objects)
  This allows easy iteration and searching.

---

2. User and Passenger Management

Class: Person

Attributes:

- name (str)
- age (int)
- gender (str)

Methods:

- __init__(self, name, age, gender)

Class: Passenger (inherits from Person)

Attributes:

- pnr (str): Unique ticket number
- train_id (int): Associated train
- timestamp (datetime): Booking time

Methods:

- __init__(self, name, age, gender, pnr, train_id, timestamp)
- Display_passenger_info(self): String representation for passenger details

Data Structure:

- Store passengers in a dictionary keyed by PNR:
  self.passengers = {pnr: Passenger}
  This provides efficient cancellation, and status checking.

---

3. Ticket Booking

Implemented In: RailwayManagement class

Methods:

- add_train(self, Train):

  add trains to self.trains[]

- display_train(self,Train)
  display all trains

- book_ticket(self, train_id, name, age, gender):
  Validates train_id, checks seat availability, generates unique PNR, saves booking in-memory and to file, returns booking confirmation.

- generate_pnr(self, train_id):
  Generates a unique PNR in format train_id-DDMMYY-RANDOMSTRING using datetime and random modules.

- def find_train(self, train_id):

  Finds the train with its ID using iteration

- cancel_ticket(self, pnr):
  Removes passenger by PNR, increments train seat count, raises PNRNotFoundError if PNR not found.

- get_passengers_by_train(self, train_id):
  Returns a list of passengers for the given train using filter() and lambda.

- check_pnr_status(self, pnr):
  Returns passenger details and booking status for a given PNR, raises PNRNotFoundError if invalid.

- save_booking_to_file(self ,passenger):
  Append booking details to bookings.txt.

Error Handling

Custom Exceptions:

class BookingError(Exception):

```
    pass


class PNRNotFoundError(Exception):
    pass
```

When to Raise:

- BookingError
  For invalid train IDs or no seats available during booking.
- PNRNotFoundError
  For invalid or missing PNRs during cancellation or status checks.

File Handling

File: bookings.txt

PNR,Name,Age,Gender,TrainID,Timestamp

Write all details in file

Recommended File Structure

railway_management/

```
├── main.py             # Entry point for CLI
├── models.py           # Train, Person, Passenger classes
├── management.py       # RailwayManagement class with booking logic
├── exceptions.py       # BookingError, PNRNotFoundError exceptions
├── bookings.txt        # Booking records (created at runtime)
```

Summary of Data Structures

Component	Data Structure	Purpose
Trains	List[Train]	Store all train objects, easy to iterate/find next available train
Passengers	Dict[str, Passenger]	Quick lookup by PNR for booking, cancellation, status checking
Booking Records	Text file (bookings.txt)	Persistent booking storage

Coding Template

#models.py

```
class Train:
    def __init__(self, train_id, name, seats):
        self.train_id = train_id
        self.name = name
        self.seats = seats
        self.booked_seats = 0

    def get_available_seats(self):
        pass   # To Do: return True if seats > 0

    def book_seat(self):
        pass     #  To Do: reduce seat by 1 if available

    def cancel_seat(self):
        pass      #  To Do: increase seat by 1
```

```
    def display_train_info(self):
        return f"Train ID: {self.train_id}, Name: {self.name}, Available Seats: {self.get_available_seats()}"

class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

class Passenger(Person):
    def __init__(self, name, age, gender, pnr, train_id, timestamp):
        super().__init__(name, age, gender)
        self.pnr = pnr
        self.train_id = train_id
        self.timestamp = timestamp

    def display_passenger_info(self):
        return (f"PNR: {self.pnr}, Name: {self.name}, Age: {self.age}, Gender: {self.gender}, "
                f"Train ID: {self.train_id}, Booking Time: {self.timestamp}")

#exceptions.py
class BookingError(Exception):
    """Raised when train ID is invalid or no seat is available"""
```

```
    pass

class PNRNotFoundError(Exception):
    """Raised when PNR is invalid"""
    pass

#management.py
import random
from datetime import datetime

class RailwayManagement:
    def __init__(self):
        self.trains = []
        self.passengers = {}

    def add_train(self, train):
        pass  #  Add new Train to self.trains

    def display_trains(self):
         pass  # Loop through trains and print details

    def find_train(self, train_id):
        """        Find a train by its ID.
        Returns the train object if found, else None."""
        pass # Loop through self.trains and return the train with matching train_id
```

```
def generate_pnr(self, train_id):
    pass  # Return formatted PNR: train_id-DDMMYY-RANDOM


def book_ticket(self, train_id, name, age, gender):
    """
    TODO:
    1. Find train by train_id.
    2. If seats are available, book and generate PNR.
    3. If not available, then raise BookingError.
    4. Store booking details file (dict with pnr, passenger, train_id, timestamp).
    """
    pass  # Full booking flow: check seat, assign, create passenger


def cancel_ticket(self, pnr):
    """
    TODO:
    1. Find booking by PNR ,if PNR not found raise PNRNotFoundError.
    2. Remove booking and increment train's seat count.
    """
    pass  # Cancel booking by removing from dict and restoring seat


def get_passengers_by_train(self, train_id):
    """
    Get a list of all passengers for a specific train using train_id.
    """
    pass
```

```
    def check_pnr_status(self, pnr):
        """
        TODO:
        1. Find booking by PNR.
        2. Display details if found, else raise PNRNotFoundError.
        """
        pass

    def save_booking_to_file(self, passenger):
            pass  #  To Do: write booking details to file

#main.py
#import necessary modules

system = RailwayManagement()

# Preload some trains
system.add_train(Train(101, "Rajdhani Express", 5))
system.add_train(Train(102, "Shatabdi Express", 3))
system.add_train(Train(103, "Duronto Express", 2))

while True:
    print("\n===== Railway Management System =====")
    print("1. Display Trains")
```

```
print("2. Book Ticket")
print("3. Cancel Ticket")
print("4. Check PNR Status")
print("5. View Passengers by Train")
print("6. Exit")

choice = input("Enter your choice (1-6): ")

try:
    if choice == '1':
        pass # Display all trains with their info

    elif choice == '2':
        """ TODO:
        1. Get train_id, name, age, gender
        2. book ticket using system.book_ticket()
        3. dispplay passenger info on success
        """
        pass



    elif choice == '3':
        """ TODO:
        1. Get PNR from user.
        2. Call system.cancel_ticket(pnr)
        3. Display success message.
```

```
    """
    pass

elif choice == '4':
    """ TODO:
    1. Get PNR from user.
    2. Call system.check_pnr_status(pnr)
    3. Display status.
    """
    pass

elif choice == '5':
    """ TODO:
    1. Get train_id from user.
    2. Call system.get_passengers_by_train(train_id)
    3. Display passengers.
    """
    pass

elif choice == '6':
    print("Thank you for using the Railway Management System.")
    break

else:
    print("Invalid choice! Please try again.")
```

```
except BookingError as be:
    print(f"Booking Error: {be}")
except PNRNotFoundError as pe:
    print(f"PNR Error: {pe}")
except Exception as e:
    print(f"Unexpected Error: {e}")
```