NAIT

Edmonton, Alberta

Building a web camera with CodeIgnitor MVC framework

As a submission to

Mr. Kelly Shepherd, Instructor

English and Communications Department

Submitted by

Xiao Liu, Student

Computer Engineering Technology

Oct 21, 2018

12345 - 67 Ave NW

Edmonton, AB T5R 1A1

Oct 21, 2018

Mr. Kelly Shepherd

Instructor, English and Communications Department

NAIT

11762 - 106 St NW

Edmonton, AB T5G 2R1

Dear Mr. Kelly,

I am submitting the report Building a web camera with CodeIgnitor MVC framework, as you requested, for your evaluation. This report partially fulfills my obligations towards the requirements of CMPE2900: Technical Report Writing.

This report details the design and construction of a web camera application using CodeIgnitor framework based on MVC (Model-View-Controller) software design pattern. It goes over in details how the framework works and how it is useful for our application software development. It also goes over the design and construction of hardware component of the camera.

I’d like to thank all of my instructors at NAIT for their dedication to work for the success of their students. Their passion for what they teach is a great motivator in the classroom.

Sincerely,

Xiao Liu

CNT Student

Table of Contents

List of Figures and Tables ............................................................................................... v

Abstract...........................................................................................................................vi

1.0 Introduction ............................................................................................................... 1

2.0 MVC Overview ........................................................................................................ 2

2.1 CodeIgnitor framework........................................................................................................... 2

2.2 File structure...................................................................................................... 3

3.0 Project Design......................................................................................................... 12

3.1 Features and Requirements ................................................................................ 12

3.2 Design Patterns ................................................................................................... 14

3.3 Code Architecture................................................................................................ 16

3.3.1Controller……………………………………………………………………………

3.3.2 View……………………………………………………………………………………

3.3.3 Model……………………………………………………………………………………………

4.0 Hardware Implementation ..................................................................................... 18

4.1 servo circuit diagram.......................................................................................................... 19

4.2 camera controller

5.0 Conclusion .............................................................................................................. 30

References.................................................................................................................... 32

List of Figures and Tables

Figures

Figure 1: Program Flow when Calling an Asynchronous Method..................................

Figure 2: Application Class Model Diagram ..................................................................

Figure 3: Visual Studio’s Visualization of Class Interaction...........................................

Tables

Table 1: Client-Server Stream Setup and Authentication Messages...............................

**Abstract**

?https://www.tutorialspoint.com/mvc\_framework/mvc\_framework\_introduction.htm

This report discusses the implementation of MVC pattern through user authentication in the web camera application using CodeIgnitor MVC framework. The report also introduces the hardware design of the web camera using Raspberry Pi.

Building a web camera with CodeIgniter MVC framework

# Introduction

With the increasing popularity of web application, software developers are seeking for the framework which integrates the common web functionalities with built-in supporting libraries, provides scalabilities for enterprise environment and facilitates team-based coding practice. Therefor the MVC framework has been emerged as one of the popular frameworks to accommodate the software industry’s demand.

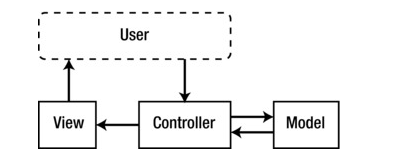
The purpose of this report is to work through the process of designing and implementing the MVC architecture pattern using CodeIgniter- a PHP-based framework to build a lightweight web camera application where users need to register and login through CodeIgniter-powered authentication process before access to web camera controller page. Compare with other frameworks that are using MVC pattern, such as Java Spring, Asp.net MVC, the CodeIgniter framework is the most suitable one found for this web camera application for its compatibility with Apache server installed on Raspberry Pi, light weight design for small and medium web project.

However there are many features in CodeIgniter that will not be covered in this report because CodeIgniter provides various other functionalities such as unit testing, custom libraries support, many of which are not used in this application due to the time constraint of development process. The focus of this report will be the user authorization implementation as a demonstration of how MVC works in web camera application.

An overview of MVC design pattern will be covered in section two, introducing CodeIgnitor framework. Overall design of the project and how the application will function will be covered in the third section. Hardware design and web camera controller page code structure will be covered in the fourth section. The last part of the project will discuss the challenges during the development process and possible future enhancement and improvement.

# 2.0 MVC Overview

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.



**Figure1 MVC flow (Pitt, 2012)**

As the above figure 1 depicts, the controller serves as the “brain” of the application which handles all users’ requests according to the business logic. It acts as “middleman” between view and model.

Model mainly handles all database interactions by setting up connection to database, performing SQL queries and updating data in database or retrieving data for controller.

View is where user interactive elements are kept (HTML, CSS, and JavaScript), any data that are presented to user on webpage is to be processed by Controller first. Sometimes what the user sees are the combination of different views.

An example of work flow of MVC framework is that user clicks a button on the webpage to update his password, the controller receives the update request, calls the model. The Model then retrieve the user’s row in database table and update the password. Upon completion, the model will notify controller and controller will update the view with success message to the user.

As Pitt (2012) points out, in any unstructured application, the development aspects (UI design, business logic, debugging, testing…) tend to melt together in an incoherent mess. When the database needs to be changed to accommodate a new product line, or the company decides to rebrand, it doesn’t only affect the code it should. More developers have to get involved to make sure that changes in one part of the application don’t immediately break other parts of the application. Changes that should only affect a tiny section of code end up spilling into all sorts of strange and problematic areas.

Therefore the MVC has come for rescue. It defines strict containers for all of an application’s code and features so that when there are changes to database, views and controllers will not break. When an application’s UI design changes drastically, its controller and model are not affected by the changes. All of which makes it possible for front-end and back end developers to work independently which is essential to any enterprise application development.

### 2.1 CodeIgniter

CodeIgniter is a powerful open-source PHP framework with a very small footprint, created by Rick Ellis in 2006. It was born from [ExpressionEngine](https://expressionengine.com/), essentially a collection of refactored classes originally written for EllisLab's flagship CMS. Stripped of the application-specific functionality, CodeIgniter was made to be a simple and elegant toolkit, enabling rapid development of both web sites and web applications, attracting thousands of talented PHP developers. (EllisLab, 2018)

<https://nait.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma992388764204191&context=L&vid=01NOALTECH_INST:01NOALTECH&search_scope=MyInst_and_CI&tab=OneTab&lang=en_US>

https://ebookcentral.proquest.com/lib/nait-ebooks/reader.action?ppg=38&docID=946937&tm=1540265330593

Pitt, C. (2012). *Pro PHP MVC*. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4302-4165-2

**https://ellislab.com/codeigniter**