

ains:

Project Documentation
Project Code

CA 2 TEAM PROJECT - EAD

Documents

BEERME!

Julianna Bicz

X0014xxxx

James Cadden

X00143093

Team:

Julianna Bicz

James Cadden

Tomasz Kutela

Tomasz Kutela

X0014xxxx

Table of Contents

Project documentation.....	2
Github & Azure Links.....	2
Project Proposal.....	2
Database Schema.....	3
Screen Shots.....	4
Project Code.....	5
Models.....	11-12
Controllers.....	13-21
Views.....	22-25
Links to used images.....	25

Project Documentation

GitHub & Azure links

MVC Project on GitHub: https://github.com/jamesfcadden/APP_BEER_ME/tree/master/APP_BEER_ME

Azure Link: <https://appbeerme.azurewebsites.net>

Project Proposal

Our proposal is to create an app for people who enjoy a refreshing beverage but also demand the most bang for their buck.

Users will be able to query a database of beers and shops to investigate exactly how much alcohol they can purchase on a specified budget. The app will store data on craft beers and where they are available. The user will input an amount of money, e.g. €20, the app will interrogate the database and return exactly which combination of bottles or cans of beer that will provide the highest level of units of alcohol that the user can purchase, and return the amount of change due.

For the developers, the app will analyse beer consumption habits. The client app will point out, for example, most popular beer brands, types, and other habits. The app will have three tables: beers, shops and stock.

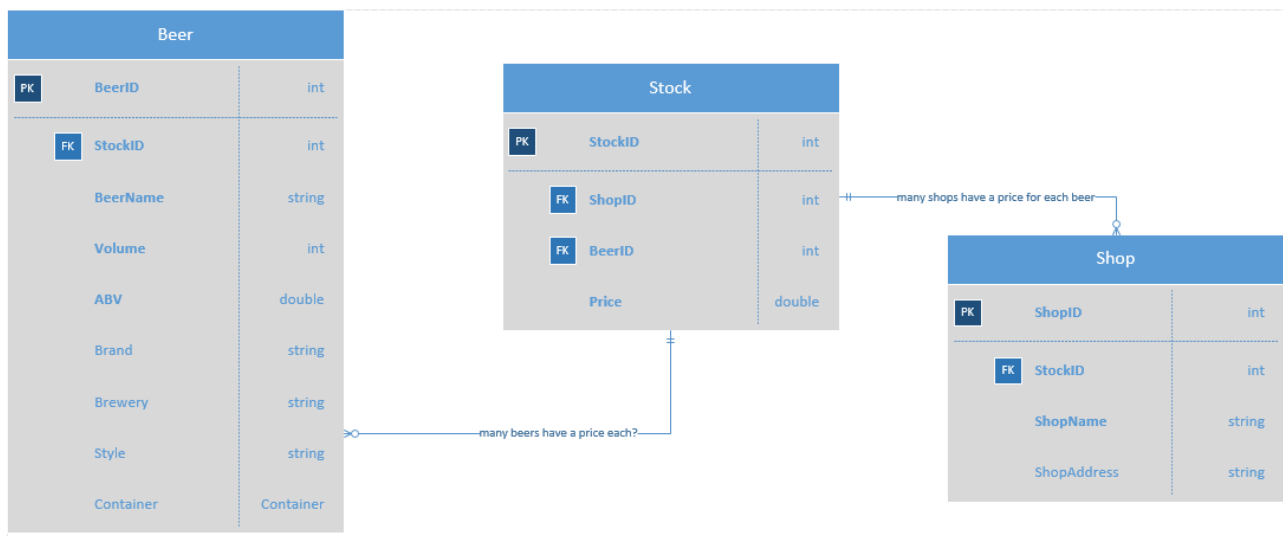
Database Schema

3 Tables.

Beer –Primary Key BeerID and Foreign Key StockID (From Stock table). Linked with Stock table one to many relationship

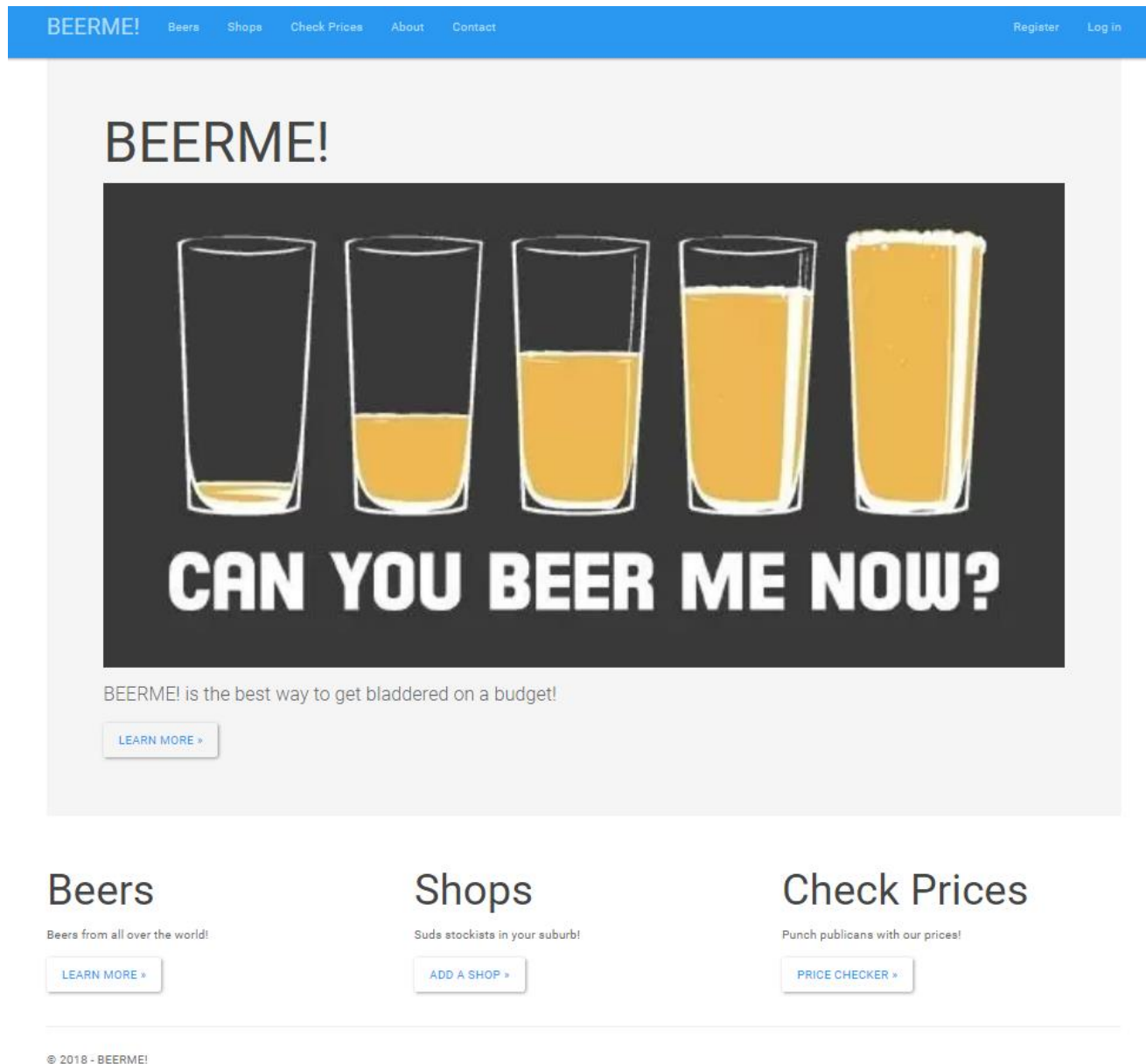
Stock – Primary Key StockID and two Foreign Keys – ShopID and BeerID. Linked with Beer and Shop tables many to one relationship

Shop - Primary Key Shop and Foreign Key StockID (From Stock table). Linked with Stock table one to many relationship



Screen Shots

Home Screen



Beers page

On this page, users can view the list of beers already on the database. They can also edit, view details for and delete a beer from the database. Beers are sorted by Name. A search bar is available for the user to search beer name, brewery or style. The user can also create a new beer.

BEERME!							Register	Log in
Beers								
Shops								
Check Prices								
About								
Contact								
Index								
Create New								
Find by name: <input type="text"/> <input type="button" value="Search"/>								
Name	Style	Brand	Container	Volume	ABV			
Boyne Brewhouse Amber Ale	Red Ale - American Amber / Red	Boyne Brewhouse	Bottle	500	4.8%	Edit	Details	Delete
Boyne Brewhouse American Pale Ale	Pale Ale - American	Boyne Brewhouse	Can	330	4.5%	Edit	Details	Delete
Boyne Brewhouse Imperial Stout Sherry Cask	Stout - Russian Imperial	Boyne Brewhouse	Bottle	500	10.8%	Edit	Details	Delete
Boyne Brewhouse Irish Craft IPA	IPA - American	Boyne Brewhouse	Bottle	500	6.8%	Edit	Details	Delete
Boyne Brewhouse Lager	Lager - Dortmunder / Export	Boyne Brewhouse	Bottle	500	4.8%	Edit	Details	Delete
Boyne Brewhouse Oatmeal Stout	Stout - Oatmeal	Boyne Brewhouse	Bottle	500	6.2%	Edit	Details	Delete
Boyne Brewhouse Pale Ale	Pale Ale - International	Boyne Brewhouse	Bottle	500	4.8%	Edit	Details	Delete
Boyne Brewhouse Saison	Saison / Farmhouse Ale	Boyne Brewhouse	Bottle	500	5.5%	Edit	Details	Delete
Boyne Brewhouse Session IPA	IPA - Session / India Session Ale	Boyne Brewhouse	Can	330	4%	Edit	Details	Delete
Boyne Brewhouse Vienna Lager	Lager - Vienna	Boyne Brewhouse	Can	330	5%	Edit	Details	Delete
Brown Bear Brown Ale	Brown Ale - English	Brown Bear	Bottle	500	4.5%	Edit	Details	Delete
Brown Bear Double IPA	IPA - Imperial / Double	Brown Bear	Bottle	500	6.3%	Edit	Details	Delete
Brown Bear IPA	IPA - English	Brown Bear	Bottle	500	5.2%	Edit	Details	Delete
Canadian	Lager beer	Canadian	Bottle	330	4%	Edit	Details	Delete
Darkside IPA	IPA - American	Brú Brewery	Bottle	500	5.2%	Edit	Details	Delete
Darragh's Session IPA	IPA - Session / India Session Ale	McGargles	Bottle	500	3.8%	Edit	Details	Delete

New Beer Details

Each field is mandatory.

Edit

Beer

Name Boyne Brewhouse Amber Ale

Style Red Ale - American Amber / Red

Brand Boyne Brewhouse

Brewery Boyne Brewhouse

Container Bottle

Volume 500

ABV 4.8

SAVE

[Back to List](#)

Check Prices Page

Index

[Create New](#)

Name	Shop Name	Price	Container	Units	Price per Unit	
O'Shea's IPA	Aldi	€1.89	Bottle	2.5	€0.76	Edit Details Delete
Rebel Red	Bradleys Off-Licence Cork City	€2.99	Can	1.4	€2.11	Edit Details Delete
Boyne Brewhouse Imperial Stout Sherry Cask	Centra	€3.99	Bottle	5.4	€0.74	Edit Details Delete
Darkside IPA	Centra	€5	Bottle	2.6	€1.92	Edit Details Delete
O'Hara's Irish Stout	Dunnes	€3	Bottle	2.2	€1.4	Edit Details Delete
Winter Star Spice Rye Pale Ale	Dunnes	€3	Bottle	2.5	€1.2	Edit Details Delete
Grafters IPA	Dunnes	€2.45	Bottle	3.3	€0.75	Edit Details Delete
Darkside IPA	Dunnes	€4.5	Bottle	2.6	€1.73	Edit Details Delete
Canadian	Lidl	€2	Bottle	1.3	€1.52	Edit Details Delete
The Crafty Brewing Company - Irish Red Ale	Lidl	€1.89	Bottle	2.1	€0.92	Edit Details Delete
The Crafty Brewing Company - Irish Pale Ale	Lidl	€1.89	Bottle	2.3	€0.84	Edit Details Delete
The Crafty Brewing Company - Irish Stout	Lidl	€1.89	Bottle	2.3	€0.84	Edit Details Delete
The Crafty Brewing Company - Irish Lager	Lidl	€1.89	Bottle	2.5	€0.76	Edit Details Delete
The Crafty Brewing Company - Irish IPA	Lidl	€2.35	Bottle	3	€0.78	Edit Details Delete

New Shop

Create Shop

Shop Name

Shop Address

CREATE

[Back to List](#)

Add new on Check Price page

BEERME!

BeersShopsCheck PricesAboutContact

RegisterLog in

Create

Stock

BeerIDDarkside IPA

ShopIDLidl

Price

Back to List

Lidl

Dunnes

Tesco Jervis

Centra

Tesco Tallaght

Aldi

Redmonds

Tesco Baggot St

Bradleys Off-Licence Cork City

© 2018 - BEERME!

BEERME!

BeersShopsCheck PricesAboutContact

RegisterLog in

Create

Stock

BeerIDDarkside IPA

ShopIDLidl

Price

Back to List

CREATE

© 2018 - BEERME!

About Us

We are three beer enthusiasts.

We are passionate about beer and we have got everything you might wish for!

- Julianna - "To beer or not to beer"
 - Hop House 13
 - Boyne Brewhouse Lager
 - Francis' Big Bangin' IPA
 - Brown Bear Brown Ale
- James - "It's all about the stout"
 - Darkside IPA
 - Newcastle Brown Ale
 - La Niña Barbuda
 - Guinness
 - Oyster Stout
 - Check out my [Untappd profile](#)
- Tomasz - "Beer to dine for"
 - O'Hara's Irish Red
 - Boyne Brewhouse Lager
 - Grafters' Pale Ale
 - Solas Hops and Grains Brown Porter

That's what we are, Beer Fans, or Brew Fans. Simple, to the point, but let's everyone know where we stand. Not only do we drink beer, we "cheer" for it, especially when it's tasty. Haven't we all done it? You ask what kind of beer there is at a particular establishment, expecting the same weak fare. Then we are told that there is a delicious microbrew or other tasty beer on tap, the only response is, "Yes!!"

Company Address

IT Tallaght Blessington Rd, Tallaght,
Dublin 24, D24 FKT9y

Support: Support@ittallaght.com

Marketing: Marketing@ittallaght.com

Contact Us!

Please fill in all required fields

Name

Email

Notes

SUBMIT

Project Code

Models

Beer Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace APP_BEER_ME.Models
{
    public enum Container { Bottle, Can }

    public class Beer
    {
        [Key]
        public int BeerID { get; set; }
        [Required(ErrorMessage = "The Name field is required !")]
        public string Name { get; set; }
        [Required(ErrorMessage = "The Style field is required !")]
        public string Style { get; set; }
        [Required(ErrorMessage = "The Brand field is required !")]
        public string Brand { get; set; }
        [Required(ErrorMessage = "The Brewery field is required !")]
        public string Brewery { get; set; }
        [Required(ErrorMessage = "The Container field is required !")]
        public Container Container { get; set; }
        public int Volume { get; set; }
        [Required(ErrorMessage = "The ABV field is required !")]
        public double ABV { get; set; }

        public virtual ICollection<Stock> Stocks { get; set; }
    }
}
```

Shop Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace APP_BEER_ME.Models
{
    public class Shop
    {
        [Key]
        public int ShopID { get; set; }
        [Required]
        [Display(Name = "Shop Name")]
        public string ShopName { get; set; }
        [Required]
        [Display(Name = "Shop Address")]
        public string ShopAddress { get; set; }

        public virtual ICollection<Stock> Stocks { get; set; }
    }
}
```

Stock Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace APP_BEER_ME.Models
{
    public class Stock
    {
        [Required]
        public int StockID { get; set; }
        public int BeerID { get; set; }
        public int ShopID { get; set; }
        [Required]
        public double Price { get; set; }

        public virtual Beer Beer { get; set; }
        public virtual Shop Shop { get; set; }
    }
}
```

Controllers

Beer Controller

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using APP_BEER_ME.DAL;
using APP_BEER_ME.Models;
using System.Data.Entity.Infrastructure;

namespace APP_BEER_ME.Controllers
{
    public class BeerController : Controller
    {
        private APP_BEER_MEContext db = new APP_BEER_MEContext();

        // GET: Beer
        public ActionResult Index(string sortOrder, string searchString)
        {
            ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";

            var beers = from b in db.Beers
                        select b;

            switch (sortOrder)
            {
                default:
                    beers = beers.OrderBy(s => s.Name);
                    break;
            }

            if (!String.IsNullOrEmpty(searchString))
            {
                beers = beers.Where(b => b.Name.Contains(searchString) ||
b.Style.Contains(searchString) || b.Brewery.Contains(searchString));
            }

            return View(beers.ToList());
        }

        // GET: Beer/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Beer beer = db.Beers.Find(id);
            if (beer == null)
            {
                return HttpNotFound();
            }
            return View(beer);
        }
    }
}
```

```

// GET: Beer/Create
public ActionResult Create()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "Name, Style, Brand, Brewery, Container,
Volume, ABV")]Beer beer)
{
    try
    {
        if (ModelState.IsValid)
        {
            db.Beers.Add(beer);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        catch (DataException /* dex */)
        {
            ModelState.AddModelError("", "Unable to save changes. Try again, and if the problem
persists see your system administrator.");
        }
        return View(beer);
    }
}

// GET: Beer/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Beer beer = db.Beers.Find(id);
    if (beer == null)
    {
        return HttpNotFound();
    }
    return View(beer);
}

// POST: Beer/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include =
"BeerID,Name,Style,Brand,Brewery,Container,Volume,ABV")] Beer beer)
{
    if (ModelState.IsValid)
    {
        db.Entry(beer).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(beer);
}

// GET: Beer/Delete/5
public ActionResult Delete(int? id, bool? saveChangesError = false)

```



```

{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    if (saveChangesError.GetValueOrDefault())
    {
        ViewBag.ErrorMessage = "Delete failed. Try again, and if the problem persists
see your system administrator.";
    }
    Beer beer = db.Beers.Find(id);
    if (beer == null)
    {
        return HttpNotFound();
    }
    return View(beer);
}

// POST: Beer/Delete/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Delete(int id)
{
    try
    {
        Beer beer = db.Beers.Find(id);
        db.Beers.Remove(beer);
        db.SaveChanges();
    }
    catch (DataException/* dex */)
    {
        //Log the error (uncomment dex variable name and add a line here to write a
log.
        return RedirectToAction("Delete", new { id = id, saveChangesError = true });
    }
    return RedirectToAction("Index");
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
}
}
}

```

Shop Controller

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using APP_BEER_ME.DAL;
using APP_BEER_ME.Models;
using System.Data.Entity.Infrastructure;

namespace APP_BEER_ME.Controllers
{
    public class ShopController : Controller
    {
        private APP_BEER_MEContext db = new APP_BEER_MEContext();

        // GET: Shop
        public ActionResult Index(string sortOrder)
        {
            ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "shopname_desc" : "";

            var shops = from s in db.Shops
                        select s;

            switch (sortOrder)
            {
                default:
                    shops = shops.OrderBy(s => s.ShopName);
                    break;
            }

            return View(shops.ToList());
        }

        // GET: Shop/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Shop shop = db.Shops.Find(id);
            if (shop == null)
            {
                return HttpNotFound();
            }
            return View(shop);
        }

        // GET: Shop/Create
        public ActionResult Create()
        {
            return View();
        }
    }
}
```

```

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "ShopName, ShopAddress")]Shop shop)
{
    try
    {
        if (ModelState.IsValid)
        {
            db.Shops.Add(shop);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
    catch (DataException /* dex */)
    {
        //Log the error (uncomment dex variable name and add a line here to write a
log.
        ModelState.AddModelError("", "Unable to save changes. Try again, and if the
problem persists see your system administrator.");
    }
    return View(shop);
}

// GET: Shop/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Shop shop = db.Shops.Find(id);
    if (shop == null)
    {
        return HttpNotFound();
    }
    return View(shop);
}

// POST: Shop/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "ShopID,ShopName,ShopAddress")] Shop shop)
{
    if (ModelState.IsValid)
    {
        db.Entry(shop).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(shop);
}

// GET: Shop/Delete/5
public ActionResult Delete(int? id, bool? saveChangesError = false)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    if (saveChangesError.GetValueOrDefault())
    {

```

```

        ViewBag.ErrorMessage = "Delete failed. Try again, and if the problem persists
see your system administrator.";
    }
    Shop shop = db.Shops.Find(id);
    if (shop == null)
    {
        return HttpNotFound();
    }
    return View(shop);
}

// POST: Shop/Delete/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Delete(int id)
{
    try
    {
        Shop shop = db.Shops.Find(id);
        db.Shops.Remove(shop);
        db.SaveChanges();
    }
    catch (DataException/* dex */)
    {
        return RedirectToAction("Delete", new { id = id, saveChangesError = true });
    }
    return RedirectToAction("Index");
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
}
}
}

```

Stock Controller

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using APP_BEER_ME.DAL;
using APP_BEER_ME.Models;

namespace APP_BEER_ME.Controllers
{
    public class StockController : Controller
    {
        private APP_BEER_MEContext db = new APP_BEER_MEContext();

        public double CalcUnits(double ABV, int Volume)
        {
            double Units = 0;
            Units = Volume * (ABV / 100);
            return Units;
        }

        public double CalcPricePerUnit(double ABV, int Volume, double Price)
        {
            double PricePerUnit = 0;
            PricePerUnit = Price / (Volume * (ABV / 100));
            return PricePerUnit;
        }

        // GET: Stock
        public ActionResult Index(string sortOrder)
        {
            ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "shopname_desc" : "";

            var stocks = db.Stocks.Include(s => s.Beer).Include(s => s.Shop);

            switch (sortOrder)
            {
                default:
                    stocks = stocks.OrderBy(s => s.Shop.ShopName);
                    break;
            }

            return View(stocks.ToList());
        }

        // GET: Stock/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Stock stock = db.Stocks.Find(id);
            if (stock == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.NotFound);
            }
        }
    }
}
```

```

        return HttpNotFound();
    }
    return View(stock);
}

// GET: Stock/Create
public ActionResult Create()
{
    ViewBag.BeerID = new SelectList(db.Beers, "BeerID", "Name");
    ViewBag.ShopID = new SelectList(db.Shops, "ShopID", "ShopName");
    return View();
}

// POST: Stock/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "StockID,BeerID,ShopID,Price")] Stock
stock)
{
    try
    {
        if (ModelState.IsValid)
        {
            db.Stocks.Add(stock);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
    catch (DataException /* dex */)
    {
        ModelState.AddModelError("", "Unable to save changes. Try again, and if the
problem persists see your system administrator.");
    }

    ViewBag.BeerID = new SelectList(db.Beers, "BeerID", "Name", stock.BeerID);
    ViewBag.ShopID = new SelectList(db.Shops, "ShopID", "ShopName", stock.ShopID);
    return View(stock);
}

// GET: Stock/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Stock stock = db.Stocks.Find(id);
    if (stock == null)
    {
        return HttpNotFound();
    }
    ViewBag.BeerID = new SelectList(db.Beers, "BeerID", "Name", stock.BeerID);
    ViewBag.ShopID = new SelectList(db.Shops, "ShopID", "ShopName", stock.ShopID);
    return View(stock);
}

// POST: Stock/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "StockID,BeerID,ShopID,Price")] Stock stock)
{
    if (ModelState.IsValid)

```

```

        {
            db.Entry(stock).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        ViewBag.BeerID = new SelectList(db.Beers, "BeerID", "Name", stock.BeerID);
        ViewBag.ShopID = new SelectList(db.Shops, "ShopID", "ShopName", stock.ShopID);
        return View(stock);
    }

    // GET: Stock/Delete/5
    public ActionResult Delete(int? id, bool? saveChangesError = false)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        if (saveChangesError.GetValueOrDefault())
        {
            ViewBag.ErrorMessage = "Delete failed. Try again, and if the problem persists  
see your system administrator.";
        }
        Stock stock = db.Stocks.Find(id);
        if (stock == null)
        {
            return HttpNotFound();
        }
        return View(stock);
    }

    // POST: Stock/Delete/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Delete(int id)
    {
        try
        {
            Stock stock = db.Stocks.Find(id);
            db.Stocks.Remove(stock);
            db.SaveChanges();
        }
        catch (DataException/* dex */)
        {
            //Log the error (uncomment dex variable name and add a line here to write a  
log.
            return RedirectToAction("Delete", new { id = id, saveChangesError = true });
        }
        return RedirectToAction("Index");
    }

    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {
            db.Dispose();
        }
        base.Dispose(disposing);
    }
}

```

Views

View Index Beer

```
@model IEnumerable<APP_BEER_ME.Models.Beer>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

@using (Html.BeginForm())
{
    <p>
        Find by name: @Html.TextBox("SearchString")
        <input type="submit" value="Search" />
    </p>
}

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Style)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Brand)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Container)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Volume)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ABV)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Style)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Brand)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Container)
            </td>
        </tr>
    }
}
```



```

        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Volume)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ABV)%
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id = item.BeerID }) |
            @Html.ActionLink("Details", "Details", new { id = item.BeerID }) |
            @Html.ActionLink("Delete", "Delete", new { id = item.BeerID })
        </td>
    </tr>
}
</table>

```

View Create Shop

```

@model APP_BEER_ME.Models.Shop

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Shop</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.ShopName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ShopName, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ShopName, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ShopAddress, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ShopAddress, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ShopAddress, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">

```

```

        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

```

View Stock Index

```

@model IEnumerable<APP_BEER_ME.Models.Stock>

@{
    ViewBag.Title = "Check Prices";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Beer.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Shop.ShopName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Price)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Beer.Container)
        </th>
        <th>
            Units
        </th>
        <th>
            Price per Unit
        </th>
    </tr>

    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Beer.Name)
            </td>
            <td>

```

```

        @Html.DisplayFor(modelItem => item.Shop.ShopName)
    </td>
    <td>
        €@Html.DisplayFor(modelItem => item.Price)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Beer.Container)
    </td>
    <td>
        @Math.Round(((item.Beer.Volume * (item.Beer.ABV / 1000))), 1,
MidpointRounding.AwayFromZero)
    </td>
    <td>
        €@Math.Round(item.Price / ((item.Beer.Volume * (item.Beer.ABV / 1000))), 2,
MidpointRounding.AwayFromZero)
    </td>
    <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.StockID }) |
        @Html.ActionLink("Details", "Details", new { id = item.StockID }) |
        @Html.ActionLink("Delete", "Delete", new { id = item.StockID })
    </td>
</tr>
}
</table>



```

Links of used images

<https://www.bottleyourbrand.com/beer-label-funny-beer-me-4400>

<https://www.memecenter.com/fun/12997/Can-You-Beer-Me-Now>