**Part 1**

1. **Feature: Customer books flight in the past**

Scenario: Customer wants to book a flight in the past

Given the date of today

When the Customer tries to select a departing date being the date of yesterday

Then this should not be possible

And the first possibility should be the date of today

1. **Feature: subscription to newsletter**

Scenario: incorrect e-mailadress

Given the e-mailadress is invalid

When the customer types in an e-mail without the ‘@’ symbol

Then their would prompt a notification

And the e-mailadress should at least have this format ‘name@name.nl’

P.S. I’ve tried two scenario’s. One without the ‘@’ symbol and one with a invalid character (‘!’). For the first one I would expect that the notification tells me that the ‘@’ symbol is missing. It doesn’t. It gives for both scenario’s the same notification…

1. **Feature: Customer books round trip**

Scenario: Customer wants to book a round trip

Given the date of today

And the ‘Return on’ checkbox is not checked

When the Customer tries to fill in a ‘Return on’ date

Then this should be possible manualy and by selecting a date

And after filling a ‘Return on’ date, the checkbox is automatically checked

**Part 2**

1. Automating error dates

var today = DateTime.Now;

var yesterday = today.AddDays(-1);

**namespace** Transavia.Specs

{

[Binding]

**public class** CalculateYesterdaysDate

{

**private int** result { **get**; **set**; }

**private** YesterdaysDate yesterdaysDate = **new** YesterdaysDate();

[Given(@"I have entered the date of yesterday into the ‘Depart on’ field")] **public void** GivenIHaveEnteredTheDateOfYesterday(**int** yesterday)

{

yesterdaysDate.Date = yesterday;

}

[When(@"I press enter")] **public void** WhenIPressEnter() { result = yesterdaysdate.Add(); }

[Then(@"the result should be giving an error on the screen")] **public void** ThenTheResultShouldBeOnTheScreen(**int** expectedResult)

{

Assert.Error(expectedResult, result);

}

}

}

1. Automating email validation

var email = String.Format(“name”, @, “name”

**namespace** Transavia.Specs

{

[Binding]

**public class** ValidateEmail

{

**private int** result { **get**; **set**; }

**private** Email email = **new** validateEmail();

[Given(@"I have entered a random@email in the newsletter field")] **public void** GivenIHaveEnteredRandomEmail(**string** email)

{

validateEmail = email;

}

[When(@"I press enter")] **public void** WhenIPressEnter() { result = ValidateEmail(); }

[Then(@"the result should be giving an error on the screen")] **public void** ThenTheResultShouldBeOnTheScreen(**int** expectedResult)

{

Assert.Error(expectedResult, result);

}

}

}