Five Plus Project Proposal

Table of Contents

[Proposal 2](#_Toc403983603)

[High Level Plan 2](#_Toc403983604)

[Work Plan: Stage 1 3](#_Toc403983605)

[Status Tracking: 3](#_Toc403983606)

[Day One 3](#_Toc403983607)

[Day Two 3](#_Toc403983608)

[Five Plus Project Break: 4](#_Toc403983609)

[Day Three 4](#_Toc403983610)

[Five Plus Project Break 4](#_Toc403983611)

[Day Four 4](#_Toc403983612)

[Day Five 4](#_Toc403983613)

[Work Plan: Stage 2 4](#_Toc403983614)

# Proposal

Build a POC/Demonstrator Responsive Web Application called "**Adventure Bike Shop**".

I would need five dedicated days in October to build out as much of the following project.

Advantages for:

1. Personally and professionally
   1. Skills Update on Current Microsoft Technologies.
   2. Applying course studies by actually developing a complete application end to end.
2. American Express
   1. Having a functional prototype with a code base covering potential areas for new development or improvement
   2. Initially developed for .NET code could be replicated in Azure.
   3. Generation of discussion points for improvement to existing methodologies used in the .NET and Azure RA.

# High Level Plan

**The proposed project is a full application to be built out and demonstrate the following using current Microsoft technologies.**

1. Presentation layer Using
   1. MVC 5
   2. Bootstrap 3
   3. JQuery and JQueryUI
2. T4 Code Generator prototypes that build the following:
   1. Functional Domain Layer using (DDD) encompassing the:
      1. BLL Components
      2. Repository Components
      3. DAL Components
      4. MicroORM (Prototype)
   2. REST Services Layer using WEBAPI
3. Database - BikeShopDB
   1. designed current industry usage
   2. using Convention Over Configuration
   3. Active Record pattern
   4. Based on the MSBI AdventureWorks Database.

# Work Plan: Stage 1

Build base infrastructure:

1. **Database**
   1. Built prior to starting.
2. **Domain Layer**
   1. Use “CodeGen” framework that was built in class to generate:
      1. CodeGen – Domain Layer containing Custom MicroORM, Domain Objects, Repositories and DB Context.
      2. Test Domain layer code
3. **Presentation Layer**
   1. Build base Presentation layer
   2. Integrate “SimpleMemberShip” Provider and presentation code
   3. Build Standard Pages, About, Welcome, etc…
   4. Build Shopping Section
      1. CRUD and Detail Pages
      2. Main Shopping page - Most advanced page which is a Hybrid using the following technologies.
         1. SPA (Single Page Application) concepts
         2. JQUERY
         3. Bootstrap
         4. HTML5 History Browser Manipulation work

## Status Tracking:

Day One**:**

1. Domain Layer Built with ease as expected.
2. Test most of the objects using immediate window and some junk test code.
3. No unit tests. Have to figure out how to MOCK out with the ORM. No time for that allocated.
4. Started presentation layer.

Day Two**:**

1. Base MVC 5 project added to solution.
2. Main structure:
   1. Layout
   2. Menu System
   3. Home and Administration Areas Setup
   4. Home Controller and Routing
   5. Welcome Page
   6. About Page
   7. Contact Page
   8. Error Page and Error handing
      1. Installed and Tested error handling configuration
      2. Installed and “**Last Chance**” error handling and **Error** **logging to database**
   9. Administration Controller and Routing
   10. Started Authentication and Authorization setup thru **SimpleMemberShip**.
3. Glitch setting up “**SimpleMemberShip**” system Infrastructure and UI. Apparently Microsoft has phased it out for ASP.NET Identity with MVC 5 and VS2013 Update#3 that I was unaware of.

Five Plus Project Break: Had NuGet and normal work to do. Plus I need to research “**SimpleMemberShip**” issues with MVC 5.

Research Update: **\*\*\*** Microsoft is moving to **ASP.NET Identity** System which is very complex that replaces and deprecates the “**SimpleMemberShip**” system which was released with MVC 4 and VS2010 SP1 and VS2012 updates. Apparently “**SimpleMemberShip**” was just part of the EARLY work for ASP.NET Identity. Simple was good and worked well but looks like it required SQL Server and no one wants a pure SQL Server based Identity system is what I’m learning.

Research Update: **ASP.NET Identity** is extremely different current database at best will need to be modified for “Five Plus Project” no time to full rewrite. Will need to establish a second database for Identity and integrate (**Kludge together until I can revisit at another time**). Database will at some point require full integration of the new **ASP.NET Identity** tables. ***Note that ASP.NET Identity is designed to be extended to other databases like MySQL, Oracle, etc...***

Research Update: **ASP.NET Identity** is design to do it all!! It even has social integration with Facebook, Google, Two Factor Auth, and more. Also a lot of stuff is now **OWIN** based with MVC 5!!

Day Three**:**

1. Make minimum modifications to **BikeShopDB** User table for KLUDGE to integrate with a separate Identity Database outside the current domain model. New connection to the new “**AdvenBikeShopIdentity**” Database added.
2. Rebuilt and tested Domain Layer. It was a great test for “**CodeGen**” as it was designed to be able to handle changes to database schema and update code.  **☺ Major Time Saver.**
3. **Wow** so far built out four controllers for **ASP.NET Identity** system. Brain frying experience. Good learning experience though.
4. Much learning and debugging while figuring out how parts go together and integrating into current design. Brutal but great learning experience to be sure.
5. Added **UserSync** and **CurrentUser** static classes.
6. View Routing in Admin stuff having issues with all things “**SIGNING OUT**”. I can’t get it to log out of Identity system. Imagine that. Also figuring out why some stuff does work cleanly.
7. Finally solved routing and sign out issues. This was tied to the Routing actually. ASP.NET Identity could have been its **OWN** Five Plus Project.

Five Plus Project Break: Returned back to normal work days. I need to schedule two more days.

Research Update: First thing to do next session is update Product table for categories needed for pagination, rebuild domain and test. Add pagination classes from toolkit to modify for project. Add menu for admin dropdown and authorizations. Add Shop Area, Controller and develop base View Models. Still need to add stuff to find and update User table in **BikeShopDb** with **AdvenBikeShopIdentity** databases.

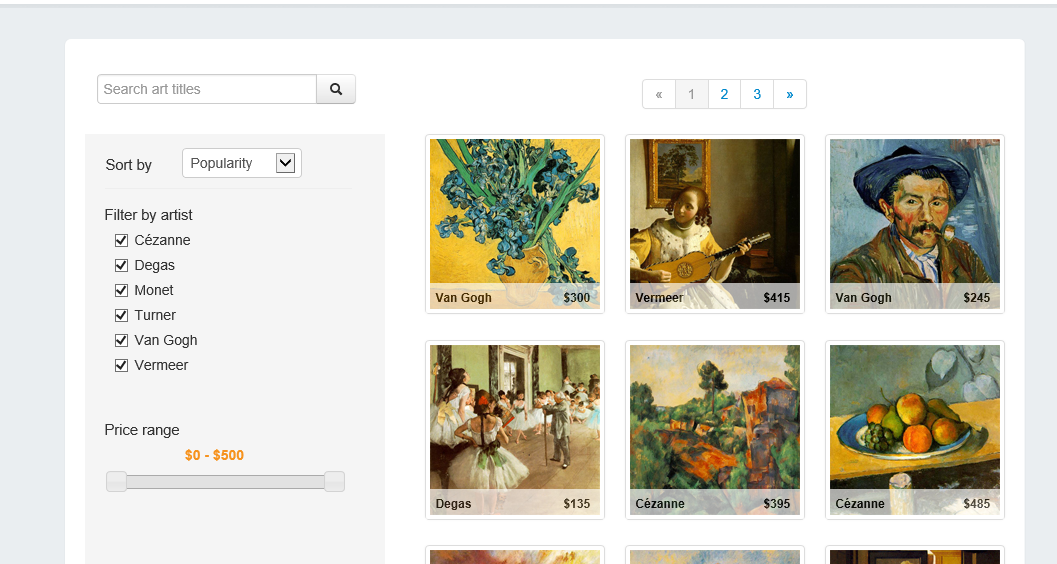
Day Four**:**

1. Started on Shop Area, Product Controller, Pagination and two modifications to BikeShopDB for Categories. Started on Products and \_Products views. Menu modified for main Shop section. Domain layer regenerated after multiple database tweaks. Added base for BikeShopJS library in Scripts section needed for various Shopping page (Products.cshtml).

Research Update: Needed to brush up on JQuery in between day four and five as had some issues working on BikeShipJS code.

Day Five**:**

1. Still building out Models, Pagination and the base model to transfer data to view.
2. Target is to build out a basic pages (almost a SPA) that dynamically filters by Selectable list of Catalogs or Vendors (AKA Artist below).
3. Items ae displayed SIMILAR to below but using Bootstrap responsive images with captions.
4. Clicking with take customer to a product page.



# Work Plan: Stage 2

Build Out Administration Infrastructure:

1. Build out user administration features
2. Reporting
3. Dashboard
4. Ad Hoc Reporting