Test Plan

Happy Paths

The happy path in the system is best shown by the activity diagram. I have built a text based interface for the Railways Booking Software application, which has validators and checks after each input from the user to bring back the system onto the happy path.

The system can never go into an unstable or unknown state as the interface keeps prompting the user for valid inputs and thus reverting back the system from the exceptional path back to the happy path.

Exceptional Paths

The exceptional path in the system is triggered whenever the user inputs an inconsistent and/or invalid data. The system is built to detect this immediately and inform the user about the inconsistency and prompting the user to input the data again.

For example: The user enters the aadhaar Number incorrectly {1234567890ABC} There are letters in this aadhaar Number and is thus invalid. The program immediately detects this and prompts the user for the aadhaar Number again and again till a correct and valid aadhaar Number is entered.

Unit Tests

Note: Positive Test Cases are marked with + and negative test cases are marked with - .

Unit Testing Name Class

Test Scenarios for Construction of Objects

Consider the static createName(std::string firstName_val,std::string middleName_val,std::string lastName_val) constructor. The test scenario is:

- +The data members are correctly initialized upon giving a valid name.
- - Check if exception is thrown upon trying to create an invalid name.
    - First and Last Name both are not provided.

Test Scenarios for Checking the GetFirstName() member function

Consider the GetFirstName() member function. The test scenario is:

- +Check if the function returns the first name correctly.

Test Scenarios for Checking the GetMiddleName() member function

Consider the GetMiddleName() member function. The test scenario is:

- +Check if the function returns the middle name correctly.

Test Scenarios for Checking the GetLastName() member function

Consider the GetLastName() member function. The test scenario is:

- +Check if the function returns the last name correctly.

Test Scenarios for insertion operator

Consider the overloaded insertion operator std::ostream& operator<<(std::ostream&, const Name&). The test scenario is:

- +Check if the operator inserts the name to the console in the right format.

Unit Testing Date Clas

Test Scenarios for Checking the Static Data Members

Consider the static qualified members of the class. The test scenarios are:

- +Check the list of month names in static constant monthNames.
- +Check the list of day names in static constant dayNames.
- +Check the value of the maximum valid year stored in static constant MAX_VALID_YR.
- +Check the value of the minimum valid year stored in static constant MIN_VALID_YR.

Test Scenarios for Checking the Static Member Functions

Consider the static qualified member funcitons of the class. The test scenarios are:

- Check the isLeap utility function.
  1. +Check if it returns true upon a leap year input.
  2. - Check if it returns false upon non-leap year input.
- Check the validate utility function.
  1. +Check if it returns true upon entering a valid date example 29/02/2020
  2. - Check if it returns false upon entering a invalid date example 29/02/2021 or 31/04/2020.

Test Scenarios for Construction of Objects

Consider the static createDate(unsigned int d, unsigned int m, unsigned int y) constructor. The test scenario is:

- +The data members are correctly initialized. To check if the data members date_ of type unsigned int, month_ of enum type Month and year_ of type unsigned int.
- - Check if exception is thrown upon trying to create an invalid Date.

Test Scenarios for Checking the print() member function

Consider the print() member function. The test scenario is:

- +Check if the function prints in dd/MMM/yy format. Example: 2/Mar/2021

Test Scenarios for Checking the validDate() member function

Consider the validDate() member function. The test scenario is:

- +Check if the function returns true upon inputting a valid date.
- - Check if it returns false upon invalid date input. Example: 29/Feb/2021

Test Scenarios for Checking the day() member function

Consider the day() member function. The test scenario is:

- +Check if the function returns the correct day corresponding to the date. Example: “Tuesday” for 2/Mar/2021.

Test Scenarios for equality operator

Consider the overloaded equality operator friend bool operator==(const Date&) const. The test scenarios are :

- +Check if the operator returns true if both dates are same.
- - Check if it returns false if they are different.

Test Scenarios for inequality operator

Consider the overloaded inequality operator bool operator!=(const Date&) const. The test scenarios are :

- +Check if the operator returns false if both dates are same.
- - Check if it returns true if they are different.

Test Scenarios for DiffOfYears Function

Consider the unsigned int DiffOfYears(const Date&) const function. The test scenarios are :

- +Check if the function returns the difference of years between two dates correctly.

Test Scenarios for CompareDate Function

Consider the unsigned int CompareDate(const Date&) const function. The test scenarios are :

- +Check if the function returns -1 when this pointer object’s date behind the date passed as argument.
- +Check if the function returns 0 when this pointer object’s date equal to the date passed as argument.
- +Check if the function returns 1 when this pointer object’s date ahead of the date passed as argument.

Test Scenarios for isWithinOnYear Function

Consider the unsigned int isWithinOneYear(const Date&) const function. The test scenarios are :

- +Check if the function returns true when this pointer object’s date is within one year from the date passed as argument.
- - Check if the function returns false when this pointer object’s date is not within one year from the date passed as argument.

Test Scenarios for isWithinOnDay Function

Consider the unsigned int isWithinOneDay(const Date&) const function. The test scenarios are :

- +Check if the function returns true when this pointer object’s date is within one day from the date passed as argument.
- - Check if the function returns false when this pointer object’s date is not within one day from the date passed as argument.

Test Scenarios for insertion operator

Consider the overloaded insertion operator std::ostream& operator<<(std::ostream&, const Date&). The test scenario is:

- +Check if the operator inserts the date to the console in dd/MMM/yy format. Example: 2/Mar/2021

Unit Testing Station Class

Test Scenarios for Construction of Objects

Consider the createStation(std::string) constructor. The test scenario is:

- +The data members are correctly initialized. To check if the data member name_ of type std::string is correctly initialized.
- - Check if exception is thrown upon trying to create an invalid Date.

Test Scenarios for Checking the GetName() member function

Consider the GetName() member function. The test scenario is:

- +Check if the function returns the name of the station correctly.

Test Scenarios for Checking the GetDistance() member function

Consider the GetDistance() member function. The test scenario is:

- +Check if the function returns the distance between itself and another station correctly according to the distance matrix in Railways Class.

Test Scenarios for equality operator

Consider the overloaded equality operator bool operator==(const Station&) const. The test scenarios are :

- +Check if the operator returns true if both stations are same.
- - Check if it returns false if they are different.

Test Scenarios for inequality operator

Consider the overloaded inequality operator bool operator!=(const Station&) const. The test scenarios are :

- +Check if the operator returns false if both stations are same.
- - Check if it returns true if they are different.

Test Scenarios for insertion operator

Consider the overloaded insertion operator friend std::ostream& operator<<(std::ostream&, const Station&). The test scenario is:

- +Check if the operator inserts the name of the Station to the console.

Unit Testing Railways Class

Test Scenarios for Checking the Static Data Members

Consider the static qualified members of the class. The test scenarios are:

- +Check the list of stations in static constant sStations.
- +Check the list of distance matrix in static constant sDistances.

Test Scenarios for Construction of the Singleton Object

Consider the static const Railways& IndianRailways() instance creator. The test scenario is:

- +Check whether multiple calls to the instance creator results in obtaining a singleton object.

Test Scenarios for Checking the GetDistance() member function

Consider the GetDistance() member function. The test scenario is:

- +Check if the function returns the distance between two stations correctly according to the distance matrix.

Test Scenarios for insertion operator

Consider the overloaded insertion operator friend std::ostream& operator<<(std::ostream&, const Railways&). The test scenario is:

- +Check if the operator inserts the all the information of the Railways to the console.

Unit Testing DivyaangTypes Class

Test Scenarios for Checking the Static Data Members

Consider the static qualified members of the class. The test scenarios are:

- +Check the value stored in the sName.
- +Check the values stored in Concession Factor Matrix sConcessionFactor.

Test Scenarios for Construction of the Singleton Object

Consider the static DivyaangTypes<T>& Type() instance creator. The test scenario is:

- +Check whether multiple calls to the instance creator results in obtaining a singleton object.

Test Scenarios for Checking the GetName() member function

Consider the GetName() member function. The test scenario is:

- +Check if the function returns the correct name of the Disability as was stored in sName.

Test Scenarios for Checking the GetConcession() member function

Consider the GetConcession(const BookingClass&) member function. The test scenario is:

- +Check if the function returns the correct Concession Factor corresponding to the BookingClass and as per the data stored in Concession Factor Matrix sConcessionFactor.

Unit Testing Concessions Class

Test Scenarios for Checking the GetConcessionFactor() member function

Consider the GetConcessionFactor() member function. The test scenario is:

- +Check if the function returns the correct Concession Factor i.e. 0.

Unit Testing General_Concession Class

Test Scenarios for Checking the GetConcessionFactor() member function

Consider the GetConcessionFactor() member function. The test scenario is:

- +Check if the function returns the correct Concession Factor i.e. 0.

Unit Testing Ladies_Concession Class

Test Scenarios for Checking the GetConcessionFactor() member function

Consider the GetConcessionFactor(const Passenger&) member function. The test scenario is:

- +Check if the function returns the correct Concession Factor i.e. 0 (for now, later may be extended based on the Passenger).

Unit Testing SeniorCitizen_Concession Class

Test Scenarios for Checking the GetConcessionFactor() member function

Consider the GetConcessionFactor(const Passenger&) member function. The test scenario is:

- +Check if the function returns the correct Concession Factor i.e. 0.4 if the Passenger is male and above 60.
- +Check if the function returns the correct Concession Factor i.e. 0.5 if the Passenger is female and above 58.
- - Check if the function returns the correct Concession Factor i.e. 0 if the Passenger does not satisfy the above criteria.

Unit Testing Divyaang_Concession Class

Test Scenarios for Checking the GetConcessionFactor() member function

Consider the GetConcessionFactor(const BookingClass&, const Passenger&) member function. The test scenario is:

- +Check if the function returns the correct Concession Factor if the Passenger has a disability and for a valid BookingClass based on the Disability Concession Factor matrix stored as a map in the Divyaang class.
- - Check if the function returns the correct Concession Factor i.e. 0 if the Passenger does not satisfy the above criteria (i.e. does not have a disability).

Unit Testing Passenger Class

Test Scenarios for Construction of the Object

Consider the static createPassenger(std::string firstName_val, std::string middleName_val, std::string lastName_val, unsigned int date_val, unsigned int month_val, unsigned int year_val, const Gender& gender_val, std::string aadhaarNumber_val,std::string mobileNumber_val,const Divyaang* const disabilityType_val, const std::string disabilityID_val) object creator. The test scenario is:

- +Check whether all the data members of the object are correctly initialized upon creation of a valid Passenger.
- - Check if exception is thrown upon trying to create an invalid Passenger.

Test Scenarios for Checking the validateAadhaar() static member function

Consider the validateAadhaar(const std::string&) static member function. The test scenario is:

- +Check if the function returns the true upon giving a valid Aadhaar Number.
- - Check if the function returns the false upon giving a invalid Aadhaar Number.
  - Less than 12 digits.
  - More than 12 digits.
  - Usage of symbols other than decimal digits.

Test Scenarios for Checking the validateMobile() static member function

Consider the validateMobile(const std::string&) static member function. The test scenario is:

- +Check if the function returns the true upon giving a valid Mobile Number.
- - Check if the function returns the false upon giving a invalid Mobile Number.
  - Less than 10 digits.
  - More than 10 digits.
  - Usage of symbols other than decimal digits.

Test Scenarios for Checking the GetName() member function

Consider the GetName() member function. The test scenario is:

- +Check if the function returns the name of the Passenger correctly.

Test Scenarios for Checking the GetDOB() member function

Consider the GetDOB() member function. The test scenario is:

- +Check if the function returns the Date Of Birth of the Passenger correctly.

Test Scenarios for Checking the GetAge() member function

Consider the GetAge() member function. The test scenario is:

- +Check if the function returns the age of the Passenger correctly.

Test Scenarios for Checking the GetGender() member function

Consider the GetGender() member function. The test scenario is:

- +Check if the function returns the gender of the Passenger correctly.

Test Scenarios for Checking the GetAadhaar() member function

Consider the GetAadhaar() member function. The test scenario is:

- +Check if the function returns the aadhaar Number of the Passenger correctly.

Test Scenarios for Checking the GetMobile() member function

Consider the GetMobile() member function. The test scenario is:

- +Check if the function returns the mobile Number of the Passenger correctly.

Test Scenarios for Checking the GetDisability() member function

Consider the GetDisability() member function. The test scenario is:

- +Check if the function returns the disability of the Passenger correctly.

Test Scenarios for Checking the GetDisabilityID() member function

Consider the GetDisabilityID() member function. The test scenario is:

- +Check if the function returns the disability ID of the Passenger correctly.

Test Scenarios for insertion operator

Consider the overloaded insertion operator friend std::ostream& operator<<(std::ostream&, const Passeger&). The test scenario is:

- +Check if the operator inserts the all the information of the Passenger to the console.

Unit Testing BookingCategoryTypes Class

Test Scenarios for Checking the Static Data Members

Consider the static qualified members of the class. The test scenarios are:

- +Check the value stored in the sName.

Test Scenarios for Construction of the Singleton Object

Consider the static BookingCategoryTypes<T>& Type() instance creator. The test scenario is:

- +Check whether multiple calls to the instance creator results in obtaining a singleton object.

Test Scenarios for Checking the GetName() member function

Consider the GetName() member function. The test scenario is:

- +Check if the function returns the correct name of the BookingCategory as was stored in sName.

Test Scenarios for Checking the isEligible() member function

Consider the isEligible(const Passenger&) member function. The test scenario is:

- +Check if the function returns the true upon checking eligibility of the Passenger for the Booking Category.
- - Check if the function returns the false if the Passenger is ineligible to opt for that Booking Category. Example: A person not having Disability is ineligible for the Divyaang Booking Category.

Unit Testing BookingClassTypes Class

Test Scenarios for Checking the Static Data Members

Consider the static qualified members of the class. The test scenarios are:

- +Check the value stored in the sloadFactor, sName, sIsSitting, sIsAC, sNumberOfTiers, sIsLuxury, sReservationCharge, sMinimumTatkalCharges, sMaximumTatkalCharges, sMinimumTatkalDistance and sTatkalLoadFactor.

Test Scenarios for Construction of the Singleton Object

Consider the static BookingClassTypes<T>& Type() instance creator. The test scenario is:

- +Check whether multiple calls to the instance creator results in obtaining a singleton object.

Test Scenarios for Checking the IsSitting() member function

Consider the IsSitting() member function. The test scenario is:

- +Check if the function returns the correct boolean value as per data.

Test Scenarios for Checking the GetNumberOfTiers() member function

Consider the GetNumberOfTiers() member function. The test scenario is:

- +Check if the function returns the correct value as per data.

Test Scenarios for Checking the GetLoadFactor() member function

Consider the GetLoadFactor() member function. The test scenario is:

- +Check if the function returns the same value as sloadFactor.

Test Scenarios for Checking the GetName() member function

Consider the GetName() member function. The test scenario is:

- +Check if the function returns the correct value as per data.

Test Scenarios for Checking the IsAC() member function

Consider the IsAC() member function. The test scenario is:

- +Check if the function returns the correct boolean value as per data.

Test Scenarios for Checking the IsLuxury() member function

Consider the IsSitting() member function. The test scenario is:

- +Check if the function returns the correct value as per data.

Test Scenarios for Checking the GetReservationCharge() member function

Consider the GetReservationCharge() member function. The test scenario is:

- +Check if the function returns the same value as sReservationCharge.

Test Scenarios for Checking the GetMinimumTatkalCharges() member function

Consider the GetMinimumTatkalCharges() member function. The test scenario is:

- +Check if the function returns the same value as sMinimumTatkalCharges.

Test Scenarios for Checking the GetMaximumTatkalCharges() member function

Consider the GetMaximumTatkalCharges() member function. The test scenario is:

- +Check if the function returns the same value as sMaximumTatkalCharges.

Test Scenarios for Checking the GetMinimumTatkalDistance() member function

Consider the GetMinimumTatkalDistance() member function. The test scenario is:

- +Check if the function returns the same value as sMinimumTatkalDistance.

Test Scenarios for Checking the GetTatkalLoadFactor() member function

Consider the GetTatkalLoadFactor() member function. The test scenario is:

- +Check if the function returns the same value as sTatkalLoadFactor.

Unit Testing BookingTypes Class

Test Scenarios for Checking the Static Data Members

Consider the static qualified members of the class. The test scenarios are:

- +Check the value of the Bare Fare charged per Kilometer stored in static constant sBarePerKM.

Test Scenarios for Construction of Objects

Consider the Booking(Station fromStation_val, Station toStation_val, Date dateOfBooking_val, const BookingClass& bookingClass_val, const BookingCategory& bookingCategory_val, const Passenger& passenger_val) constructor. The test scenario is:

- +The data members are correctly initialized and fare is being correctly computed for a valid passenger.
- - Check if exception is thrown upon trying to create an invalid Booking.

Test Scenarios for Checking the ComputeFare() member function

Consider the ComputeFare() member function. The test scenario is:

- +Check if the function computes the fare correctly.

Test Scenarios for equality operator

Consider the overloaded equality operator bool operator==(const Booking&) const. The test scenarios are :

- +Check if the operator returns true if both bookings have the same PNR are same.
- - Check if it returns false if they have different PNRs.

Test Scenarios for inequality operator

Consider the overloaded inequality operator bool operator!=(const Booking&) const. The test scenarios are :

- +Check if the operator returns false if both bookings have the same PNR are same.
- - Check if it returns true if they have different PNRs.

Test Scenarios for insertion operator

Consider the overloaded insertion operator friend std::ostream& operator<<(std::ostream&, const Booking&). The test scenario is:

- +Check if the operator inserts all the information of the booking to the console.

Application Tests

Application Test :

Test Application :

This is the Test Application which is written by me. It has different cases of Passenger object creation when the data is valid and cases of exception handling when data of the Passenger is invalid or inconsistent. It has different cases of Booking object creation when the data is valid and cases of exception handling when data of the Booking is invalid or inconsistent.

Golden Test Output :

**********APPLICATION TEST**********

########### VALID PASSENGER CREATION #############

*****Adults*****

P1

Name : Mike Xavier

Date Of Birth : 21/Jan/2001

Gender : Male

Aadhar Number : 987654321012

Mobile Number : 1234567890

P2

Name : Alexis Carlos Xavier

Date Of Birth : 21/Jan/2000

Gender : Female

Aadhar Number : 987653321012

*****Senior Citizen*****

P3

Name : Carlos Xavier

Date Of Birth : 21/Jan/1945

Gender : Male

Aadhar Number : 987655321012

Disability Type : TB Patient

Disability ID : 123

P4

Name : Petricia Xavier

Date Of Birth : 18/Sep/1940

Gender : Female

Aadhar Number : 987656321012

*****Children*****

P5

Name : Michael Xavier

Date Of Birth : 2/Dec/2015

Gender : Male

Aadhar Number : 987657321012

P6

Name : Michelle Rodriguez Xavier

Date Of Birth : 22/Apr/2016

Gender : Female

Aadhar Number : 987657321012

Mobile Number : 4356789011

Disability Type : Blind

Disability ID : 345

##

########### INVALID PASSENGER CREATION (Exception Handling) #############

*****Create a Passenger who last both first name and last name missing*****

Exception caught successfully upon trying to create an invalid Passenger who last both first name and last name missing

Exception message: Name has inconsistent details. Please check the details again.

*****Create a Passenger who has dateOfBirth in the future*****

Exception caught successfully upon trying to create an invalid Passenger who has dateOfBirth in the future

Exception message: Passenger has invalid or inconsistent details. Please check the details again.

*****Create a Passenger who has invalid aadhaar Number*****

Exception caught successfully upon trying to create an invalid Passenger who has invalid aadhaar Number

Exception message: Passenger has invalid or inconsistent details. Please check the details again.

*****Create a Passenger who has invalid Mobile Number*****

Exception caught successfully upon trying to create an invalid Passenger who has invalid Mobile Number

Exception message: Passenger has invalid or inconsistent details. Please check the details again.

##

########### VALID BOOKING CREATION (using Valid Passengers) #############

*****General*****

B1

Passenger Details :

Name : Mike Xavier

Date Of Birth : 21/Jan/2001

Gender : Male

Aadhar Number : 987654321012

Mobile Number : 1234567890

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 1

From Station : Delhi

To Station : Mumbai

Travel Date : 9/Dec/2021

Travel Class : AC 3 Tier

Travel Category : General

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 3

--> Luxury : No

Fare : 1849

B2

Passenger Details :

Name : Alexis Carlos Xavier

Date Of Birth : 21/Jan/2000

Gender : Female

Aadhar Number : 987653321012

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 2

From Station : Delhi

To Station : Mumbai

Travel Date : 27/Dec/2021

Travel Class : AC First Class

Travel Category : General

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 2

--> Luxury : Yes

Fare : 4763

*****Senior Citizen*****

B3

Passenger Details :

Name : Carlos Xavier

Date Of Birth : 21/Jan/1945

Gender : Male

Aadhar Number : 987655321012

Disability Type : TB Patient

Disability ID : 123

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 3

From Station : Delhi

To Station : Mumbai

Travel Date : 28/Nov/2021

Travel Class : AC 3 Tier

Travel Category : Senior Citizen

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 3

--> Luxury : No

Fare : 1125

B4

Passenger Details :

Name : Petricia Xavier

Date Of Birth : 18/Sep/1940

Gender : Female

Aadhar Number : 987656321012

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 4

From Station : Delhi

To Station : Mumbai

Travel Date : 9/Oct/2021

Travel Class : AC First Class

Travel Category : Senior Citizen

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 2

--> Luxury : Yes

Fare : 2411

*****Divyaang*****

B5

Passenger Details :

Name : Michelle Rodriguez Xavier

Date Of Birth : 22/Apr/2016

Gender : Female

Aadhar Number : 987657321012

Mobile Number : 4356789011

Disability Type : Blind

Disability ID : 345

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 5

From Station : Delhi

To Station : Mumbai

Travel Date : 28/Nov/2021

Travel Class : AC 3 Tier

Travel Category : Divyaang

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 3

--> Luxury : No

Fare : 492

B6

Passenger Details :

Name : Carlos Xavier

Date Of Birth : 21/Jan/1945

Gender : Male

Aadhar Number : 987655321012

Disability Type : TB Patient

Disability ID : 123

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 6

From Station : Delhi

To Station : Mumbai

Travel Date : 9/Oct/2021

Travel Class : AC First Class

Travel Category : Divyaang

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 2

--> Luxury : Yes

Fare : 4763

*****Tatkal*****

B7

Passenger Details :

Name : Petricia Xavier

Date Of Birth : 18/Sep/1940

Gender : Female

Aadhar Number : 987656321012

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 7

From Station : Delhi

To Station : Mumbai

Travel Date : 8/Apr/2021

Travel Class : AC 3 Tier

Travel Category : Tatkal

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 3

--> Luxury : No

Fare : 2249

B8

Passenger Details :

Name : Michael Xavier

Date Of Birth : 2/Dec/2015

Gender : Male

Aadhar Number : 987657321012

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 8

From Station : Chennai

To Station : Bangalore

Travel Date : 8/Apr/2021

Travel Class : AC First Class

Travel Category : Tatkal

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 2

--> Luxury : Yes

Fare : 1198

*****PremiumTatkal*****

B9

Passenger Details :

Name : Petricia Xavier

Date Of Birth : 18/Sep/1940

Gender : Female

Aadhar Number : 987656321012

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 9

From Station : Delhi

To Station : Mumbai

Travel Date : 8/Apr/2021

Travel Class : AC 3 Tier

Travel Category : Premium Tatkal

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 3

--> Luxury : No

Fare : 2649

B10

Passenger Details :

Name : Michael Xavier

Date Of Birth : 2/Dec/2015

Gender : Male

Aadhar Number : 987657321012

Booking Details :

BOOKING SUCCEEDED:

PNR Number : 10

From Station : Chennai

To Station : Bangalore

Travel Date : 8/Apr/2021

Travel Class : AC First Class

Travel Category : Premium Tatkal

Reservation Date : 8/Apr/2021

--> Mode : Sleeping

--> Comfort : AC

--> Bunks : 2

--> Luxury : Yes

Fare : 1198

##

########### INVALID BOOKING CREATION (Exception Handling) #############

*****Create a Booking with past date*****

Exception caught successfully upon trying to create an invalid Booking with past date

Exception message: Booking has invalid or inconsistent details. Please check the details again.

*****Create a Booking with fromStation same as toStation*****

Exception caught successfully upon trying to create an invalid Booking with fromStation same as toStation

Exception message: Booking has invalid or inconsistent details. Please check the details again.

*****Create a Booking with date for more than a year*****

Exception caught successfully upon trying to create an invalid Booking with date for more than a year

Exception message: Booking has invalid or inconsistent details. Please check the details again.

*****Create a Booking with Passenger Ineligible for the Booking Category*****

Exception caught successfully upon trying to create an invalid Booking with Passenger Ineligible for the Booking Category

Exception message: Booking has invalid or inconsistent details. Please check the details again.

##