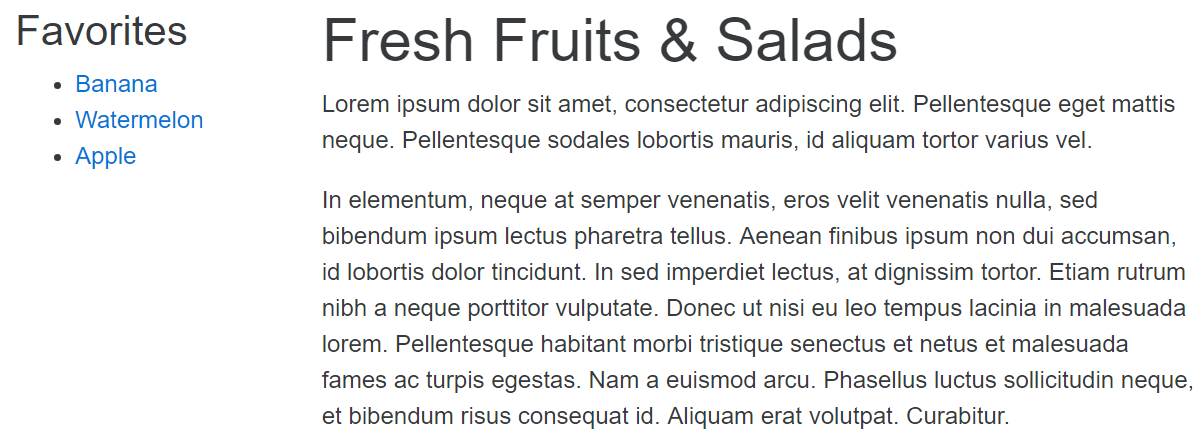
# Exercises: Bootstrap and State Management

Problems for exercises and homework for the [“C# Web Basics” course @ SoftUni](https://softuni.bg/courses/csharp-web-development-basics).

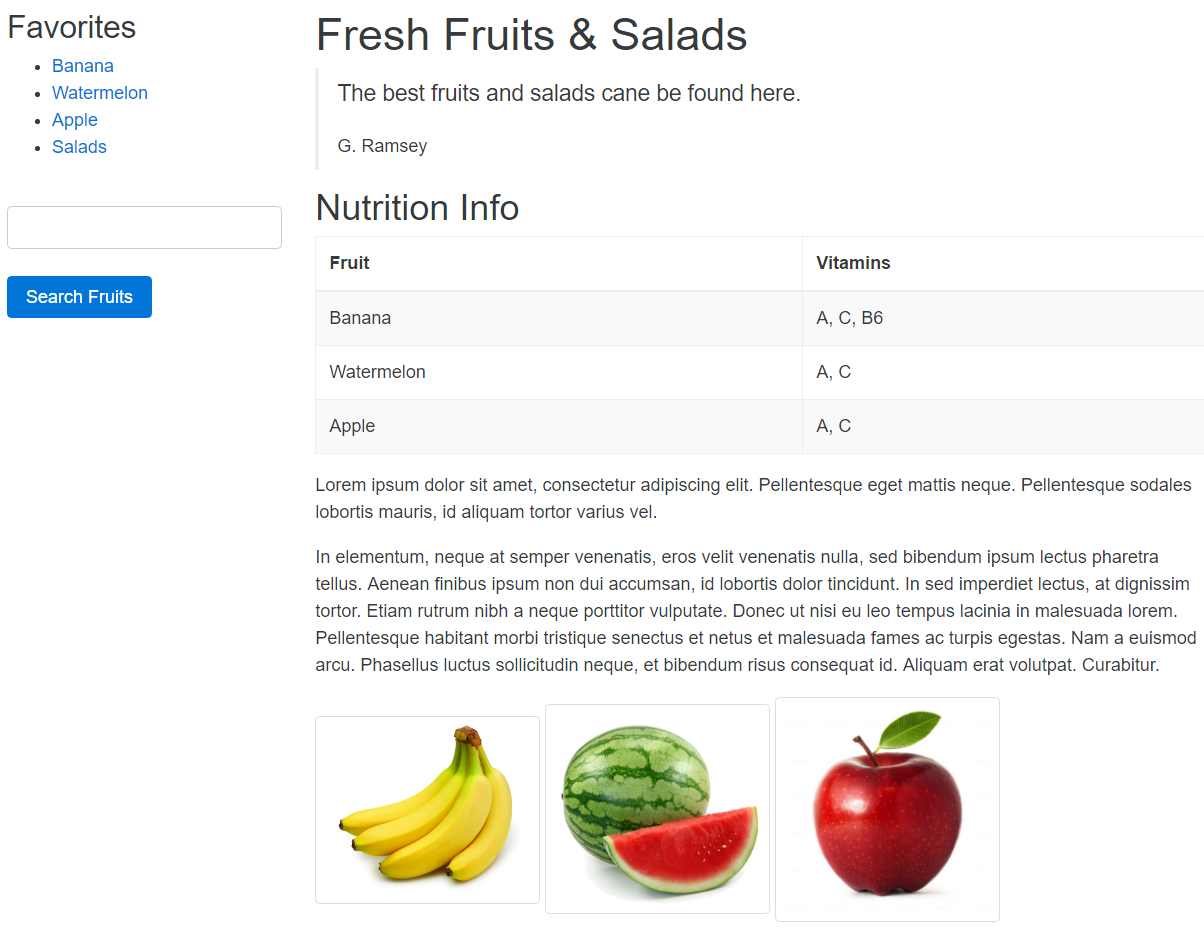
## Fruits and Salads: Initial Page Design

You are given a file **fruits.html**. Make it look like the picture below using Bootstrap.



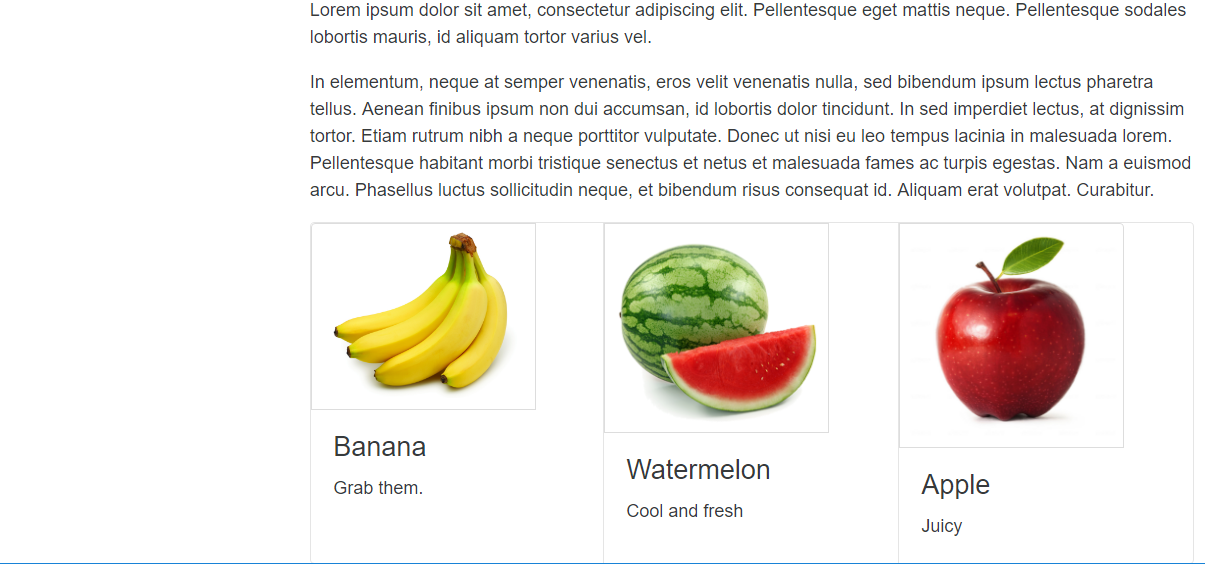
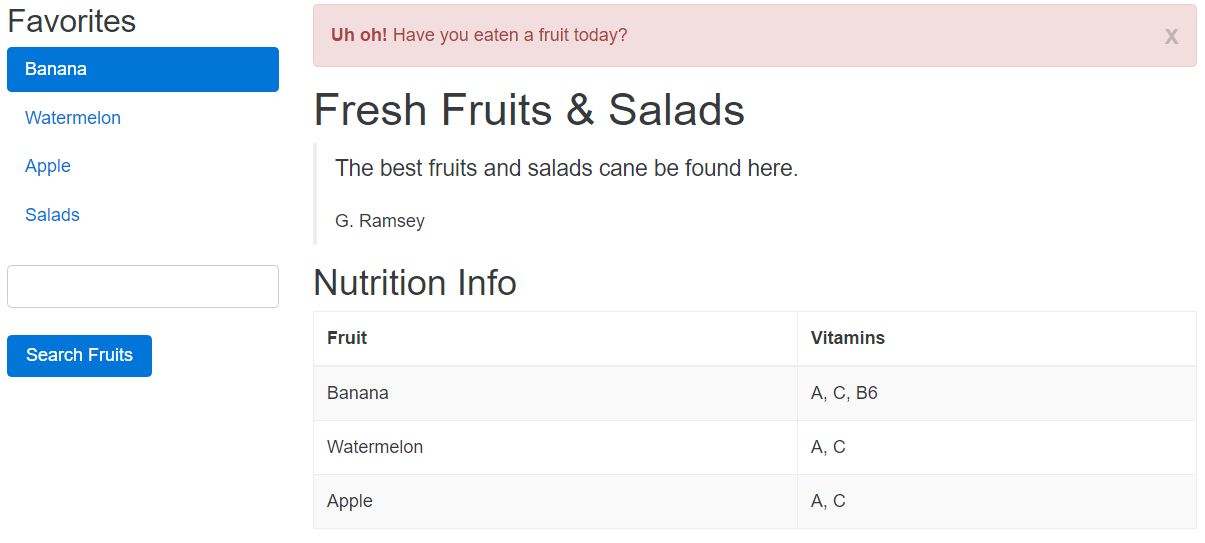
## Fruits and Salads: Improved Page Design

Improve the design of the page by **adding a salad section** and **search box** in side bar, **quote**, **table** and **3 image thumbnails**. (use any 3 pictures you want). At the end of this exercise the page should look like the picture below.



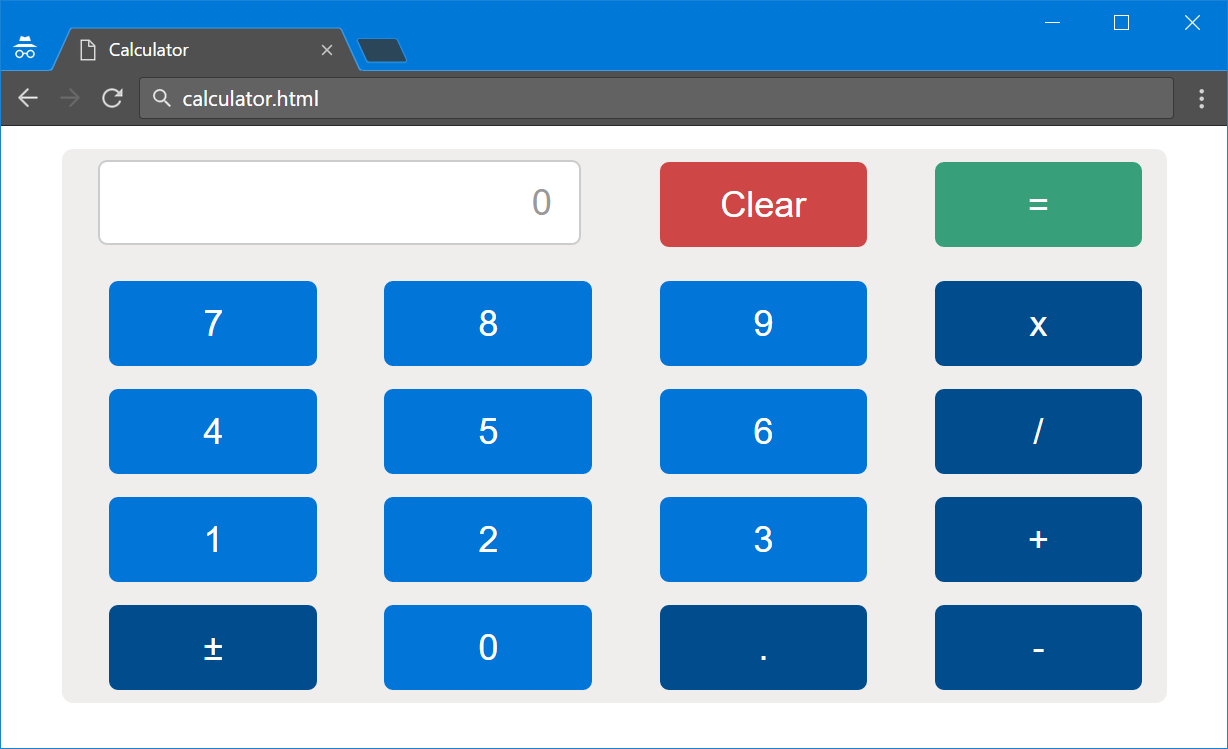
## Fruits and Salads: Final Page Design

Now let’s make final touches in our page. **Add alert** at the top of the page that can be **dismissed** and **turn the side bar into nav element**, displayed as **stacked pills**. Also, transform the image thumbnails at the bottom of the page to **cards with card block under them**, containing short information about them. The final result should look like the picture below.



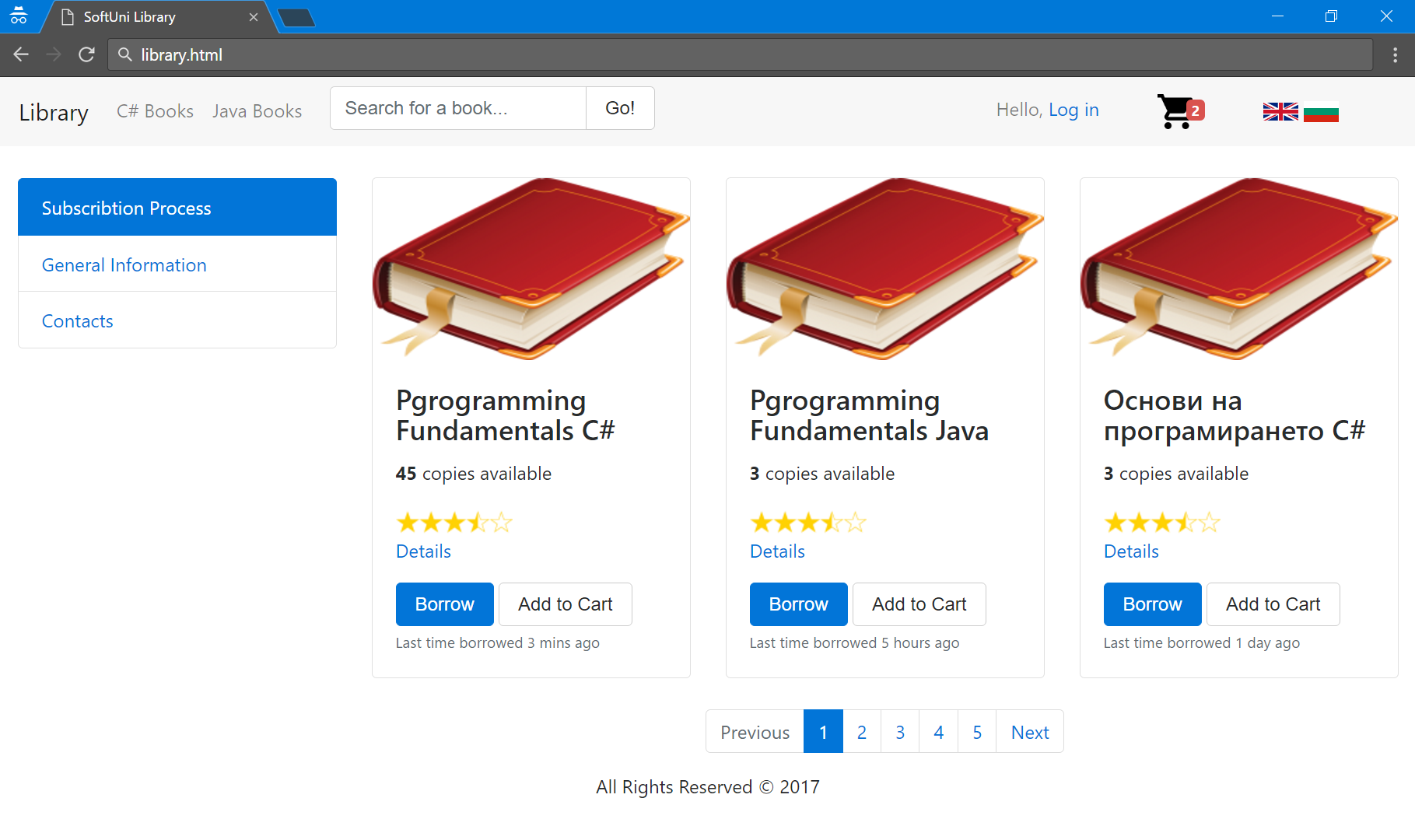
## Calculator

Using **HTML** and **Bootstrap** and **CSS** **create a calculator** that looks like the one in the picture below.



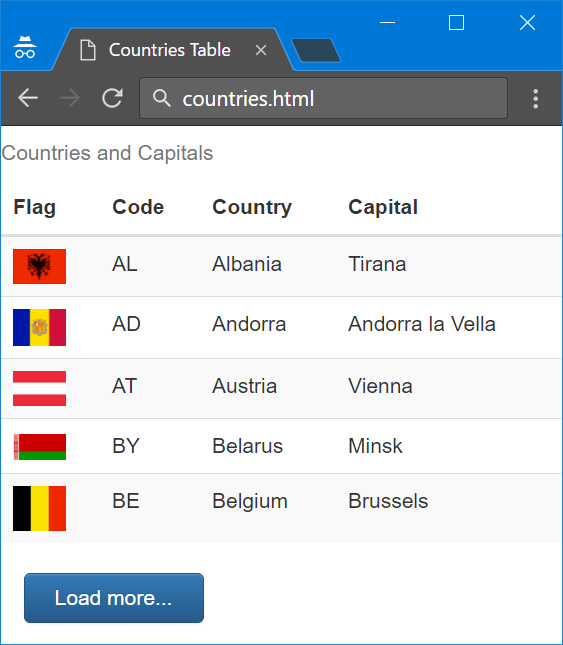
## Online Library

Using HTML and Bootstrap **create a main page for an online library** like the image provided below. Use the provided resources to help you.



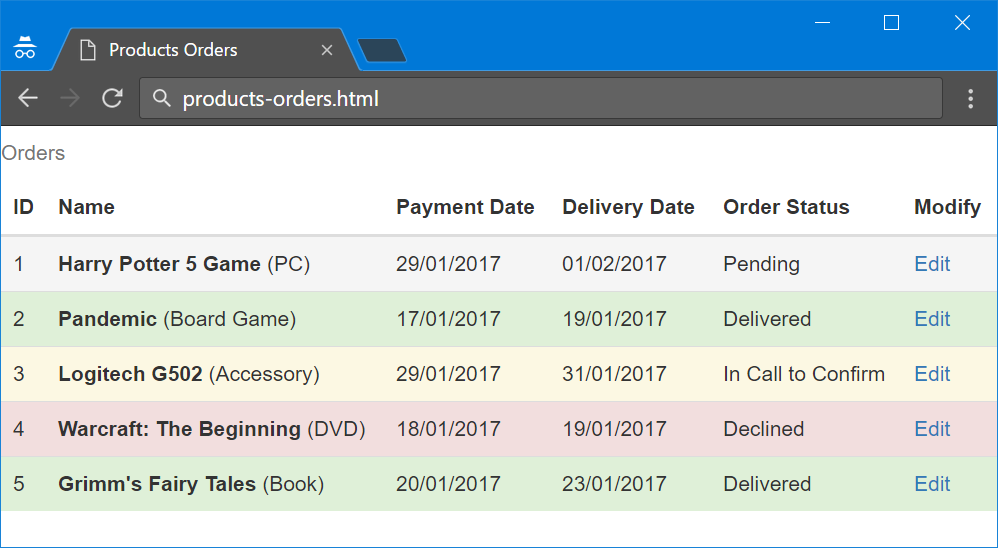
## Table of Countries

Using HTML and Bootstrap create a table containing several countries and a button to underneath the table. The final result should look like the picture below.



## Products Delivery Info

Using HTML and Bootstrap create a table containing information about some product orders. Take a look at the picture below for better understanding.

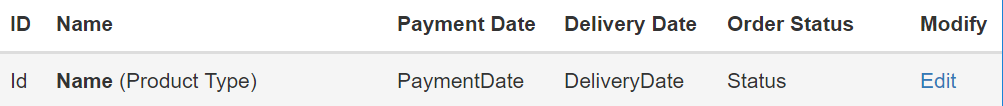


## \*Products Delivery Application

Use the previous task as a starting point but add some functionality to it. Keep information about all orders in a database with 2 tables

* **Orders** - Id, Name, ProductType, PaymentDate, DeliveryDate, Status.
* **Status** - Id, Name. With values **Pending**, **Delivered**, **Declined**, **In Call to Confirm**

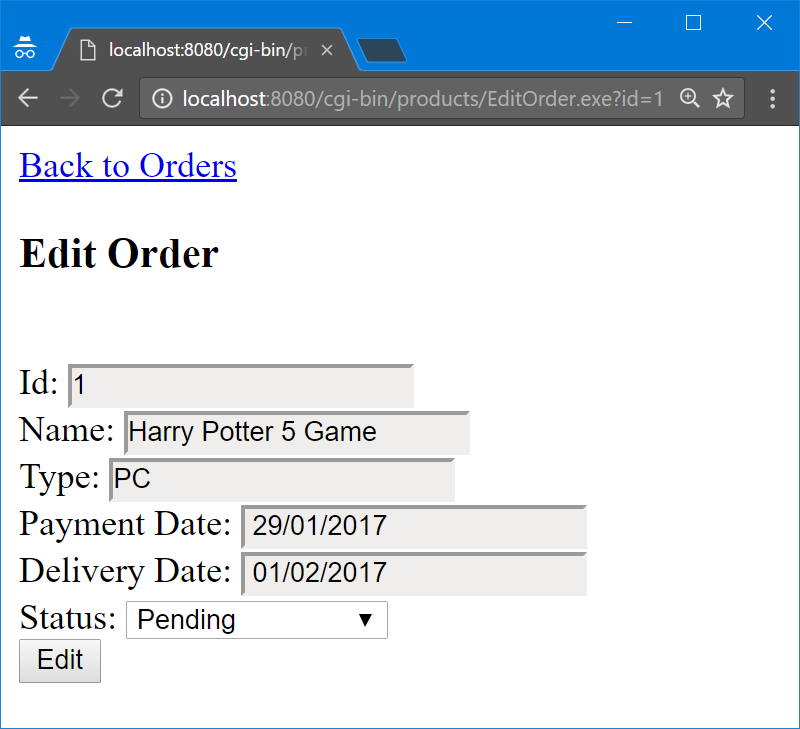
Create a C# application that prints the table within the following format:

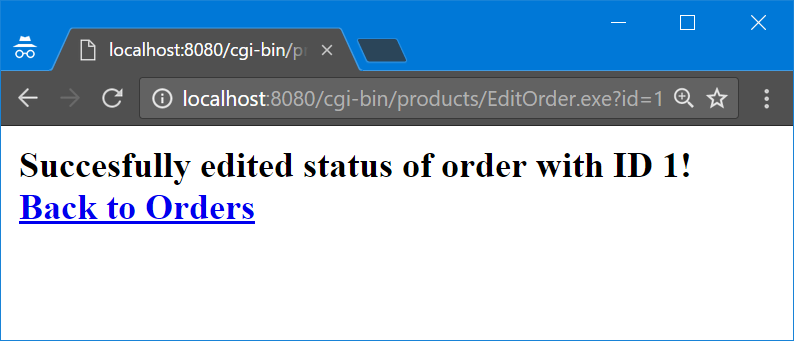


In the table of all orders the row’s background should be according to their Order Status:

* **Pending** – gray (active)
* **Delivered** – green (success)
* **Declined** – red (danger)
* **In Call to Confirm** – yellow (warning)

When the Edit link on some row is selected. A Form should appear containing all information for the order of but allowing user to **modify** **only its status**. The other fields should be read only.



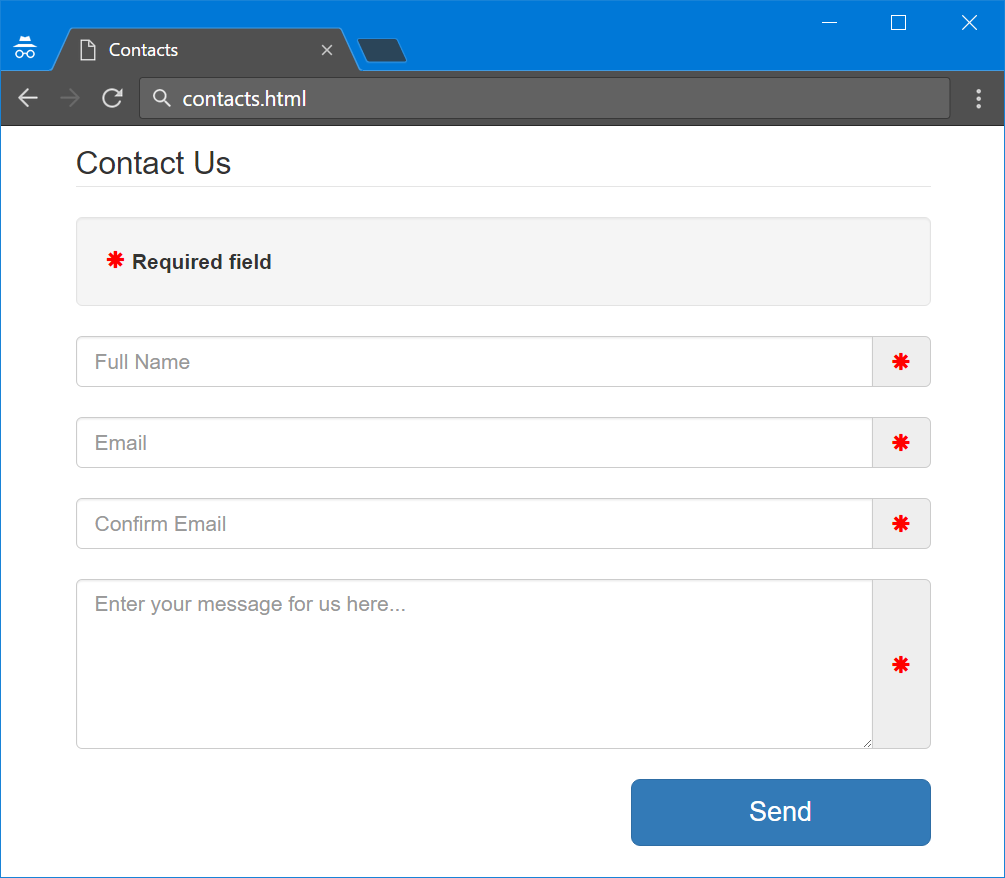


### \*Bonus Tasks

* Make also **delivery date to be editable**
* Make the editing **form look prettier** with Bootstrap

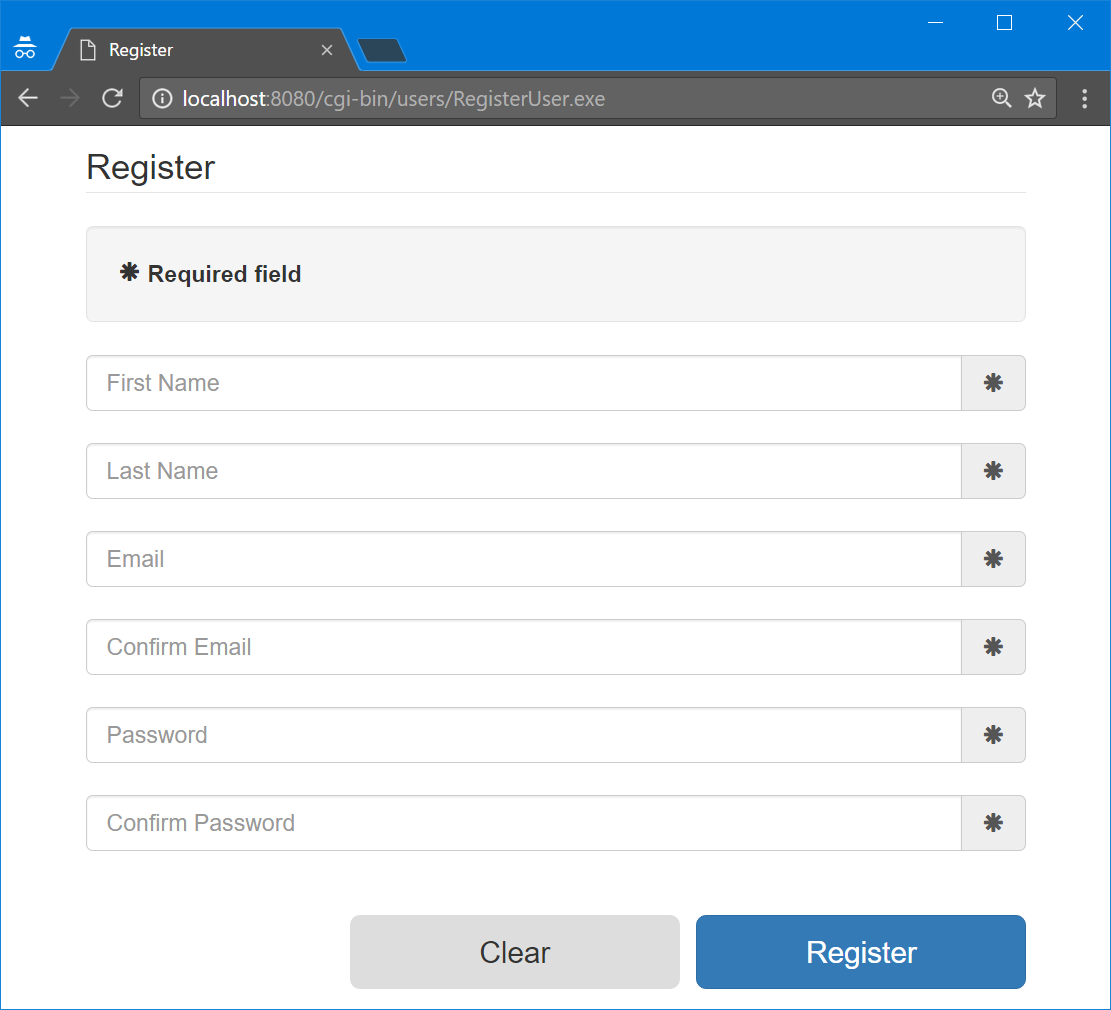
## Contacts Page

Using HTML and Bootstrap create a contact form like the one in the picture below.



## Register Page

Using html and bootstrap create a calculator that looks like the one in the picture below.



## \*Register Page Validations

Now let’s **validate the form and print informative messages** if any field is not filled or it is filled with wrong data. For example:

* **First/Last name is too long** – if the first/last name is **over** **30 symbols** long
* **First/Last name is too short** – if the first/last name is **under 2 symbols** long
* **Invalid email** – if the email is not in format **<user>@<host>.<domain>** (where **user**, **host** and **domain** can only be **alphanumeric characters**)
* **Emails does not match**
* **Invalid password** – if the password is below 4 symbols long and must contain at least 1 letter and 1 number
* **Passwords does not match**
* **Not all required fields are filled –** if some of required fields is not filled

Check the images below to get better understanding how the form should look like if the form is not valid after submission.

### Examples

