Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 17.03.2016. | 1.0 | <details> | Czako Zoltan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 5

3. Design Constraints 5

# Introduction

In this document we will describe and motivate with good arguments every design decision that we made, we will talk about non-functional requirements as availability, performance, security, testability and usability. For each non-functional element we will explain our solution and the reason of our decision. We will talk about the technologies that we will use to create this project, about design patterns that we use and we will present some very useful framework.

# Non-functional Requirements

## Availability

For a high degree of availability we will use the Azure Cloud to store our application data, because Azure is created and maintained by Microsoft which is a very reliable company, the probability of failure with this Cloud is very low, so our data will be available anytime.

## Performance

We want to build a very fast and clean system. Because this site will contain lots of data about users, and products and reading data from database needs a lots of time we will choose to use lazy load and bring data from database only when is really needed. Besides that we will use LINQ expressions for every database query because these expressions are highly optimized, fast and clean. In the UI we will use partial views and Ajax calls to bring data as fast as possible and we will use pagination because than every time we will also have to bring a small amount of data and we can obtain a fast working site.

## Security

Talking about security, we will use basic authentification with username and password, we will encrypt the passwords so that these password will be unreadable for a hacker. For payment we will use PayPal or CreditCard with secured http request (https).

## Testability

Test are very important in maintenance so we will test every service and every method which contains logics. We will write not just happy-flow test but even corner case test too. To create test methods we will use Unit tests

## Usability

Our project will be reusable, because we will use the MVC Architectural Patter which and we will separate every layer, every modul of our project with interfaces so that any layer can be replaced anytime.

# Design Constraints

For the architecture of our site we decided to choose the MVC architectural Pattern, because it has a lot of benefits, for example:

1. **Separation of concerns:**  
   a. The separation the three components, allows the re-use of the business logic across applications.  
   b. Multiple User Interfaces can be developed without concerning the codebase.
2. **Developer specialization and focus:**  
   a. The developers of UI can focus exclusively on the UI screens without bogged down with business logic.  
   b. The developer of Model / business can focus exclusively on the business logic implementations, modifications, updations without concerning the look and feel and it has nothing to with business logic.
3. **Parallel development by separate teams:**  
   a. Business logic developers can build the classes, while the UI developers can involve in designing UI screens simultaneously, resulting the interdependency issues and time conservation.  
   b. UI updations can be made without slowing down the business logic process.  
   c. Business logic rules changes are very less that needs the revision / updations of the UI.

For implementing the website’s back-end we will use C#. To easily and clearly access the data of our database we will use the Entity Framework which will help us creating the connection with the database and will help us in the CRUD operations. For mapping the data between Models and ViewModels we will use the AutoMapper framework.

The front-end of our project will be implemented using HTML, CSS and JQuery and for a fast datacontrol we will use Ajax calls to bring data to the UI.

Because the clients can change their minds in 30 days we will use the Order Design Pattern to save the track the orders of the clients and to undo orders if needed. To a clear data manipulation we will use the Generic Repository Design Pattern together with the Generic Unit Of Work Pattern and we will represent the tables of the database as Sigletones.