Running Head: *SimpleStore Research Proposal*

*A Best Practices Approach to MVC-based Web Application Architecture*

*Patrick Brossard*

Regis University

**A Best Practices Approach to MVC-based Web Application Architecture**

There are many different approaches to designing web applications, and not all of these approaches are created equal. One such architecture is dubbed “Model-View-Controller”, hereafter MVC, and is a popular way of building in formal layers of abstraction into an application, essentially separating the code responsible for the presentation, logic, persistence tiers.

However, even within the MVC umbrella, there are a wide variety of implementation considerations and architectures, each with different advantages and disadvantages. While I realize there can be no single best architecture given the widely differing requirements of today’s web applications, I propose to research and design an optimal architecture given the requirements of the SimpleStore project.

**Background**

At a high level approach, I have chosen Domain-drive Design (DDD) as my software development methodology. However, DDD encompasses an entire project, from requirements definition through delivery iterations. Thus, for the purposes of this research proposal, I am limiting my scope to the actual software architecture of the project.

Current software design methodology emphasizes the identification and prescriptive use of common *Design Patterns*. A design pattern is a “general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.” (Design Patterns, 2010) DDD is at heart an object-oriented design methodology, as object oriented programming is still our best way of expressing real-world models in code. As such, DDD makes heavy use of design patterns.

**Capstone Approach / Methodology**

In addition to specific design patterns, there are more abstract programming principles that are useful to keep in mind when creating a framework. Design goals such as loose coupling, dependency injection, unit test coverage, and persistence independence can all be met through a carefully designed architecture. Of course, each level of abstraction added increases the complexity of the solution; and with a relatively simple domain, I will quickly reach a point of diminishing returns when adding flexibility. In the interests of highlighting the design decisions made, I will document which design goals and patterns I deemed worthwhile, which were discarded, and why.

The research performed will take three approaches:

1. Identifying the technologies which will be used, at a high level: languages, platforms, and tools. These decisions will have deep impacts on the rest of the project.
2. Researching the different architectural design patterns and principles, why they are desirable, what trade-offs they imply, and identifying which would be most important for this project.
3. Downloading and analyzing some of the various sample applications available on the Internet and identifying which architectural elements mesh well with my needs.

**Time Table / Schedule**

|  |  |
| --- | --- |
| **Week** | **Tasks** |
| 1 | * Concept Draft |
| 2 | * Research Proposal * Stakeholder Letter of Agreement * Complete technology selection |
| 3 | * Complete Requirements documentation * Complete Hosting and Network documentation * Begin work on Software Architecture, research paper * Status Report |
| 4 | * Complete Database Schema * Continue work on Software Architecture, research paper * Begin work on UI Wireframes * Status Report |
| 5 | * Continue work on Software Architecture, research paper * Continue work on UI Wireframes * Begin work on actual code prototyping * Status Report |
| 6 | * Complete Software Architecture, research paper * Complete UI Wireframes * Continue code prototyping * Begin organizing the complete product paper, presentation * Status Report |
| 7 | * Complete code prototyping * Finalize Presentation * Write Reflection paper * Status Report |
| 8 | * Presentation to stakeholders * Finalize Project documentation * Submit all deliverables |

**References**

Design Patterns. (n.d.). In *Wikipedia*. Retrieved March 18, 2010, from <http://en.wikipedia.org/wiki/Design_pattern_(computer_science)>

Domain-drive Design. (n.d.). In *Wikipedia*. Retrieved March 18, 2010, from <http://en.wikipedia.org/wiki/Domain-driven_design>

Abel Avram, Floyd Marinescu. (2007). *Domain-driven Design, Quickly*. C4Media.