|  |
| --- |
|  |
| Capstone Project Document |

**DANDELION**

Report #2 – Architecture Design

|  |  |  |
| --- | --- | --- |
| **Dandelion** | | |
| **Group Members** | Vũ Công Chính | SE02585 |
| Lưu Ngọc Mạnh | SE02619 |
| Nguyễn Minh Huy | SE02723 |
| Dương Đức Anh | SE02700 |
| Vũ Ngọc Trung | SE02967 |
| Cao Thị Phương Mai | SE02908 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Project code** | DDL | |

**- Hanoi, 09/2015 -**

# SIGNATURE PAGE

AUTHOR: Lưu Ngọc Mạnh 15/09/2015

Project Manager

REVIEWERS: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 16/09/2015

Team member

APPROVAL: Nguyễn Văn Sang --/--/--15

Supervisor

Record of change

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Effective Date | Changed Item | A,M,D | Change Description | Reason for Change | Rev. Number |
| 19/Sep/2015 | Create Project Plan | A | First version | Create Architecture Design | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE OF CONTENTS

[**SIGNATURE PAGE** 1](#_Toc428304837)

[**DEFINITIONS AND ACRONYMS** 4](#_Toc428304838)

[**1** **PROJECT OVERVIEW** 5](#_Toc428304839)

[1.1 Project Description 5](#_Toc428304840)

[1.2 Scope and Purpose 5](#_Toc428304841)

[1.2.1 Purpose of Project 5](#_Toc428304842)

[1.2.2 Scope of Project 5](#_Toc428304843)

[1.2.3 The functions of Project 5](#_Toc428304844)

[1.3 Assumptions and Constraints 7](#_Toc428304845)

[1.4 Project Objectives 8](#_Toc428304846)

[1.4.1 Standard Objectives 8](#_Toc428304847)

[1.4.2 Specific Objectives 8](#_Toc428304848)

[1.5 Critical Dependencies 9](#_Toc428304849)

[1.6 Project Risk 9](#_Toc428304850)

[**2 PROJECT DEVELOPMENT APPROACH** 10](#_Toc428304851)

[2.1 Project Process 10](#_Toc428304852)

[2.1.1 FPT Software Process Model 10](#_Toc428304853)

[2.1.2 Project Life Cycle 11](#_Toc428304854)

[2.2 Requirement Change Management 12](#_Toc428304855)

[2.3 Quality Management 12](#_Toc428304856)

[2.3.1 Defect Prevention Strategy 12](#_Toc428304857)

[2.3.2 Review Strategy 13](#_Toc428304858)

[2.3.3 Unit Testing Strategy 13](#_Toc428304859)

[2.3.4 Integration Testing 14](#_Toc428304860)

[2.3.5 System Testing 14](#_Toc428304861)

[2.3.6 Estimates of Defects to be detected 14](#_Toc428304862)

[2.3.7 Measurements Program 15](#_Toc428304863)

[**3 ESTIMATION** 15](#_Toc428304864)

[3.1 Size 15](#_Toc428304865)

[3.2 Effort 15](#_Toc428304866)

[3.3 Schedule 15](#_Toc428304867)

[3.3.1 Project Milestone & Deliverables 15](#_Toc428304868)

[3.3.2 Activity Schedule 17](#_Toc428304869)

[3.4 Resource 17](#_Toc428304870)

[3.5 Infrastructure 18](#_Toc428304871)

[3.6 Training Plan 19](#_Toc428304872)

[3.7 Finance 19](#_Toc428304873)

[**4 PROJECT ORGANIZATION** 20](#_Toc428304874)

[4.1 Organization Structure 20](#_Toc428304875)

[4.2 Project Team 20](#_Toc428304876)

[4.3 External Interfaces 23](#_Toc428304877)

[4.3.1 FPT Software Interfaces 23](#_Toc428304878)

[4.3.2 FPT University’s Interfaces 23](#_Toc428304879)

[**5 COMMUNICATION & REPORTING** 24](#_Toc428304880)

[**6 CONFIGURATION MANAGEMENT** 25](#_Toc428304881)

# PROJECT OVERVIEW

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

## Scope

The scope of this document is to depict the architecture of the Dandelion website created by DDL capstone project team.

## Definitions, Acronyms, Abbreviations

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Note |
| DDL | Dandelion |  |
| MVC | Model view control |  |
| IDE | Integrated development environment |  |
| Q&A | Question and answer |  |
| GUI | Graphic user interface |  |

**Table 2-1:** Definitions and acronyms

## References

* DDL\_ Software Requirements Specification\_v1.0\_EN.docx
* DDL\_Data Design\_v1.0\_EN.docx
* Software Architecture Design Illuminated Book

## Overview

The Software Architecture Document contains the following subsections:

* **Section 1**: Provide an overview of entire Software Architecture Document.
* **Section 2**: Choice of Architecture Design
* **Section 3**: Architectural Representation
* **Section 4**: Architectural Goals and Constraints
* **Section 5**: Use-Case view
* **Section 6**: Logical View
* **Section 7**: Process View
* **Section 8**: Deployment view
* **Section 9**: Quality

# CHOICE OF ARCHITECTURE DESIGN

## MVC Model

The purpose of DDL is developing as a crowdfunding website, where people can start their ideas and make it real. The system of FAP is structured based on MVC combined with layered architecture.

### MVC Model Overview

The **model-view-controller or MVC** is software architecture commonly used for creating web applications or software. In other words, it's a structure for web applications to follow in order to ensure efficiency and consistency. Many of the most popular frameworks use the MVC architecture, including ASP.NET, CodeIgniter, Zend, Django, and Ruby on Rails. At the same time, there are many web developers who don't use a coding framework yet still set up their applications to follow the MVC structure.

The Model-View-Controller (MVC) design pattern assigns objects in an application one of  
three roles: model, view, or controller. The pattern defines not only the roles objects play in  
the application, it defines the way objects communicate with each other. Each of the three  
types of objects is separated from the others by abstract boundaries and communicates with  
objects of the other types across those boundaries. The collection of objects of a certain MVC  
type in an application is sometimes referred to as a layer—for example, model layer.



Figure 2‑1: MVC Model

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them:

* **Controller:** The controller can be considered the "middle man" of the application. It works with the user, taking in data, and then working with the model to get the appropriate data or calculation, and then working with the view to show the response to the user.
* **Model:** A model is simply a representation of something we need to deal within our application. It is a "model" for something we must represent in code, such as a book, user, bank account, or whatever. The model is responsible for holding the functions and variables that are involved with whatever it's representing. You can think of a model's logic as the core concept to object oriented programming — models are just our "classes". However, don't let this confuse you as controllers are technically structured as classes as well.
* **View:** Finally, after the controller requests information from the model it sends it to a view. A view is just like the application's templating system — there might be a view for a certain type of page layout *(profile page)*, a mobile view, or a view for a particular theme/skin. A view will contain all of the markup, CSS, and etc. that you traditionally use with creating a static web page.

### Advantages and disadvantages of MVC Model

* Advantages:
* MVC separates system into components, which can be developed, maintained and upgraded individually without pausing system.
* Develop tools is useful and easy to use.
* Large of documentary sources.
* Disadvantages:
* For small projects that apply MVC model caused cumbersome, time consuming in development process.
* Time consuming to transits data between components.
* Not suitable for agent-oriented applications such as interactive mobile and robotics applications.
* Multiple pairs of controllers and views based on the same data model make data model change expensive.
* The division between the View and the Controller is not clear in some cases.

### The reasons for choosing MVC Model

* MVC makes parts of system can be developed individually and simultaneously to reduce developing time.
* Better support for test-driven development.
* Tools is useful and documentary source is large makes MVC is easy to develop.
* DDL system is not complete system, now. We built the system that towards extensibility and maintainability in the future.

## .Net Framework

### .Net Framework Overview

.NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages.

The .NET Framework is a technology that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

### Advantages and disadvantages of .Net Framework

* Advantages
* Less time to produce product
* Less Complexity.
* Easily to access complex O.S functions
* Easily to build Data Oriented Project, support huge DB functions.
* Managed
* Support Both Windows and Web Application.
* Easy to create Dynamic sites.
* Disadvantages
* Not suitable for High End Application
* Low performance compare to C, C++.
* Unavailability of build in methods.
* .NET framework is free to download but Code Editor is costly.
* Only few O.S supports .NET.

### The reasons for choosing .Net Framework

* Consistent with DDL system.
* There are many plugins and resources which support creating a website using MVC model.
* Many members can use and have experiences using C# (a language of .Net Framework)

## AngularJS

### AngularJS Overview

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

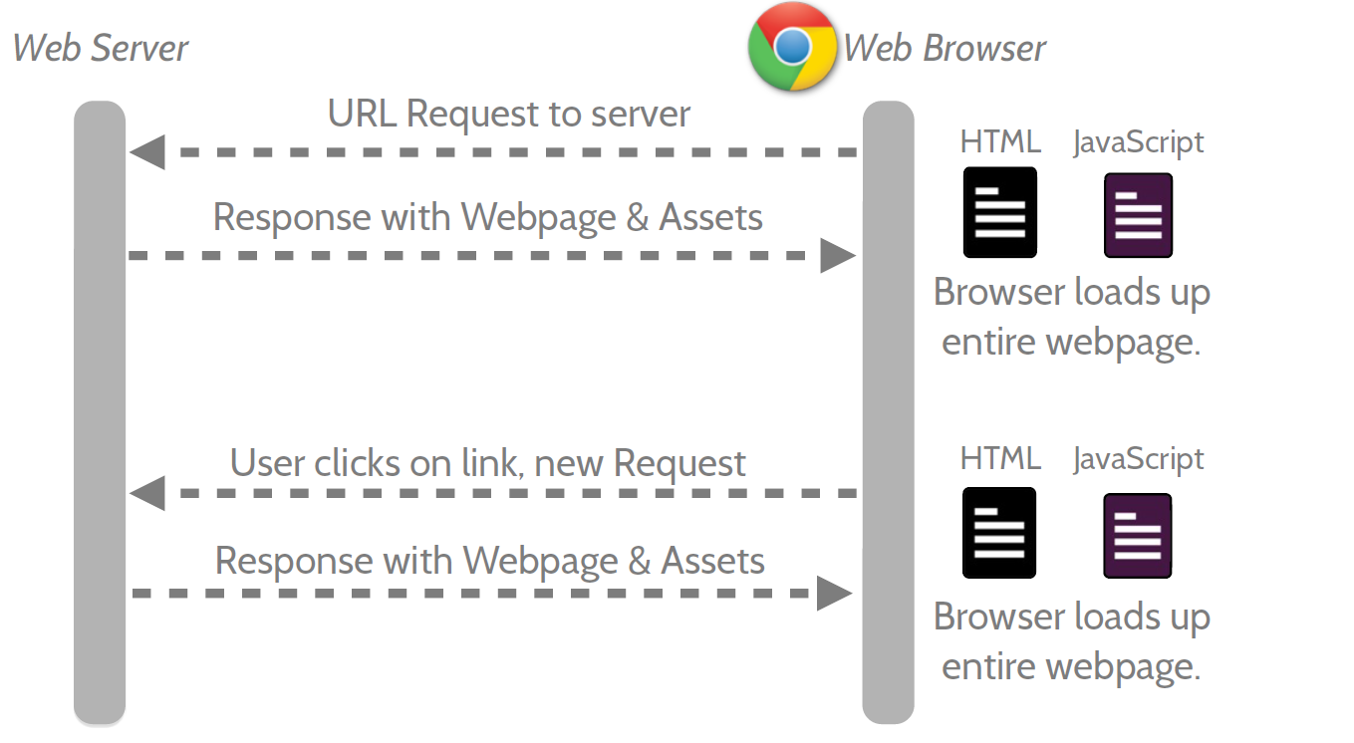


Figure 2‑2: Multi Page Application



Figure 2‑3: Single Page Application

* Features:
* AngularJS is a powerful JavaScript based development framework to create RICH Internet Application (RIA).
* AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC (Model View Controller) way.
* Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.

Overall, AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain.

* Core futures: following are most important core features of AngularJS:
* **Data-binding:** It is the automatic synchronization of data between model and view components.
* **Scope:** These are objects that refer to the model. They act as a glue between controller and view.
* **Controller:** These are JavaScript functions that are bound to a particular scope.
* **Services:** AngularJS come with several built-in services for example $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
* **Filters:** These select a subset of items from an array and returns a new array.
* **Directives:** Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ngBind, ngModel...)
* **Templates:** These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
* **Routing:** It is concept of switching views.
* **Model View Whatever:** MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
* **Deep Linking:** Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
* **Dependency Injection:** AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

### Advantages and disadvantages of AngularJS

* Advantages:
* AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
* AngularJS provides data binding capability to HTML thus giving user a rich and responsive experience
* AngularJS code is unit testable.
* AngularJS uses dependency injection and make use of separation of concerns.
* AngularJS provides reusable components.
* With AngularJS, developer write less code and get more functionality.
* In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.
* Disadvantages:
* **Not Secure:** Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
* **Not degradable:** If your application user disables JavaScript then user will just see the basic page and nothing more.

### The reason for choosing AngularJS

* AngularJS supports to create a website faster via Rest Api and MVC Model.
* Provide a more fluid user experience akin to a desktop application.
* All team members want to study a new language.