

Gestão de Alojamentos Turísticos (Testes)

Generated by Doxygen 1.10.0

1 Namespace Index	1
1.1 Namespace List	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 SmartStay Namespace Reference	7
4.2 SmartStay.Tests Namespace Reference	7
5 Data Structure Documentation	9
5.1 SmartStay.Tests.ClientTests Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Member Function Documentation	10
5.1.2.1 Client_FullData_CreatesClient()	10
5.1.2.2 Client_GenerateUniqueClientId_CreatesUniqueIds()	10
5.1.2.3 Client_InvalidAddress_ThrowsValidationException()	10
5.1.2.4 Client_InvalidEmail_ThrowsValidationException()	10
5.1.2.5 Client_InvalidPaymentMethod_ThrowsValidationException()	10
5.1.2.6 Client_InvalidPhoneNumber_ThrowsValidationException()	11
5.1.2.7 Client_SetValidAddress_UpdatesAddress()	11
5.1.2.8 Client_SetValidEmail_UpdatesEmail()	11
5.1.2.9 Client_SetValidFirstName_UpdatesFirstName()	11
5.1.2.10 Client_SetValidLastName_UpdatesLastName()	11
5.1.2.11 Client_SetValidPaymentMethod_UpdatesPaymentMethod()	11
5.1.2.12 Client_SetValidPhoneNumber_UpdatesPhoneNumber()	12
5.1.2.13 Client_ToString_ReturnsValidJson()	12
5.1.2.14 Client_ValidData_CreatesClient()	12
5.2 SmartStay.Tests.ValidatorTests Class Reference	12
5.2.1 Detailed Description	14
5.2.2 Member Function Documentation	14
5.2.2.1 ValidateAccommodationName_InvalidName_ThrowsException()	14
5.2.2.2 ValidateAccommodationName_ValidName_ReturnsName()	14
5.2.2.3 ValidateAccommodationType_InvalidType_ThrowsException()	15
5.2.2.4 ValidateAccommodationType_ValidType_ReturnsType()	15
5.2.2.5 ValidateAddress_InvalidAddress_ThrowsException()	15
5.2.2.6 ValidateAddress_ValidAddress_ReturnsAddress()	15
5.2.2.7 ValidateCheckInDate_PastDate_ThrowsException()	16
5.2.2.8 ValidateCheckInDate_ValidFutureDate_ReturnsDate()	16
5.2.2.9 ValidateCheckOutDate_InvalidDateRange_ThrowsException()	16
5.2.2.10 ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()	16

5.2.2.11 ValidateEmail_InvalidEmail_ThrowsException()	16
5.2.2.12 ValidateEmail_ValidEmail_ReturnsEmail()	17
5.2.2.13 ValidateName_InvalidName_ThrowsException()	17
5.2.2.14 ValidateName_ValidName_ReturnsName()	17
5.2.2.15 ValidatePayment_InvalidPayment_ThrowsException()	17
5.2.2.16 ValidatePayment_ValidPayment_ReturnsPayment()	18
5.2.2.17 ValidatePaymentAmount_InvalidAmount_ThrowsException()	18
5.2.2.18 ValidatePaymentAmount_ValidAmount_ReturnsAmount()	18
5.2.2.19 ValidatePaymentStatus_InvalidStatus_ThrowsException()	18
5.2.2.20 ValidatePaymentStatus_ValidStatus_ReturnsStatus()	18
5.2.2.21 ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException()	18
5.2.2.22 ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber()	19
5.2.2.23 ValidatePrice_InvalidPrice_ThrowsException()	19
5.2.2.24 ValidatePrice_ValidPrice_ReturnsPrice()	19
5.2.2.25 ValidateReservationStatus_InvalidStatus_ThrowsException()	19
5.2.2.26 ValidateReservationStatus_ValidStatus_ReturnsStatus()	20
5.2.2.27 ValidateTotalCost_InvalidCost_ThrowsException()	20
5.2.2.28 ValidateTotalCost_ValidCost_ReturnsTotalCost()	20
6 File Documentation	21
6.1 AccommodationTests.cs File Reference	21
6.2 AccommodationTests.cs	21
6.3 ClientTests.cs File Reference	21
6.4 ClientTests.cs	21
6.5 ReservationTests.cs File Reference	24
6.6 ReservationTests.cs	24
6.7 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs File Reference	24
6.8 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs	24
6.9 SmartStay.Tests.AssemblyInfo.cs File Reference	24
6.10 SmartStay.Tests.AssemblyInfo.cs	24
6.11 SmartStay.Tests.GlobalUsings.g.cs File Reference	25
6.12 SmartStay.Tests.GlobalUsings.g.cs	25
6.13 ValidationTests.cs File Reference	25
6.14 ValidationTests.cs	25
Index	31

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

SmartStay	7
SmartStay.Tests	7

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

SmartStay.Tests.ClientTests	9
SmartStay.Tests.ValidatorTests Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more	12

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

AccommodationTests.cs	21
ClientTests.cs	21
ReservationTests.cs	24
.NETCoreApp, Version=v8.0.AssemblyAttributes.cs	24
SmartStay.Tests.AssemblyInfo.cs	24
SmartStay.Tests.GlobalUsings.g.cs	25
ValidationTests.cs	25

Chapter 4

Namespace Documentation

4.1 SmartStay Namespace Reference

Namespaces

- namespace [Tests](#)

4.2 SmartStay.Tests Namespace Reference

Data Structures

- class [ClientTests](#)
- class [ValidatorTests](#)

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Chapter 5

Data Structure Documentation

5.1 SmartStay.Tests.ClientTests Class Reference

Public Member Functions

- void [Client_ValidData_CreatesClient \(\)](#)
Tests the constructor of the Client class when valid data is provided. Ensures the client is created successfully with the given information.
- void [Client_FullData_CreatesClient \(\)](#)
Tests the constructor of the Client class when all optional parameters are provided. Ensures the client is created successfully with full details.
- void [Client_InvalidEmail_ThrowsValidationException \(\)](#)
Tests the constructor of the Client class when invalid email is provided. Ensures a ValidationException is thrown.
- void [Client_InvalidPhoneNumber_ThrowsValidationException \(\)](#)
Tests the constructor of the Client class when invalid phone number is provided. Ensures a ValidationException is thrown.
- void [Client_InvalidAddress_ThrowsValidationException \(\)](#)
Tests the constructor of the Client class when invalid address is provided. Ensures a ValidationException is thrown.
- void [Client_InvalidPaymentMethod_ThrowsValidationException \(\)](#)
Tests the constructor of the Client class when an invalid payment method is provided. Ensures a ValidationException is thrown.
- void [Client_SetValidFirstName_UpdatesFirstName \(\)](#)
Tests the property setter and getter for FirstName. Ensures that a valid name can be set and retrieved correctly.
- void [Client_SetValidLastName_UpdatesLastName \(\)](#)
Tests the property setter and getter for LastName. Ensures that a valid last name can be set and retrieved correctly.
- void [Client_SetValidEmail_UpdatesEmail \(\)](#)
Tests the property setter and getter for Email. Ensures that a valid email can be set and retrieved correctly.
- void [Client_SetValidPhoneNumber_UpdatesPhoneNumber \(\)](#)
Tests the property setter and getter for PhoneNumber. Ensures that a valid phone number can be set and retrieved correctly.
- void [Client_SetValidAddress_UpdatesAddress \(\)](#)
Tests the property setter and getter for Address. Ensures that a valid address can be set and retrieved correctly.
- void [Client_SetValidPaymentMethod_UpdatesPaymentMethod \(\)](#)
Tests the property setter and getter for PreferredPaymentMethod. Ensures that a valid payment method can be set and retrieved correctly.
- void [Client_GenerateUniqueClientId_CreatesUniqueIds \(\)](#)
Tests the client ID generation to ensure it increments properly for multiple clients. Ensures that each client has a unique ID.
- void [Client_ToString_ReturnsValidJson \(\)](#)
Tests the ToString method of the Client class. Ensures the client object is serialized to a JSON string with proper formatting.

5.1.1 Detailed Description

Definition at line 11 of file [ClientTests.cs](#).

5.1.2 Member Function Documentation

5.1.2.1 Client_FullDataCreatesClient()

```
void SmartStay.Tests.ClientTests.Client_FullDataCreatesClient () [inline]
```

Tests the constructor of the Client class when all optional parameters are provided. Ensures the client is created successfully with full details.

Definition at line 36 of file [ClientTests.cs](#).

5.1.2.2 Client_GenerateUniqueIdCreatesUniqueIds()

```
void SmartStay.Tests.ClientTests.Client_GenerateUniqueIdCreatesUniqueIds () [inline]
```

Tests the client ID generation to ensure it increments properly for multiple clients. Ensures that each client has a unique ID.

Definition at line 246 of file [ClientTests.cs](#).

5.1.2.3 Client_InvalidAddress.ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidAddress.ThrowsValidationException () [inline]
```

Tests the constructor of the Client class when invalid address is provided. Ensures a ValidationException is thrown.

Definition at line 94 of file [ClientTests.cs](#).

5.1.2.4 Client_InvalidEmail.ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidEmail.ThrowsValidationException () [inline]
```

Tests the constructor of the Client class when invalid email is provided. Ensures a ValidationException is thrown.

Definition at line 61 of file [ClientTests.cs](#).

5.1.2.5 Client_InvalidPaymentMethod.ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidPaymentMethod.ThrowsValidationException ()  
[inline]
```

Tests the constructor of the Client class when an invalid payment method is provided. Ensures a ValidationException is thrown.

Definition at line 112 of file [ClientTests.cs](#).

5.1.2.6 Client_InvalidPhoneNumber_ThrowsValidationException()

```
void SmartStay.Tests.ClientTests.Client_InvalidPhoneNumber_ThrowsValidationException () [inline]
```

Tests the constructor of the Client class when invalid phone number is provided. Ensures a ValidationException is thrown.

Definition at line 77 of file [ClientTests.cs](#).

5.1.2.7 Client_SetValidAddress_UpdatesAddress()

```
void SmartStay.Tests.ClientTests.Client_SetValidAddress_UpdatesAddress () [inline]
```

Tests the property setter and getter for Address. Ensures that a valid address can be set and retrieved correctly.

Definition at line 204 of file [ClientTests.cs](#).

5.1.2.8 Client_SetValidEmail_UpdatesEmail()

```
void SmartStay.Tests.ClientTests.Client_SetValidEmail_UpdatesEmail () [inline]
```

Tests the property setter and getter for Email. Ensures that a valid email can be set and retrieved correctly.

Definition at line 167 of file [ClientTests.cs](#).

5.1.2.9 Client_SetValidFirstName_UpdatesFirstName()

```
void SmartStay.Tests.ClientTests.Client_SetValidFirstName_UpdatesFirstName () [inline]
```

Tests the property setter and getter for FirstName. Ensures that a valid name can be set and retrieved correctly.

Definition at line 131 of file [ClientTests.cs](#).

5.1.2.10 Client_SetValidLastName_UpdatesLastName()

```
void SmartStay.Tests.ClientTests.Client_SetValidLastName_UpdatesLastName () [inline]
```

Tests the property setter and getter for LastName. Ensures that a valid last name can be set and retrieved correctly.

Definition at line 149 of file [ClientTests.cs](#).

5.1.2.11 Client_SetValidPaymentMethod_UpdatesPaymentMethod()

```
void SmartStay.Tests.ClientTests.Client_SetValidPaymentMethod_UpdatesPaymentMethod () [inline]
```

Tests the property setter and getter for PreferredPaymentMethod. Ensures that a valid payment method can be set and retrieved correctly.

Definition at line 224 of file [ClientTests.cs](#).

5.1.2.12 Client_SetValidPhoneNumber_UpdatesPhoneNumber()

```
void SmartStay.Tests.ClientTests.Client_SetValidPhoneNumber_UpdatesPhoneNumber () [inline]
```

Tests the property setter and getter for PhoneNumber. Ensures that a valid phone number can be set and retrieved correctly.

Definition at line 185 of file [ClientTests.cs](#).

5.1.2.13 Client_ToString_ReturnsValidJson()

```
void SmartStay.Tests.ClientTests.Client_ToString_ReturnsValidJson () [inline]
```

Tests the ToString method of the Client class. Ensures the client object is serialized to a JSON string with proper formatting.

Definition at line 259 of file [ClientTests.cs](#).

5.1.2.14 Client_ValidData_CreatesClient()

```
void SmartStay.Tests.ClientTests.Client_ValidData_CreatesClient () [inline]
```

Tests the constructor of the Client class when valid data is provided. Ensures the client is created successfully with the given information.

Definition at line 18 of file [ClientTests.cs](#).

The documentation for this class was generated from the following file:

- [ClientTests.cs](#)

5.2 SmartStay.Tests.ValidatorTests Class Reference

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Public Member Functions

- void [ValidateName_ValidName_ReturnsName](#) (string validName)
The ValidateName_ValidName_ReturnsName Tests the ValidateName method with a valid name input and ensures it returns the same name.
- void [ValidateName_InvalidName_ThrowsException](#) (string invalidName)
The ValidateName_InvalidName_ThrowsException Tests the ValidateName method with invalid name inputs (empty or null) and ensures a ValidationException is thrown.
- void [ValidateAccommodationName_ValidName_ReturnsName](#) (string validAccommodationName)
The ValidateAccommodationName_ValidName_ReturnsName Tests the ValidateAccommodationName method with a valid accommodation name and ensures it returns the same name.
- void [ValidateAccommodationName_InvalidName_ThrowsException](#) (string invalidAccommodationName)
The ValidateAccommodationName_InvalidName_ThrowsException Tests the ValidateAccommodationName method with invalid accommodation names (empty or null) and ensures a ValidationException is thrown.
- void [ValidateEmail_ValidEmail_ReturnsEmail](#) (string validEmail)
The ValidateEmail_ValidEmail_ReturnsEmail Tests the ValidateEmail method with a valid email input and ensures it returns the same email.
- void [ValidateEmail_InvalidEmail_ThrowsException](#) (string invalidEmail)
The ValidateEmail_InvalidEmail_ThrowsException Tests the ValidateEmail method with invalid email inputs (e.g., an email missing '@') and ensures a ValidationException is thrown.
- void [ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber](#) (string validPhoneNumber)
The ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber Tests the ValidatePhoneNumber method with a valid phone number input and ensures it returns the same phone number.
- void [ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException](#) (string invalidPhoneNumber)
The ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException Tests the ValidatePhoneNumber method with an invalid phone number input and ensures a ValidationException is thrown.
- void [ValidateAddress_ValidAddress_ReturnsAddress](#) (string validAddress)
The ValidateAddress_ValidAddress_ReturnsAddress Tests the ValidateAddress method with a valid address and ensures it returns the same address.
- void [ValidateAddress_InvalidAddress_ThrowsException](#) (string invalidAddress)
The ValidateAddress_InvalidAddress_ThrowsException Tests the ValidateAddress method with an invalid address (e.g., empty) and ensures a ValidationException is thrown.
- void [ValidatePrice_ValidPrice_ReturnsPrice](#) ()
The ValidatePrice_ValidPrice_ReturnsPrice Tests the ValidatePrice method with a valid price input and ensures it returns the same price.
- void [ValidatePrice_InvalidPrice_ThrowsException](#) ()
The ValidatePrice_InvalidPrice_ThrowsException Tests the ValidatePrice method with an invalid price input (e.g., zero or negative) and ensures a ValidationException is thrown.
- void [ValidatePaymentAmount_ValidAmount_ReturnsAmount](#) ()
The ValidatePaymentAmount_ValidAmount_ReturnsAmount Tests the ValidatePaymentAmount method with a valid payment amount input and ensures it returns the same amount.
- void [ValidatePaymentAmount_InvalidAmount_ThrowsException](#) ()
The ValidatePaymentAmount_InvalidAmount_ThrowsException Tests the ValidatePaymentAmount method with an invalid payment amount input (e.g., negative amount) and ensures a ValidationException is thrown.
- void [ValidatePaymentStatus_ValidStatus_ReturnsStatus](#) ()
Validates that the payment status is correctly processed when a valid status is provided.
- void [ValidatePaymentStatus_InvalidStatus_ThrowsException](#) ()
Validates that an exception is thrown when an invalid payment status (out of the predefined enum) is provided.
- void [ValidateAccommodationType_ValidType_ReturnsType](#) ()
Validates that the accommodation type is correctly processed when a valid type is provided.
- void [ValidateAccommodationType_InvalidType_ThrowsException](#) ()
Validates that an exception is thrown when an invalid accommodation type (out of the predefined enum) is provided.
- void [ValidateCheckInDate_ValidFutureDate_ReturnsDate](#) ()
Validates that the check-in date is processed correctly when a future date is provided.

- void [ValidateCheckInDate_PastDate_ThrowsException \(\)](#)
Validates that an exception is thrown when a past date is provided as the check-in date.
- void [ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate \(\)](#)
Validates that the check-out date is processed correctly when it is after the check-in date.
- void [ValidateCheckOutDate_InvalidDateRange_ThrowsException \(\)](#)
Validates that an exception is thrown when the check-out date is before the check-in date.
- void [ValidateTotalCost_ValidCost_ReturnsTotalCost \(\)](#)
Validates that the total cost is correctly processed when a valid cost is provided.
- void [ValidateTotalCost_InvalidCost_ThrowsException \(\)](#)
Validates that an exception is thrown when a negative cost is provided.
- void [ValidatePayment_ValidPayment_ReturnsPayment \(\)](#)
Validates that the payment amount is correctly processed when a valid payment is provided.
- void [ValidatePayment_InvalidPayment_ThrowsException \(\)](#)
Validates that an exception is thrown when a negative payment value is provided.
- void [ValidateReservationStatus_ValidStatus_ReturnsStatus \(\)](#)
Validates that the reservation status is correctly processed when a valid status is provided.
- void [ValidateReservationStatus_InvalidStatus_ThrowsException \(\)](#)
Validates that an exception is thrown when an invalid reservation status (out of the predefined enum) is provided.

5.2.1 Detailed Description

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Definition at line 32 of file [ValidationTests.cs](#).

5.2.2 Member Function Documentation

5.2.2.1 ValidateAccommodationName_InvalidName_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationName_InvalidName_ThrowsException (
    string invalidAccommodationName ) [inline]
```

The `ValidateAccommodationName_InvalidName_ThrowsException` Tests the `ValidateAccommodationName` method with invalid accommodation names (empty or null) and ensures a `ValidationException` is thrown.

Parameters

<code>invalidAccommodationName</code>	The invalidAccommodationNamestring
---------------------------------------	------------------------------------

Definition at line 90 of file [ValidationTests.cs](#).

5.2.2.2 ValidateAccommodationName_ValidName_ReturnsName()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationName_ValidName_ReturnsName (
    string validAccommodationName ) [inline]
```

The `ValidateAccommodationName_ValidName_ReturnsName` Tests the `ValidateAccommodationName` method with a valid accommodation name and ensures it returns the same name.

Parameters

<i>validAccommodationName</i>	The validAccommodationNamestring
-------------------------------	----------------------------------

Definition at line 75 of file [ValidationTests.cs](#).

5.2.2.3 ValidateAccommodationType_InvalidType_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationType_InvalidType_ThrowsException ( )  
[inline]
```

Validates that an exception is thrown when an invalid accommodation type (out of the predefined enum) is provided.

Definition at line 298 of file [ValidationTests.cs](#).

5.2.2.4 ValidateAccommodationType_ValidType_ReturnsType()

```
void SmartStay.Tests.ValidatorTests.ValidateAccommodationType_ValidType_ReturnsType ( ) [inline]
```

Validates that the accommodation type is correctly processed when a valid type is provided.

Definition at line 286 of file [ValidationTests.cs](#).

5.2.2.5 ValidateAddress_InvalidAddress_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateAddress_InvalidAddress_ThrowsException (   
    string invalidAddress ) [inline]
```

The ValidateAddress_InvalidAddress_ThrowsException Tests the ValidateAddress method with an invalid address (e.g., empty) and ensures a ValidationException is thrown.

Parameters

<i>invalidAddress</i>	The invalidAddressstring
-----------------------	--------------------------

Definition at line 186 of file [ValidationTests.cs](#).

5.2.2.6 ValidateAddress_ValidAddress_ReturnsAddress()

```
void SmartStay.Tests.ValidatorTests.ValidateAddress_ValidAddress_ReturnsAddress (   
    string validAddress ) [inline]
```

The ValidateAddress_ValidAddress_ReturnsAddress Tests the ValidateAddress method with a valid address and ensures it returns the same address.

Parameters

<i>validAddress</i>	The validAddressstring
---------------------	------------------------

Definition at line 172 of file [ValidationTests.cs](#).

5.2.2.7 ValidateCheckInDate_PastDate_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckInDate_PastDate_ThrowsException () [inline]
```

Validates that an exception is thrown when a past date is provided as the check-in date.

Definition at line 325 of file [ValidationTests.cs](#).

5.2.2.8 ValidateCheckInDate_ValidFutureDate_ReturnsDate()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckInDate_ValidFutureDate_ReturnsDate () [inline]
```

Validates that the check-in date is processed correctly when a future date is provided.

Definition at line 314 of file [ValidationTests.cs](#).

5.2.2.9 ValidateCheckOutDate_InvalidDateRange_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckOutDate_InvalidDateRange_ThrowsException () [inline]
```

Validates that an exception is thrown when the check-out date is before the check-in date.

Definition at line 352 of file [ValidationTests.cs](#).

5.2.2.10 ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()

```
void SmartStay.Tests.ValidatorTests.ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate () [inline]
```

Validates that the check-out date is processed correctly when it is after the check-in date.

Definition at line 340 of file [ValidationTests.cs](#).

5.2.2.11 ValidateEmail_InvalidEmail_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateEmail_InvalidEmail_ThrowsException (
    string invalidEmail ) [inline]
```

The ValidateEmail_InvalidEmail_ThrowsException Tests the ValidateEmail method with invalid email inputs (e.g., an email missing '@') and ensures a ValidationException is thrown.

Parameters

<i>invalidEmail</i>	The invalidEmailstring
---------------------	------------------------

Definition at line 122 of file [ValidationTests.cs](#).

5.2.2.12 ValidateEmail_ValidEmail_ReturnsEmail()

```
void SmartStay.Tests.ValidatorTests.ValidateEmail_ValidEmail_ReturnsEmail (
    string validEmail ) [inline]
```

The ValidateEmail_ValidEmail_ReturnsEmail Tests the ValidateEmail method with a valid email input and ensures it returns the same email.

Parameters

<i>validEmail</i>	The validEmailstring
-------------------	----------------------

Definition at line 108 of file [ValidationTests.cs](#).

5.2.2.13 ValidateName_InvalidName_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateName_InvalidName_ThrowsException (
    string invalidName ) [inline]
```

The ValidateName_InvalidName_ThrowsException Tests the ValidateName method with invalid name inputs (empty or null) and ensures a ValidationException is thrown.

Parameters

<i>invalidName</i>	The invalidNamestring
--------------------	-----------------------

Definition at line 58 of file [ValidationTests.cs](#).

5.2.2.14 ValidateName_ValidName_ReturnsName()

```
void SmartStay.Tests.ValidatorTests.ValidateName_ValidName_ReturnsName (
    string validName ) [inline]
```

The ValidateName_ValidName_ReturnsName Tests the ValidateName method with a valid name input and ensures it returns the same name.

Parameters

<i>validName</i>	The validNamestring
------------------	---------------------

Definition at line 43 of file [ValidationTests.cs](#).

5.2.2.15 ValidatePayment_InvalidPayment_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePayment_InvalidPayment_ThrowsException ( ) [inline]
```

Validates that an exception is thrown when a negative payment value is provided.

Definition at line 406 of file [ValidationTests.cs](#).

5.2.2.16 ValidatePayment_ValidPayment_ReturnsPayment()

```
void SmartStay.Tests.ValidatorTests.ValidatePayment_ValidPayment_ReturnsPayment () [inline]
```

Validates that the payment amount is correctly processed when a valid payment is provided.

Definition at line 395 of file [ValidationTests.cs](#).

5.2.2.17 ValidatePaymentAmount_InvalidAmount_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentAmount_InvalidAmount_ThrowsException () [inline]
```

The ValidatePaymentAmount_InvalidAmount_ThrowsException Tests the ValidatePaymentAmount method with an invalid payment amount input (e.g., negative amount) and ensures a ValidationException is thrown.

Definition at line 243 of file [ValidationTests.cs](#).

5.2.2.18 ValidatePaymentAmount_ValidAmount_ReturnsAmount()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentAmount_ValidAmount_ReturnsAmount () [inline]
```

The ValidatePaymentAmount_ValidAmount_ReturnsAmount Tests the ValidatePaymentAmount method with a valid payment amount input and ensures it returns the same amount.

Definition at line 230 of file [ValidationTests.cs](#).

5.2.2.19 ValidatePaymentStatus_InvalidStatus_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentStatus_InvalidStatus_ThrowsException () [inline]
```

Validates that an exception is thrown when an invalid payment status (out of the predefined enum) is provided.

Definition at line 270 of file [ValidationTests.cs](#).

5.2.2.20 ValidatePaymentStatus_ValidStatus_ReturnsStatus()

```
void SmartStay.Tests.ValidatorTests.ValidatePaymentStatus_ValidStatus_ReturnsStatus () [inline]
```

Validates that the payment status is correctly processed when a valid status is provided.

Definition at line 259 of file [ValidationTests.cs](#).

5.2.2.21 ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException ( string invalidPhoneNumber ) [inline]
```

The ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException Tests the ValidatePhoneNumber method with an invalid phone number input and ensures a ValidationException is thrown.

Parameters

<i>invalidPhoneNumber</i>	The invalidPhoneNumberstring
---------------------------	------------------------------

Definition at line 154 of file [ValidationTests.cs](#).

5.2.2.22 ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber()

```
void SmartStay.Tests.ValidatorTests.ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber ( 
    string validPhoneNumber ) [inline]
```

The ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber Tests the ValidatePhoneNumber method with a valid phone number input and ensures it returns the same phone number.

Parameters

<i>validPhoneNumber</i>	The validPhoneNumberstring
-------------------------	----------------------------

Definition at line 140 of file [ValidationTests.cs](#).

5.2.2.23 ValidatePrice_InvalidPrice_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidatePrice_InvalidPrice_ThrowsException ( ) [inline]
```

The ValidatePrice_InvalidPrice_ThrowsException Tests the ValidatePrice method with an invalid price input (e.g., zero or negative) and ensures a ValidationException is thrown.

Definition at line 214 of file [ValidationTests.cs](#).

5.2.2.24 ValidatePrice_ValidPrice_ReturnsPrice()

```
void SmartStay.Tests.ValidatorTests.ValidatePrice_ValidPrice_ReturnsPrice ( ) [inline]
```

The ValidatePrice_ValidPrice_ReturnsPrice Tests the ValidatePrice method with a valid price input and ensures it returns the same price.

Definition at line 201 of file [ValidationTests.cs](#).

5.2.2.25 ValidateReservationStatus_InvalidStatus_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateReservationStatus_InvalidStatus_ThrowsException ( 
) [inline]
```

Validates that an exception is thrown when an invalid reservation status (out of the predefined enum) is provided.

Definition at line 433 of file [ValidationTests.cs](#).

5.2.2.26 ValidateReservationStatus_ValidStatus_ReturnsStatus()

```
void SmartStay.Tests.ValidatorTests.ValidateReservationStatus_ValidStatus_ReturnsStatus ( )  
[inline]
```

Validates that the reservation status is correctly processed when a valid status is provided.

Definition at line 421 of file [ValidationTests.cs](#).

5.2.2.27 ValidateTotalCost_InvalidCost_ThrowsException()

```
void SmartStay.Tests.ValidatorTests.ValidateTotalCost_InvalidCost_ThrowsException ( ) [inline]
```

Validates that an exception is thrown when a negative cost is provided.

Definition at line 380 of file [ValidationTests.cs](#).

5.2.2.28 ValidateTotalCost_ValidCost_ReturnsTotalCost()

```
void SmartStay.Tests.ValidatorTests.ValidateTotalCost_ValidCost_ReturnsTotalCost ( ) [inline]
```

Validates that the total cost is correctly processed when a valid cost is provided.

Definition at line 369 of file [ValidationTests.cs](#).

The documentation for this class was generated from the following file:

- [ValidationTests.cs](#)

Chapter 6

File Documentation

6.1 AccommodationTests.cs File Reference

6.2 AccommodationTests.cs

[Go to the documentation of this file.](#)
00001

6.3 ClientTests.cs File Reference

Data Structures

- class [SmartStay.Tests.ClientTests](#)

Namespaces

- namespace [SmartStay](#)
- namespace [SmartStay.Tests](#)

6.4 ClientTests.cs

[Go to the documentation of this file.](#)

```
00001 using Microsoft.VisualStudio.TestTools.UnitTesting;
00002 using SmartStay.Models;
00003 using SmartStay.Models.Enums;
00004 using SmartStay.Services;
00005 using SmartStay.Validation;
00006 using System;
00007
00008 namespace SmartStay.Tests
00009 {
00010     [TestClass]
00011     public class ClientTests
00012     {
00013         [TestMethod]
00014         public void Client_ValidData_CreatesClient()
00015         {
00016             var firstName = "John";
00017
00018             Assert.IsNotNull(client);
00019             Assert.AreEqual("John", client.FirstName);
00020         }
00021     }
00022 }
```

```
00021     var lastName = "Doe";
00022     var email = "john.doe@example.com";
00023     var client = new Client(firstName, lastName, email);
00024
00025     Assert.IsNotNull(client);
00026     Assert.AreEqual(firstName, client.FirstName);
00027     Assert.AreEqual(lastName, client.LastName);
00028     Assert.AreEqual(email, client.Email);
00029 }
00030
00035 [TestMethod]
00036 public void Client_FullData_CreatesClient()
00037 {
00038     var firstName = "Jane";
00039     var lastName = "Smith";
00040     var email = "jane.smith@example.com";
00041     var phoneNumber = "+351999999999";
00042     var address = "123 Elm St, Springfield";
00043     var preferredPaymentMethod = PaymentMethod.BankTransfer;
00044
00045     var client = new Client(firstName, lastName, email, phoneNumber, address,
00046     preferredPaymentMethod);
00047
00048     Assert.IsNotNull(client);
00049     Assert.AreEqual(firstName, client.FirstName);
00050     Assert.AreEqual(lastName, client.LastName);
00051     Assert.AreEqual(email, client.Email);
00052     Assert.AreEqual(phoneNumber, client.PhoneNumber);
00053     Assert.AreEqual(address, client.Address);
00054     Assert.AreEqual(preferredPaymentMethod, client.PreferredPaymentMethod);
00055 }
00056
00060 [TestMethod]
00061 public void Client_InvalidEmail_ThrowsValidationException()
00062 {
00063     var firstName = "John";
00064     var lastName = "Doe";
00065     var invalidEmail = "invalid-email";
00066
00067     var exception =
00068         Assert.ThrowsException<ValidationException>(() => new Client(firstName, lastName,
00069         invalidEmail));
00070     Assert.AreEqual(ValidationErrorCode.InvalidEmail, exception.ErrorCode);
00071 }
00076 [TestMethod]
00077 public void Client_InvalidPhoneNumber_ThrowsValidationException()
00078 {
00079     var firstName = "John";
00080     var lastName = "Doe";
00081     var email = "john.doe@example.com";
00082     var invalidPhoneNumber = "invalid-phone";
00083
00084     var exception = Assert.ThrowsException<ValidationException>(
00085         () => new Client(firstName, lastName, email, invalidPhoneNumber, "123 Elm St"));
00086     Assert.AreEqual(ValidationErrorCode.InvalidPhoneNumber, exception.ErrorCode);
00087 }
00088
00093 [TestMethod]
00094 public void Client_InvalidAddress_ThrowsValidationException()
00095 {
00096     var firstName = "John";
00097     var lastName = "Doe";
00098     var email = "john.doe@example.com";
00099     var phoneNumber = "+351999999999";
00100     var invalidAddress = "";
00101
00102     var exception = Assert.ThrowsException<ValidationException>(
00103         () => new Client(firstName, lastName, email, phoneNumber, invalidAddress));
00104     Assert.AreEqual(ValidationErrorCode.InvalidAddress, exception.ErrorCode);
00105 }
00106
00111 [TestMethod]
00112 public void Client_InvalidPaymentMethod_ThrowsValidationException()
00113 {
00114     var firstName = "Jane";
00115     var lastName = "Smith";
00116     var email = "jane.smith@example.com";
00117     var phoneNumber = "+351999999999";
00118     var address = "123 Elm St, Springfield";
00119     var invalidPaymentMethod = (PaymentMethod)999; // Invalid payment method
00120
00121     var exception = Assert.ThrowsException<ValidationException>(
00122         () => new Client(firstName, lastName, email, phoneNumber, address, invalidPaymentMethod));
00123     Assert.AreEqual(ValidationErrorCode.InvalidPaymentMethod, exception.ErrorCode);
00124 }
00125 }
```

```
00130     [TestMethod]
00131     public void Client_SetValidFirstName_UpdatesFirstName()
00132     {
00133         var firstName = "John";
00134         var lastName = "Doe";
00135         var email = "john.doe@example.com";
00136         var client = new Client(firstName, lastName, email);
00137
00138         var newFirstName = "Johnny";
00139         client.FirstName = newFirstName;
00140
00141         Assert.AreEqual(newFirstName, client.FirstName);
00142     }
00143
00144     [TestMethod]
00145     public void Client_SetValidLastName_UpdatesLastName()
00146     {
00147         var firstName = "John";
00148         var lastName = "Doe";
00149         var email = "john.doe@example.com";
00150         var client = new Client(firstName, lastName, email);
00151
00152         var newLastName = "Smith";
00153         client.LastName = newLastName;
00154
00155         Assert.AreEqual(newLastName, client.LastName);
00156     }
00157
00158     [TestMethod]
00159     public void Client_SetValidEmail_UpdatesEmail()
00160     {
00161         var firstName = "John";
00162         var lastName = "Doe";
00163         var email = "john.doe@example.com";
00164         var client = new Client(firstName, lastName, email);
00165
00166         var newEmail = "johnny.doe@example.com";
00167         client.Email = newEmail;
00168
00169         Assert.AreEqual(newEmail, client.Email);
00170     }
00171
00172     [TestMethod]
00173     public void Client_SetValidPhoneNumber_UpdatesPhoneNumber()
00174     {
00175         var firstName = "John";
00176         var lastName = "Doe";
00177         var email = "john.doe@example.com";
00178         var phoneNumber = "+351999999999";
00179         var client = new Client(firstName, lastName, email, phoneNumber, "123 Elm St");
00180
00181         var newPhoneNumber = "+351888888888";
00182         client.PhoneNumber = newPhoneNumber;
00183
00184         Assert.AreEqual(newPhoneNumber, client.PhoneNumber);
00185     }
00186
00187     [TestMethod]
00188     public void Client_SetValidAddress_UpdatesAddress()
00189     {
00190         var firstName = "John";
00191         var lastName = "Doe";
00192         var email = "john.doe@example.com";
00193         var phoneNumber = "+351999999999";
00194         var address = "123 Elm St, Springfield";
00195         var client = new Client(firstName, lastName, email, phoneNumber, address);
00196
00197         var newAddress = "456 Oak St, Springfield";
00198         client.Address = newAddress;
00199
00200         Assert.AreEqual(newAddress, client.Address);
00201     }
00202
00203     [TestMethod]
00204     public void Client_SetValidPaymentMethod_UpdatesPaymentMethod()
00205     {
00206         var firstName = "John";
00207         var lastName = "Doe";
00208         var email = "john.doe@example.com";
00209         var phoneNumber = "+351999999999";
00210         var address = "123 Elm St, Springfield";
00211         var preferredPaymentMethod = PaymentMethod.PayPal;
00212
00213         var client = new Client(firstName, lastName, email, phoneNumber, address,
00214             preferredPaymentMethod);
00215
00216         var newPaymentMethod = PaymentMethod.BankTransfer;
```

```

00236     client.PreferredPaymentMethod = newPaymentMethod;
00237
00238     Assert.AreEqual(newPaymentMethod, client.PreferredPaymentMethod);
00239 }
00240
00241 [TestMethod]
00242 public void Client_GenerateUniqueIdCreatesUniqueIds()
00243 {
00244     var firstClient = new Client("John", "Doe", "john.doe@example.com");
00245     var secondClient = new Client("Jane", "Smith", "jane.smith@example.com");
00246
00247     Assert.AreNotEqual(firstClient.Id, secondClient.Id);
00248 }
00249
00250 [TestMethod]
00251 public void Client_ToString_ReturnsValidJson()
00252 {
00253     var firstName = "John";
00254     var lastName = "Doe";
00255     var email = "john.doe@example.com";
00256     var client = new Client(firstName, lastName, email);
00257
00258     var json = client.ToString();
00259
00260     Assert.IsTrue(json.Contains("\"FirstName\": \"John\""));
00261     Assert.IsTrue(json.Contains("\"LastName\": \"Doe\""));
00262     Assert.IsTrue(json.Contains("\"Email\": \"john.doe@example.com\""));
00263 }
00264 }
00265
00266 }
```

6.5 ReservationTests.cs File Reference

6.6 ReservationTests.cs

[Go to the documentation of this file.](#)

00001

6.7 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs File Reference

6.8 .NETCoreApp,Version=v8.0.AssemblyAttributes.cs

[Go to the documentation of this file.](#)

```

00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp, Version=v8.0",
    FrameworkDisplayName = ".NET 8.0")]

```

6.9 SmartStay.Tests.AssemblyInfo.cs File Reference

6.10 SmartStay.Tests.AssemblyInfo.cs

[Go to the documentation of this file.](#)

```

00001 //-----
00002 // <auto-generated>
00003 //   This code was generated by a tool.
00004 //   Runtime Version:4.0.30319.42000
00005 // 
00006 //   Changes to this file may cause incorrect behavior and will be lost if

```

```

00007 //      the code is regenerated.
00008 // </auto-generated>
00009 //-----
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Tests")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
    System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+547a43e26c0e6800e3b3c5899fba97a9e932c12b")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Tests")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Tests")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023

```

6.11 SmartStay.Tests.GlobalUsings.g.cs File Reference

6.12 SmartStay.Tests.GlobalUsings.g.cs

[Go to the documentation of this file.](#)

```

00001 // <auto-generated>
00002 global using global::Microsoft.VisualStudio.TestTools.UnitTesting;
00003 global using global::System;
00004 global using global::System.Collections.Generic;
00005 global using global::System.IO;
00006 global using global::System.Linq;
00007 global using global::System.Net.Http;
00008 global using global::System.Threading;
00009 global using global::System.Threading.Tasks;

```

6.13 ValidationTests.cs File Reference

Data Structures

- class [SmartStay.Tests.ValidatorTests](#)

Defines the ValidatorTests class which tests the validation logic of various input parameters such as names, email addresses, accommodation types, prices, payment details, and more.

Namespaces

- namespace [SmartStay](#)
- namespace [SmartStay.Tests](#)

6.14 ValidationTests.cs

[Go to the documentation of this file.](#)

```

00001
00002 using SmartStay.Models.Enums;
00003 using SmartStay.Validation;
00004
00005 namespace SmartStay.Tests
00006 {
00007     [TestClass]
00008     public class ValidatorTests
00009     {
00010         #region Name Validation Tests

```

```
00035
00041 [TestMethod]
00042 [DataRow("John Doe")]
00043 public void ValidateName_ValidName_ReturnsName(string validName)
00044 {
00045     var result = Validator.ValidateName(validName);
00046     Assert.AreEqual(validName, result);
00047 }
00048
00055 [TestMethod]
00056 [DataRow("")]
00057 [DataRow(null)]
00058 public void ValidateName_InvalidName_ThrowsException(string invalidName)
00059 {
00060     var exception = Assert.ThrowsException<ValidationException>(() =>
00061         Validator.ValidateName(invalidName));
00062     Assert.AreEqual(ValidationErrorCode.InvalidName, exception.ErrorCode);
00063 }
00064 #endregion
00065
00066 #region Accommodation Name Validation Tests
00067
00073 [TestMethod]
00074 [DataRow("Ocean View Resort")]
00075 public void ValidateAccommodationName_ValidName_ReturnsName(string validAccommodationName)
00076 {
00077     var result = Validator.ValidateAccommodationName(validAccommodationName);
00078     Assert.AreEqual(validAccommodationName, result);
00079 }
00080
00087 [TestMethod]
00088 [DataRow("")]
00089 [DataRow(null)]
00090 public void ValidateAccommodationName_InvalidName_ThrowsException(string invalidAccommodationName)
00091 {
00092     var exception = Assert.ThrowsException<ValidationException>(
00093         () => Validator.ValidateAccommodationName(invalidAccommodationName));
00094     Assert.AreEqual(ValidationErrorCode.InvalidAccommodationName, exception.ErrorCode);
00095 }
00096
00097 #endregion
00098
00099 #region Email Validation Tests
00100
00106 [TestMethod]
00107 [DataRow("user@example.com")]
00108 public void ValidateEmail_ValidEmail_ReturnsEmail(string validEmail)
00109 {
00110     var result = Validator.ValidateEmail(validEmail);
00111     Assert.AreEqual(validEmail, result);
00112 }
00113
00120 [TestMethod]
00121 [DataRow("invalid-email")]
00122 public void ValidateEmail_InvalidEmail_ThrowsException(string invalidEmail)
00123 {
00124     var exception = Assert.ThrowsException<ValidationException>(() =>
00125         Validator.ValidateEmail(invalidEmail));
00126     Assert.AreEqual(ValidationErrorCode.InvalidEmail, exception.ErrorCode);
00127 }
00128
00129 #endregion
00130 #region Phone Number Validation Tests
00131
00138 [TestMethod]
00139 [DataRow("+3351234567890")]
00140 public void ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber(string validPhoneNumber)
00141 {
00142     var result = Validator.ValidatePhoneNumber(validPhoneNumber);
00143     Assert.AreEqual(validPhoneNumber, result);
00144 }
00145
00152 [TestMethod]
00153 [DataRow("12345")]
00154 public void ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException(string invalidPhoneNumber)
00155 {
00156     var exception =
00157         Assert.ThrowsException<ValidationException>(() =>
00158             Validator.ValidatePhoneNumber(invalidPhoneNumber));
00159     Assert.AreEqual(ValidationErrorCode.InvalidPhoneNumber, exception.ErrorCode);
00160 }
00161
00162 #endregion
00163 #region Address Validation Tests
```

```
00164
00170     [TestMethod]
00171     [DataRow("123 Main St")]
00172     public void ValidateAddress_ValidAddress_ReturnsAddress(string validAddress)
00173     {
00174         var result = Validator.ValidateAddress(validAddress);
00175         Assert.AreEqual(validAddress, result);
00176     }
00177
00184     [TestMethod]
00185     [DataRow("")]
00186     public void ValidateAddress_InvalidAddress_ThrowsException(string invalidAddress)
00187     {
00188         var exception = Assert.ThrowsException<ValidationException>(() =>
00189             Validator.ValidateAddress(invalidAddress));
00190         Assert.AreEqual(ValidationErrorCode.InvalidAddress, exception.ErrorCode);
00191     }
00192 #endregion
00193
00194 #region Price Validation Tests
00195
00200     [TestMethod]
00201     public void ValidatePrice_ValidPrice_ReturnsPrice()
00202     {
00203         decimal validPrice = 100.0m;
00204         var result = Validator.ValidatePrice(validPrice);
00205         Assert.AreEqual(validPrice, result);
00206     }
00207
00213     [TestMethod]
00214     public void ValidatePrice_InvalidPrice_ThrowsException()
00215     {
00216         decimal invalidPrice = 0.0m;
00217         var exception = Assert.ThrowsException<ValidationException>(() =>
00218             Validator.ValidatePrice(invalidPrice));
00219         Assert.AreEqual(ValidationErrorCode.InvalidPrice, exception.ErrorCode);
00220     }
00221 #endregion
00222
00223 #region Payment Amounts Validation Tests
00224
00229     [TestMethod]
00230     public void ValidatePaymentAmount_ValidAmount_ReturnsAmount()
00231     {
00232         decimal validAmount = 150.0m;
00233         var result = Validator.ValidatePaymentAmount(validAmount);
00234         Assert.AreEqual(validAmount, result);
00235     }
00236
00242     [TestMethod]
00243     public void ValidatePaymentAmount_InvalidAmount_ThrowsException()
00244     {
00245         decimal invalidAmount = -10.0m;
00246         var exception =
00247             Assert.ThrowsException<ValidationException>(() =>
00248                 Validator.ValidatePaymentAmount(invalidAmount));
00249         Assert.AreEqual(ValidationErrorCode.InvalidPaymentValue, exception.ErrorCode);
00250     }
00251 #endregion
00252
00253 #region Payment Status Validation Tests
00254
00258     [TestMethod]
00259     public void ValidatePaymentStatus_ValidStatus_ReturnsStatus()
00260     {
00261         var validStatus = PaymentStatus.Completed;
00262         var result = Validator.ValidatePaymentStatus(validStatus);
00263         Assert.AreEqual(validStatus, result);
00264     }
00265
00269     [TestMethod]
00270     public void ValidatePaymentStatus_InvalidStatus_ThrowsException()
00271     {
00272         var invalidStatus = (PaymentStatus)999;
00273         var exception =
00274             Assert.ThrowsException<ValidationException>(() =>
00275                 Validator.ValidatePaymentStatus(invalidStatus));
00276         Assert.AreEqual(ValidationErrorCode.InvalidPaymentStatus, exception.ErrorCode);
00277     }
00278 #endregion
00279
00280 #region Accommodation Type Validation Tests
00281
```

```
00285     [TestMethod]
00286     public void ValidateAccommodationType_ValidType_ReturnsType()
00287     {
00288         var validType = AccommodationType.Hotel;
00289         var result = Validator.ValidateAccommodationType(validType);
00290         Assert.AreEqual(validType, result);
00291     }
00292
00293     [TestMethod]
00294     public void ValidateAccommodationType_InvalidType_ThrowsException()
00295     {
00296         var invalidType = (AccommodationType)999;
00297         var exception =
00298             Assert.ThrowsException<ValidationException>(() =>
00299                 Validator.ValidateAccommodationType(invalidType));
00300         Assert.AreEqual(ValidationErrorCode.InvalidAccommodationType, exception.ErrorCode);
00301     }
00302
00303 #endregion
00304
00305 #region Check In Date Validation Tests
00306
00307     [TestMethod]
00308     public void ValidateCheckInDate_ValidFutureDate_ReturnsDate()
00309     {
00310         var futureDate = DateTime.Now.AddDays(1);
00311         var result = Validator.ValidateCheckInDate(futureDate);
00312         Assert.AreEqual(futureDate, result);
00313     }
00314
00315     [TestMethod]
00316     public void ValidateCheckInDate_PastDate_ThrowsException()
00317     {
00318         var pastDate = DateTime.Now.AddDays(-1);
00319         var exception = Assert.ThrowsException<ValidationException>(() =>
00320             Validator.ValidateCheckInDate(pastDate));
00321         Assert.AreEqual(ValidationErrorCode.InvalidDate, exception.ErrorCode);
00322     }
00323
00324 #endregion
00325
00326 #region Check Out Date Validation Tests
00327
00328     [TestMethod]
00329     public void ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate()
00330     {
00331         var checkInDate = DateTime.Now.AddDays(1);
00332         var checkOutDate = checkInDate.AddDays(1);
00333         var result = Validator.ValidateCheckOutDate(checkOutDate, checkInDate);
00334         Assert.AreEqual(checkOutDate, result);
00335     }
00336
00337     [TestMethod]
00338     public void ValidateCheckOutDate_InvalidDateRange_ThrowsException()
00339     {
00340         var checkInDate = DateTime.Now.AddDays(1);
00341         var checkOutDate = checkInDate.AddDays(-1);
00342         var exception = Assert.ThrowsException<ValidationException>(
00343             () => Validator.ValidateCheckOutDate(checkOutDate, checkInDate));
00344         Assert.AreEqual(ValidationErrorCode.InvalidDateRange, exception.ErrorCode);
00345     }
00346
00347 #endregion
00348
00349 #region Total Cost Validation Tests
00350
00351     [TestMethod]
00352     public void ValidateTotalCost_ValidCost_ReturnsTotalCost()
00353     {
00354         decimal validCost = 300.0m;
00355         var result = Validator.ValidateTotalCost(validCost);
00356         Assert.AreEqual(validCost, result);
00357     }
00358
00359     [TestMethod]
00360     public void ValidateTotalCost_InvalidCost_ThrowsException()
00361     {
00362         decimal invalidCost = -5.0m;
00363         var exception = Assert.ThrowsException<ValidationException>(() =>
00364             Validator.ValidateTotalCost(invalidCost));
00365         Assert.AreEqual(ValidationErrorCode.InvalidTotalCost, exception.ErrorCode);
00366     }
00367
00368 #endregion
00369
00370 #region Payment Validation Tests
00371
```

```
00394     [TestMethod]
00395     public void ValidatePayment_ValidPayment_ReturnsPayment()
00396     {
00397         decimal validPayment = 10.0m;
00398         var result = Validator.ValidatePayment(validPayment);
00399         Assert.AreEqual(validPayment, result);
00400     }
00401
00402     [TestMethod]
00403     public void ValidatePayment_InvalidPayment_ThrowsException()
00404     {
00405         decimal invalidPayment = -100.0m;
00406         var exception = Assert.ThrowsException<ValidationException>(() =>
00407             Validator.ValidatePayment(invalidPayment));
00408         Assert.AreEqual(ValidationErrorCode.InvalidPaymentValue, exception.ErrorCode);
00409     }
00410
00411 }
00412
00413 #endregion
00414
00415 #region Reservation Status Validation Tests
00416
00417     [TestMethod]
00418     public void ValidateReservationStatus_ValidStatus_ReturnsStatus()
00419     {
00420         var validStatus = ReservationStatus.Confirmed;
00421         var result = Validator.ValidateReservationStatus(validStatus);
00422         Assert.AreEqual(validStatus, result);
00423     }
00424
00425     [TestMethod]
00426     public void ValidateReservationStatus_InvalidStatus_ThrowsException()
00427     {
00428         var invalidStatus = (ReservationStatus)999;
00429         var exception =
00430             Assert.ThrowsException<ValidationException>(() =>
00431                 Validator.ValidateReservationStatus(invalidStatus));
00432         Assert.AreEqual(ValidationErrorCode.InvalidReservationStatus, exception.ErrorCode);
00433     }
00434
00435 }
00436
00437 #endregion
00438
00439 }
```


Index

.NETCoreApp, Version=v8.0.AssemblyAttributes.cs, 24
AccommodationTests.cs, 21
Client_FullData_CreatesClient
 SmartStay.Tests.ClientTests, 10
Client_GenerateUniqueClientId_CreatesUniquelds
 SmartStay.Tests.ClientTests, 10
Client_InvalidAddress_ThrowsValidationException
 SmartStay.Tests.ClientTests, 10
Client_InvalidEmail_ThrowsValidationException
 SmartStay.Tests.ClientTests, 10
Client_InvalidPaymentMethod_ThrowsValidationException
 SmartStay.Tests.ClientTests, 10
Client_InvalidPhoneNumber_ThrowsValidationException
 SmartStay.Tests.ClientTests, 10
Client_SetValidAddress_UpdatesAddress
 SmartStay.Tests.ClientTests, 11
Client_SetValidEmail_UpdatesEmail
 SmartStay.Tests.ClientTests, 11
Client_SetValidFirstName_UpdatesFirstName
 SmartStay.Tests.ClientTests, 11
Client_SetValidLastName_UpdatesLastName
 SmartStay.Tests.ClientTests, 11
Client_SetValidPaymentMethod_UpdatesPaymentMethod
 SmartStay.Tests.ClientTests, 11
Client_SetValidPhoneNumber_UpdatesPhoneNumber
 SmartStay.Tests.ClientTests, 11
Client_ToString_ReturnsValidJson
 SmartStay.Tests.ClientTests, 12
Client_ValidData_CreatesClient
 SmartStay.Tests.ClientTests, 12
ClientTests.cs, 21
ReservationTests.cs, 24
SmartStay, 7
SmartStay.Tests, 7
SmartStay.Tests.AssemblyInfo.cs, 24
SmartStay.Tests.ClientTests, 9
 Client_FullData_CreatesClient, 10
 Client_GenerateUniqueClientId_CreatesUniquelds,
 10
 Client_InvalidAddress_ThrowsValidationException,
 10
 Client_InvalidEmail_ThrowsValidationException,
 10
 Client_InvalidPaymentMethod_ThrowsValidationException, ValidatePaymentStatus_InvalidStatus_ThrowsException,
 10
 Client_InvalidPhoneNumber_ThrowsValidationException, ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException,
 10
Client_SetValidAddress_UpdatesAddress, 11
Client_SetValidEmail_UpdatesEmail, 11
Client_SetValidFirstName_UpdatesFirstName, 11
Client_SetValidLastName_UpdatesLastName, 11
Client_SetValidPaymentMethod_UpdatesPaymentMethod,
 11
Client_SetValidPhoneNumber_UpdatesPhoneNumber,
 11
Client_ToString_ReturnsValidJson, 12
Client_ValidData_CreatesClient, 12
SmartStay.Tests.GlobalUsings.g.cs, 25
SmartStay.Tests.ValidatorTests, 12
 ValidateAccommodationName_InvalidName_ThrowsException,
 14
 ValidateAccommodationName_ValidName_ReturnsName,
 14
 ValidateAccommodationType_InvalidType_ThrowsException,
 15
 ValidateAccommodationType_ValidType_ReturnsType,
 15
 ValidateAddress_InvalidAddress_ThrowsException,
 15
 ValidateAddress_ValidAddress_ReturnsAddress,
 15
 ValidateCheckInDate_PastDate_ThrowsException,
 16
 ValidateCheckInDate_ValidFutureDate_ReturnsDate,
 16
 ValidateCheckOutDate_InvalidDateRange_ThrowsException,
 16
 ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate,
 16
 ValidateEmail_InvalidEmail_ThrowsException, 16
 ValidateEmail_ValidEmail_ReturnsEmail, 17
 ValidateName_InvalidName_ThrowsException, 17
 ValidateName_ValidName_ReturnsName, 17
 ValidatePayment_InvalidPayment_ThrowsException,
 17
 ValidatePayment_ValidPayment_ReturnsPayment,
 18
 ValidatePaymentAmount_InvalidAmount_ThrowsException,
 18
 ValidatePaymentAmount_ValidAmount_ReturnsAmount,
 18
 ValidatePaymentStatus_InvalidStatus_ThrowsException,
 18
 ValidatePaymentStatus_ValidStatus_ReturnsStatus,
 18

ValidatePrice_ValidPrice_ReturnsPrice
SmartStay.Tests.ValidatorTests, 19

ValidateReservationStatus_InvalidStatus_ThrowsException
SmartStay.Tests.ValidatorTests, 19

ValidateReservationStatus_ValidStatus_ReturnsStatus
SmartStay.Tests.ValidatorTests, 19

ValidateTotalCost_InvalidCost_ThrowsException
SmartStay.Tests.ValidatorTests, 20

ValidateTotalCost_ValidCost_ReturnsTotalCost
SmartStay.Tests.ValidatorTests, 20

ValidationTests.cs, 25

ValidateAccommodationName_InvalidName_ThrowsException
SmartStay.Tests.ValidatorTests, 14

ValidateAccommodationName_ValidName_ReturnsName
SmartStay.Tests.ValidatorTests, 14

ValidateAccommodationType_InvalidType_ThrowsException
SmartStay.Tests.ValidatorTests, 15

ValidateAccommodationType_ValidType_ReturnsType
SmartStay.Tests.ValidatorTests, 15

ValidateAddress_InvalidAddress_ThrowsException
SmartStay.Tests.ValidatorTests, 15

ValidateAddress_ValidAddress_ReturnsAddress
SmartStay.Tests.ValidatorTests, 15

ValidateCheckInDate_PastDate_ThrowsException
SmartStay.Tests.ValidatorTests, 16

ValidateCheckInDate_ValidFutureDate_ReturnsDate
SmartStay.Tests.ValidatorTests, 16

ValidateCheckOutDate_InvalidDateRange_ThrowsException
SmartStay.Tests.ValidatorTests, 16

ValidateCheckOutDate_ValidDateRange_ReturnsCheckOutDate
SmartStay.Tests.ValidatorTests, 16

ValidateEmail_InvalidEmail_ThrowsException
SmartStay.Tests.ValidatorTests, 16

ValidateEmail_ValidEmail_ReturnsEmail
SmartStay.Tests.ValidatorTests, 17

ValidateName_InvalidName_ThrowsException
SmartStay.Tests.ValidatorTests, 17

ValidateName_ValidName_ReturnsName
SmartStay.Tests.ValidatorTests, 17

ValidatePayment_InvalidPayment_ThrowsException
SmartStay.Tests.ValidatorTests, 17

ValidatePayment_ValidPayment_ReturnsPayment
SmartStay.Tests.ValidatorTests, 18

ValidatePaymentAmount_InvalidAmount_ThrowsException
SmartStay.Tests.ValidatorTests, 18

ValidatePaymentAmount_ValidAmount_ReturnsAmount
SmartStay.Tests.ValidatorTests, 18

ValidatePaymentStatus_InvalidStatus_ThrowsException
SmartStay.Tests.ValidatorTests, 18

ValidatePaymentStatus_ValidStatus_ReturnsStatus
SmartStay.Tests.ValidatorTests, 18

ValidatePhoneNumber_InvalidPhoneNumber_ThrowsException
SmartStay.Tests.ValidatorTests, 18

ValidatePhoneNumber_ValidPhoneNumber_ReturnsPhoneNumber
SmartStay.Tests.ValidatorTests, 19

ValidatePrice_InvalidPrice_ThrowsException
SmartStay.Tests.ValidatorTests, 19