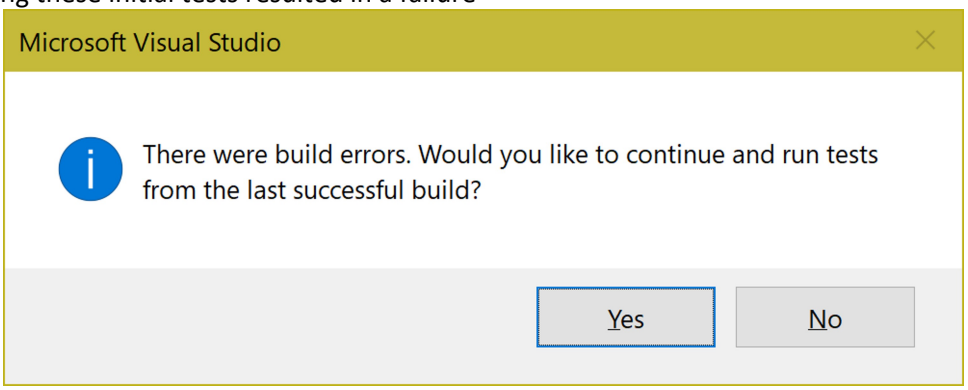
Table Booking TDD

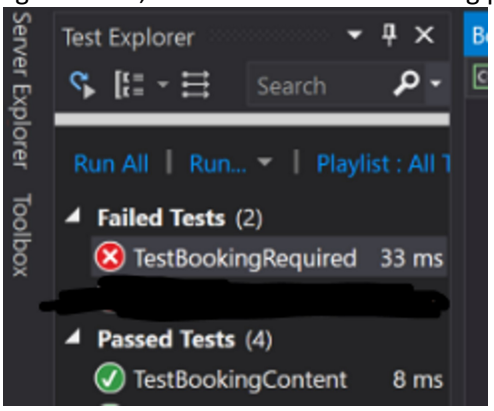
Tuesday, 9 May 2017 10:11 AM

- The initial tests were created by Kevin
  -
```
[TestMethod]
[DataRow(1, 50, 20, "Branch1")]
0 references
public void TestBookingContent(int number, int cap, int able, string name)
{
    Booking Book = new Booking() { Number = number, TableCapacity = cap, Available = able, StoreName = name };

    Assert.AreEqual(number, Book.Number);
    Assert.AreEqual(cap, Book.TableCapacity);
    Assert.AreEqual(able, Book.Available);
    Assert.AreEqual(name, Book.StoreName);
    if (Book.TableCapacity < Book.Available)
    {
        Assert.Fail();
    }

    //Required Attribute
    Assert.IsNotNull(Book.TableCapacity);
    Assert.IsNotNull(Book.Available);
    Assert.IsNotNull(Book.StoreName);
}
```
  -
```
[TestMethod]
[DataRow(3, null, 10, "")]
[DataRow(1, 78, null, "branch")]
[DataRow(2, 78, 22, null)]
0 references
public void TestBookingRequired(int number, int cap, int able, string name)
{
    Booking Book = new Booking() { Number = number, TableCapacity = cap, Available = able, StoreName = name };
    if (Book.TableCapacity == null)
        Assert.Fail();
    if (Book.Available == null)
        Assert.Fail();
    if (Book.StoreName == null)
        Assert.Fail();
}
```
- Running these initial tests resulted in a failure
  - 

- To make these work, we need to implement the class for Booking and the required properties

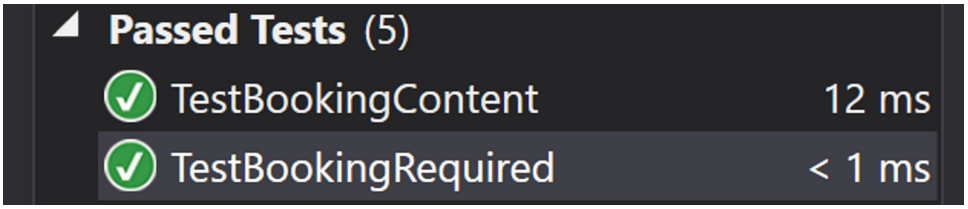
```
///<summary>
/// Bookings encompass the table logic of the restaurant.
///</summary>
0 references
public class Booking
{
    ///<summary>
    /// The name of the Store which the booking relates to.
    ///</summary>
    0 references
    public string StoreName { get; set; }
    ///<summary>
    /// The table number
    ///</summary>
    0 references
    public int Number { get; set; }
    ///<summary>
    /// The capacity of the table, used to figure out if available is
    ///</summary>
    0 references
    public int TableCapacity { get; set; }
    ///<summary>
    /// The amount that's been booked
    ///</summary>
    ///<remarks>
    /// Might change name of this, I don't like the structure
    ///</remarks>
    0 references
    public int Available { get; set; }
}
```

- Running the tests, we see that content testing passes but the required tests fail



- For testing purposes it is right to assume these tests would pass as a value would have to be passed in as Int cannot be null; changing the parameters that are passed in to non-nullable integers would pass these tests

```
[TestMethod]
[DataRow(3, null, 10, "")]
[DataRow(1, 78, null, "branch")]
[DataRow(2, 78, 22, null)]
public void TestBookingRequired(int number, int cap, int able, string name)
{
    Booking Book = new Booking() { Number = number, TableCapacity = cap, Available = able,
    if (Book.TableCapacity == null)
        Assert.Fail();
    if (Book.Available == null)
        Assert.Fail();
    if (Book.StoreName == null)
        Assert.Fail();
}
```

- 

- Booking has now been implemented to satisfy these unit tests