|  |
| --- |
| Mercer university |
| Project 2 |
| SSE 657 - Object Oriented Project Methods |
|  |
| **Tanya Do & John Robison** |
| **10/27/2014** |

|  |
| --- |
|  |

Table of Contents

[Introduction 1](#_Toc402022775)

[Problem Statement 2](#_Toc402022776)

[Discussion With Customer 2](#_Toc402022777)

[Commonality and Variability Analysis of System 2](#_Toc402022778)

[Features List 3](#_Toc402022779)

[Domain Analysis 4](#_Toc402022780)

[Use Case Diagrams 4](#_Toc402022781)

[Design Pattern Selection 5](#_Toc402022782)

[Determining the Architectural Significance of Components 7](#_Toc402022783)

[Risk Analysis and Reduction 7](#_Toc402022784)

[Use of Design Principles 7](#_Toc402022785)

[Mock Scenarios of System Interaction 7](#_Toc402022786)

[Appendix A: References 8](#_Toc402022787)

**Table of Figures**

[Figure 1: MakerStore Use Case Diagram 5](#_Toc402022788)

[Figure 2: MVC Interaction Diagram 6](#_Toc402022789)

# Introduction

This report and the project that it corresponds to are intended to display mastery of the concepts contained in Chapters 6 - 8 of the book Head First Object-Oriented Analysis and Design by McLaughlin, Pollice, and West. These chapters build on the previous chapters by explaining how those Object-Oriented Analysis and Design (OOAD) concepts can be applied to larger, real-world problems to make them more manageable to design.

The first step offered by the text to make a large problem less intimidating to approach is to break the project into smaller, logical pieces that can be developed using the OOAD concepts provided in the previous chapters. To determine these logical pieces, techniques such as conversations with the customer and commonality and variability analysis can be used to determine the features of the system. Once the features are determined, the most architecturally significant features are determined using the three Q's of architecture and are then designed first to reduce risk. After the architectural design of the system is completed, design principles are applied to ensure that the system is not rigid, fragile, or immutable.

# Problem Statement

With the opening of SparkMacon, Macon's own Maker Space, there is now a need for a web service that will allow makers of all trades and skill sets to advertise their products and promote their business. This service will allow makers to create a profile that contains: a biography describing their craft, previously completed or sold projects, products currently for sale, user reviews of the maker's products, and a place for users to request unlisted or new products. The people behind SparkMacon are devoted to building a strong maker community in the Macon area and want a system that helps to push their makers further. To build this community that SparkMacon aspires to, the web service will need features that promote community involvement: featured makers (i.e. Maker of the Week), community forums, the ability to like or follow products and users, etc.

# Discussion With Customer

# Commonality and Variability Analysis of System

In order to ensure that this web service meets all of the customer's expectations, our team performed a commonality and variability analysis on the system. This means that we compared the proposed systems to preexisting web services and described what these systems offered or did that the customer either wanted their system to be like or not be like. In addition to ensuring that the designed system meets the customer's expectations, this commonality and variability analysis will aid our team in determining the major features of the system which will allow us to determine the most architecturally significant components.

Based on the discussion with the customer, the core functionality of the web service should be much like Etsy, an e-commerce marketplace for creators of unique items to sell their goods. The customer stressed that they wanted posted items to be actually made by the users that posted the listings, so the web service should not be like Amazon or Ebay. Since the customer is interested in strengthening the Macon community, the system should not be international or even open to anyone like Etsy, Amazon, and Ebay are. The area-based product listing system used by web services such as Craigslist is ideal for building this strengthened community of Macon makers.

# Features List

After talking to the customer and doing a commonality and variability analysis of the system, our team compiled a list of features that the system will support. Listed below are the features that were determined as well as the requirements that make up each feature:

* User Accounts
  + User biography
  + List of products available
  + List of previously sold products
  + Product Requests
  + User Rating/ Review
* Product Listings
  + Product description
  + Product Category
  + Price
  + Product Rating/ Review
* Home Page
  + Maker of the Week
  + Featured Products
  + News
  + About
  + Link to Spark Macon
* Community Involvement
  + Community forum
  + Event coordination/ planning
  + Liking or following products or users

The User Account feature will be responsible for storing all of a user's information: biography, products, user rating, and requests. A user will be able to create an account at any time and begin selling and marketing their products. Once an account is created, the user will be able to login to the system and edit/ update their account information as well as create postings for products.

The Product Listing will contain all of the information for a user's product: product description, category, price, and product reviews. A user with a registered account will be able to post listings for their products. The use of the description and category will allow other users to search for products that interest them, such as robotics or art.

The Home Page will be the main access point of the web service. It will be used to display important pieces of information, such as the featured Maker of the Week, upcoming events, and news about the makerspace. This page will also contain information about what SparkMacon's mission is as well as provide links to their website.

There will be several components that will aim to build community involvement, including user forums, announcing local makerspace events, and the ability to like or follow other users' accounts and products.

# Domain Analysis

# Use Case Diagrams

The users should be able to interact with the system in many different ways. These interactions include:

* Creating and editing accounts
* Creating and editing product listings
* Purchasing products
* Contributing on forums
* Navigating to SparkMacon's website
* Rating users and products.

The account interactions will be handled through registration and log in processes. A new user will have to register with the system in order to have a username and password that they will be able to log in with. Once they are registered, they will be given the ability to edit their account information, such as their biography and available products.

Only registered users will be able to create and edit product listings. Once a user is registered, they will be able to post as many products as they have to sell. Once a product is listed, only the user who posted it will be able to make edits to the listing. These edits include the product's description and price.

Registered users will also be able to contribute to the community by rating other users and their products as well as commenting on forums. If a user is interested in learning more about the SparkMacon makerspace or learning about upcoming events, they will be able to click a link that navigates to the SparkMacon site regardless of if the user is registered.

Figure 1 below visualizes the users' interactions with the system in a UML Use Case Diagram.

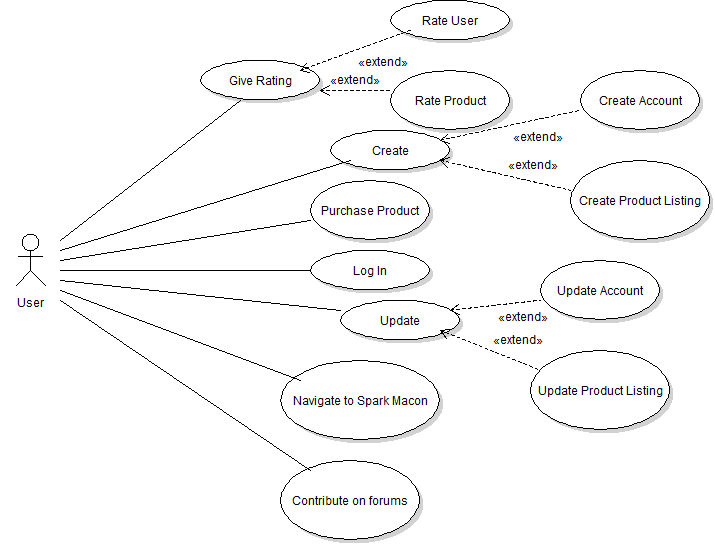


Figure : MakerStore Use Case Diagram

# Design Pattern Selection

The designed system will be built on the ASP.NET MVC Framework. The use of this framework utilizes the MVC (Model-View-Controller) design pattern which encourages designs to be loosely coupled through a separation of concerns. This separation is achieved by using the three main components: Models which handle the data and logic of the system, Views which display the data in a way that makes sense to the user, and Controllers which are responsible for dealing with user inputs by updating the models and view. Below in Figure 2 is a diagram that visualizes the interactions between these components.



Figure : MVC Interaction Diagram

The ASP.NET framework builds on the MVC design pattern by relying on a "convention over configuration" approach which reduces the amount of code required for a project. This approach requires certain design decisions, such that items are placed in the correct locations (i.e. Controllers must be in the Controller folder) and that they are named appropriately (i.e. Controllers will be named ending with "Controller"). Not only does this convention based approach reduce code requirements, it also aids in overall readability of the system.

Due to the loose coupling offered by the MVC design pattern, our team chose ASP.NET MVC as the framework for this system because it allowed for code reuse and parallel development. Since the models, views, and controllers are separated, they can easily be implemented into another application. An example of this would be if the customer requested a mobile application since the model and controller classes could be reused. The only major change in this example is that new views would have to be created specifically for the mobile platform. This separation also allows components to be designed and built separately. One developer can build the store user interface in the views while another developer designs the underlying business logic in the models.

In addition to all of the benefits of MVC's separation of concerns listed above, our team decided to develop with the ASP.NET MVC Framework for its use of .NET languages (C# and Visual Basic) which we are familiar with as well as the use of its easy to learn view engine, Razor. C#, our .NET language of choice, is extremely powerful and easy to develop with due to its number of available libraries and the use of Microsoft's Intellisense, which can auto-fill code for the developer such as variable and method names. The Razor view engine is a combination of HTML and either C# or Visual Basic, which makes it very natural for .NET developers to write and learn. The use of C# in the markup allows for very powerful dynamic web pages. Razor also allows the use of layouts which enables developers to have a single Razor file act as a template for all other views. These layouts reduce duplicate code by encapsulating common view elements into the layout. The final advantage of Razor that will aid in development is the ability to use Intellisense to quickly write the markup and code contained in the Razor files.

# Determining the Architectural Significance of Components

# Risk Analysis and Reduction

# Use of Design Principles

# Mock Scenarios of System Interaction

# Appendix A: References

* McLaughlin, Brett, Gary Pollice, and David West. *Head First Object-oriented Analysis and Design*. Sebastopol, CA: O'Reilly, 2007. Print.
* "Easy Intro to ASP.NET MVC." *Easy Intro to ASP.NET MVC*. N.p., n.d. Web. 21 Oct. 2014. <http://www.beansoftware.com/ASP.NET-Tutorials/Intro-ASP.NET-MVC.aspx>.