A Major Project Report

On

PHP MVC Framework

Submitted in Partial Fulfilment of Requirements for the Award of the Degree of

Bachelor of Technology

In Computer Science Engineering

To

Guru Gobind Singh Indraprastha University, Delhi

Under the guidance of

Mr. Gourav Sharma

Submitted By:

ASHISH KUMAR SINGH {06425602714}

DEEPAK CHOPRA {06325602714}

Affiliated to GGSIP University, New Delhi
Approved by AICTE & Council of Architecture

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNICAL CAMPUS

KNOWLEDGE PARK-III, GREATER NOIDA,

APRIL – 2018

CANDIDATE’S DECLARATION

We, Ashish Kumar Singh and Deepak Chopra, (Enrollment No: 06425602714 and 06325602714), presently studying in 8th Semester, B.Tech - CSE (2014-2018 Batch), DTC, Guru Gobind Singh Indraprastha University, hereby declare that the work being presented in the Major Project & Report entitled "**PHP MVC Framework**" is an authentic and original record of our own work under the technical guidance/inputs of Mr. Gourav Sharma, Asst. Prof, DTC. We declare that the work in the said Major Project has not been submitted in part or in full for any diploma or degree course of this or any other University/Institute to the best of my knowledge and belief. We will be solely responsible ourselves for any copyright infringement or plagiarism, if any, in the said work.

Date:

Place: DTC, Greater Noida

ASHISH KUMAR SINGH

(06425602714)

DEEPAK CHOPRA

(06325602714)

CERTIFICATE

This is to certify that the Minor Project Report entitled "**PHP MVC Framework**" is an original and authentic work carried out by Ashish Kumar Singh and Deepak Chopra (Enrolment No: 06425602714 and 06325602714), a student of B.Tech-CSE (2014-2018 Batch) 8th Semester, DTC, GGS Indraprastha University, under my technical guidance/inputs to fulfil his academic requirements of the above said program. To the best of my knowledge and belief, the matter embodied in this work, in my opinion, has not been submitted in full or in part of any diploma or degree of this or any other University/Institute.

Date: **Mr. Gourav Sharma**

Place: DTC, Greater Noida (Assistant Professor, CSE)

ACKNOWLEDGEMENT

We express our sincere gratitude to Mr. Ravindar Kumar (HOD, CSE) and Mr. Gourav Sharma Faculty Guide (A.P, CSE), Delhi Technical Campus, for his valuable guidance and timely suggestions during the entire duration of my dissertation work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members of Computer Science Department, who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this work. Finally we would also like to thank our friends for their advice and pointing out our mistakes.

ABSTRACT

PHP is a powerful language to develop dynamic and interactive web applications. One of the defining features of PHP is the ease for developers to connect and manipulate a database. PHP prepares the functions for database manipulation. However, database management is done by the Structure Query Language (SQL). Most novice programmers often have trouble with SQL syntax. In this paper, we present the PHP framework for database management based on the MVC pattern. The MVC pattern is very useful for the architecture of web applications, separating the model, view and controller of a web application. The PHP framework encapsulated, common database operations are INSERT, UPDATE, DELETE and SELECT. Developers will not be required to consider the specific SQL statement syntax, just to call it the method in the model module. In addition, we use White-Box testing for the code verification in the model module. Lastly, one or two web application example is shown to illustrate the process of the PHP framework.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

MVC	Model, View and Control
PHP	Hypertext Pre-processor
WAMP	Windows, Apache, MySQL and PHP
SQL	Structured Query Language
MySQL	My Structured Query Language
ORM	Object Relational Mapper
DBMS	Database Management System
HTTP	Hypertext Transfer Protocol
CRUD	Create, Read, Update and Delete
CSRF	Cross-site Request Forgery
HTML	Hypertext Markup Language
CSS	Cascading Stylesheets
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
XML	Extensible Markup Language
API	Application Programming Interface
UI	User Interface
CLI	Command-Line Interface

CHAPTER 1: INTRODUCTION

The project title “**PHP MVC Framework**” is actually a software framework project. A software framework is a universal, reusable software platform used to develop applications, products and solutions. The MVC framework is a well-established design pattern that is widely used for applications with a user interface component .Then this project also includes this feature. This type of framework avoids the unauthorized access and it authenticates the authorized users or hosts.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows. Although originally developed for desktop computing, MVC has been widely adopted as an architecture for World Wide Web applications in major programming languages. Several web frameworks have been created that enforce the pattern. These software frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server.

Even though MVC was originally designed for personal computing, it has been adapted and is widely used by web developers due to its emphasis on separation of concerns, and thus indirectly, reusable code. The pattern encourages the development of modular systems, allowing developers to quickly update, add, or even remove functionality.

It is worthwhile considering the use of a PHP framework when time is a limitation and the developer’s PHP coding skills do not match the high level demanded to build a complex application. Frameworks handle all the repetitive basic tasks of a PHP project, letting the developer concentrate her/his efforts on the business logic and the general structure of the project as a whole, in doing so, frameworks are becoming an ideal tool used by said developers to rapidly build complex operational prototypes in a matter of hours with minimal time spent on coding. Frameworks offer also whole range of ready-made utilities and libraries.

1.1 PROBLEM STATEMENT

The main objective of this project is to create a web development framework that provides all the essential functionalities and libraries to kick start a project easily. A platform that can be used by any level web developer from beginner to an experienced one to get started on an application easily and swiftly. The platform/framework should be easy to understand and use of developers, and can be modified easily and the libraries pre-included could be removed or modified or can be added as per the application requirements.

1.2 PURPOSE

The fundamental purpose of designing this project is to provide a structure and some inbuilt functions that are required by every web application in order to function optimally. Web applications share some common data amongst themselves through API that can be added in this framework. The main reason to develop this framework was to provide just the basic functionalities not any extra classes that might slow down the overall process cycle of request-response. The idea for choosing the mvc pattern was that it is easily upgradable and the user can add its own functionalities and its own libraries and usually can even improve the pre-built core files. The Framework has got a built in functionality to load all the classes automatically and the user does not need to add any include files on its own.

Customers Intended:

Any web application or website project that are intended for a large scale users and can be scaled to even more users.

Features

Following are some important features that come along with this project these are:-

- The application built on framework is secured.
- It is easy to understand the whole structure.
- Easy to troubleshoot errors because of its easy understanding.
- We can extend the modules.
- Separate backend logic.
- Separate frontend logic.

Assumption

There are some assumptions that are needed to be taken before studying the project that are:

- MVC architecture pattern and its working knowledge.

- Most of the web applications need to follow some structure in order to easily deploy.
- Basic knowledge of web applications html, css and php is required for this framework.

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project can be used by any type of web application, a sample website, or a prototype of an application. Any level of developers could use this framework and start building applications may it be a beginner or an experienced one. A basic knowledge of the MVC pattern or working knowledge of PHP could start an application on this framework. MVC is not just any other web development framework architecture for building elegant and systematic websites, but it has a full capability to support rapid web application development and dynamic interactivity with the database as well.

1.4 PROJECT SCOPE

The scope of this project is to have a secure php framework that can help in easier and faster development of the web applications that can be easily deployed to the web servers and can be scaled even further for the use of many users.

The MVC design architectural pattern promotes cohesion (classes that represent or define only one type of object) and aims for low coupling (a measure of how much classes are interrelated). In UI object oriented programming, it is considered bad practice to mix presentation and data code. Instead, classes called controllers are defined to mediate communication between presentation and data classes. These controller classes decouple the presentation layer and the data layer classes, consequently allowing them to be defined more specifically.

These frameworks use coding standards and development guidelines so help in standardizing the process and stabilizing the product. All PHP frameworks use Model View Controller (MVC) architecture, where the development of the business logic is independent and view and models are separately interlinked with each other.

Future upgrades of this framework:-

1.) Excellent community support.
2.) Easy to upgrade and maintain the developed applications.
3.) Default Security features such as CSRF Protection and Output Encoding.
4.) Rapid development using pre-built libraries and frameworks.
5.) Very strong encryption packages.
6.) Totally developer friendly doesn’t need any special dependencies or supports
7.) Outperforms most other frameworks (non-MVC)
8.) Good Documentation and easy to use

CHAPTER 2: OVERALL DESCRIPTION

Model-View-Controller is typically made up of three classes as mentioned in its name. The controller is the intermediary class between the model and the view classes. It controls the flow of information by accepting user input from the view, and instructs both the model and view to perform actions based on that information. The model is responsible for the data management routines in the application. These are commonly create, read, update, and delete (CRUD) database operations. The view is responsible for presenting the data from the model, and normally contains mostly mark-up displayed as web pages, or for example, RSS feeds.

An example of a webpage with MVC could be a news page. A user uses a web browser to access the website, which connects to the controller. Controller needs data from the database so it will call a function in the model-class. The model connects to the database and handles the queries and is returning the data to the controller. Controller then contacts and retrieves View, that mostly contains HTML and only uses PHP to loop out the data and returns it to the user as seen in point.

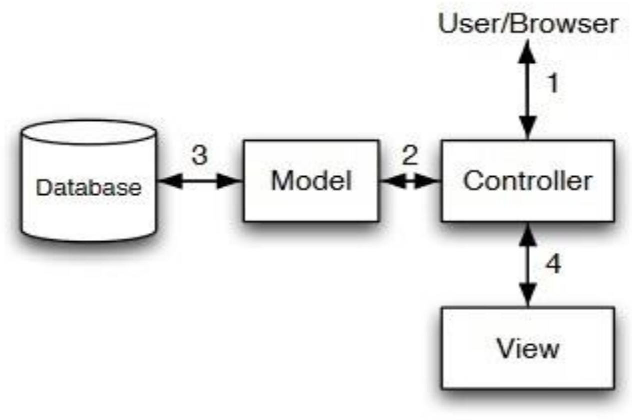


Figure 1: A SIMPLE MVC STRUCTURE

2.1 Components of a web framework

Web application frameworks differ highly on their levels of abstraction and the amount of features they make available to the programmer. Most frameworks contain myriads of features, some of them general purpose, some of them specialized towards specific application architectures encouraged by the specific framework. Four basic components are, however, made available by all web application frameworks, a dispatcher, a decoder, a generator, and a store. This section presents an overview of these components and the design space for each of them.

2.1.1 The Dispatcher

A dispatcher defines the relationship between HTTP requests and web application code by locating and invoking code based on the contents of HTTP requests. A dispatcher can be explicitly configured through a configuration file or it may be implicitly configured through conventions. In the latter case, conventions typically determine how the dispatcher generates mappings from URLs to code based on the class or file structure of the program code. The design of the dispatcher defines what the basic unit of the web application framework is. A basic unit corresponds to a single, possible entry point that can be invoked by the dispatcher. Basic units can be source files in which the dispatcher will typically start executing the code from the beginning of the file. It can be classes where the dispatcher can invoke a method based on a predefined interface.

2.1.2 The Decoder

A decoder decodes requests from clients and provides means for the web application to read parameters, headers, and request body data sent as part of the request. We can categorize decoders into two types:

1) **Pull decoders:** They provide an interface to the decoder itself. The programmer retrieves the request parameters by invoking methods on this interface. A map from names to parameter values is the simplest form of such a pull decoder. Examples of such decoders exist in PHP, JSP, and the Servlet framework.
2) **Push decoders:** They inject the request values into method parameters or properties before the dispatcher invokes the entry point. The choice of parameters or properties typically depends on the choice of basic unit for the decoder: parameters are used if the basic unit is functions. The push decoder approach makes it simple to identify the interface of a web application, while the pull approach provides the programmer with flexibility. Most of the push decoder frameworks also provide means for the programmer to read parameters in pull style for situations where the added flexibility is necessary. Of the pull decoder frameworks discussed later in the chapter, only Hop and Seaside are purely push style and provide no pull features.

2.1.3 The Generator

The generator constructs output that is returned to clients as response to requests to the server. The generator has a large variety of design features which we will try to categorize here. The web framework may provide a domain specific language (DSL) for writing templates intermixed with program code or it may rely on general purpose language syntax only. Template languages typically permit the programmer to write HTML code fragments in exactly the same syntax as is used for writing static HTML documents and they provide some way to combine these templates into a document. Frameworks without templates typically allow the programmer to generate output through function calls that have side effects in the generator.
Templates are used in the PHP, Servlets, JSP, JSF, Struts, and JWIG frameworks while the Servlets, Seaside and Hop frameworks rely on the syntax of Java, Smalltalk and a Scheme

like language respectively. Furthermore, JSP and Struts allow the programmer to intermix templates and calls that have side effect in the generator. Orthogonally to the inclusion of template syntax, the output might be represented as a first class value in the programming language or be implicitly represented by the runtime system, for example as a result of appending data to an output stream read by the client. While first class values offer a high degree of freedom to the programmer, the stream approach allows data to be retrieved and rendered by the client while the server side is still executing. A further design choice for the output representation is whether the values are mutable or immutable. On the client side, the document is represented as a first class, mutable structure, the DOM. There are only few examples of such a representation in server side web application frameworks. Orthogonally to these types, the generator may offer various degrees of protection against cross-site scripting attacks.

2.1.4 The Store

The store holds inter-request data. The store is often separated into various scopes, such as an application scope where the data is shared between all requests and clients, a session scope where the data is shared among requests from the same client, and a request scope where the data is shared only for the current request. Frameworks may provide fewer or more scopes than this or they may provide only the scope features of the hosting programming language. The store may require separation from the rest of the code, so that the structure of the store must be represented as classes that are separate from the code that interacts with the generator. It might also allow integration so that the programmer can store data as part of the same code that interacts with the generator. Finally, the store may be typed so that the type of a value is guaranteed by the type system of the programming language or it may be untyped and leave it up to the programmer to ensure type correctness. In Java the values may be represented as properties of Java Beans to have a typed store or represented as a string-to-object maps to have an untyped store. In the latter case, the programmer must cast the value to the expected type.

2.1.5 URL Mapping

A framework's URL mapping or routing facility is the mechanism by which the framework interprets URLs. Some frameworks, such as Drupal and Django, match the provided URL against pre-determined patterns using regular expressions, while some others use rewriting techniques to translate the provided URL into one that the underlying engine will recognize. Another technique is that of graph traversal such as used by Zope, where a URL is decomposed in steps that traverse an object graph (of models and views). A URL mapping system that uses pattern matching or rewriting to route and handle requests allows for shorter more "friendly URLs" to be used, increasing the simplicity of the site and allowing for better indexing by search engines. For example, a URL that ends with "/page.cgi?cat=science&topic=physics" could be changed to simply "/page/science/physics". This makes the URL easier for people to remember, read and write, and provides search engines with better information about the structural layout of the site. A graph traversal approach also tends to result in the creation of friendly URLs. A shorter URL such as "/page/science" tends to exist by default as that is simply a shorter form of the longer traversal to "/page/science/physics".

2.1.6 Database Access

Many web frameworks create a unified API to a database backend, enabling web applications to work with a variety of databases with no code changes, and allowing programmers to work with higher-level concepts. Additionally, some object-oriented frameworks contain mapping tools to provide object-relational mapping, which maps objects to tuples. Some frameworks minimize web application configuration through the use of introspection and/or following well-known conventions. For example, many Java frameworks use Hibernate as a persistence layer, which can generate a database schema at runtime capable of persisting the necessary information. This allows the application designer to design business objects without needing to explicitly define a database schema. Frameworks such as Ruby on Rails can also work in reverse, that is, define properties of model objects at runtime based on a database schema. Other features web frameworks may provide include transactional support and database migration tools.

2.1.7 Cohesion, coupling, and the MVC pattern

The notions of high cohesion and low coupling describe applications that are structured into multiple loosely coupled components. In lowly coupled applications, changes can be made locally without affecting other parts of the application. Low coupling improves maintainability of the code. Dually, in highly cohesive systems, code that is closely related in functionality is also closely related in the code. High cohesion improves readability of the code. High cohesion and low coupling can be properties of the overall architecture as well as the implementation and structure of the application code. Several architectural patterns promote high cohesion and low coupling. In particular the model-view-controller (MVC) pattern has gained popularity in web applications frameworks. In this pattern, the use of the generator – the view is separated from the model that represents the values in the store and the controller that updates the model and view upon client interaction. Many good sources explain the MVC pattern and the reader is assumed to be familiar with the pattern.

2.2 Project Components

This project consist of three main modules basically model, controller and views part and each one of them is important to the application. The purpose of modules and frameworks is not so much to achieve independence of functions as it is to achieve reusability and flexibility. In most other occupations, this kind of reuse of function and flexibility is central to achieving success.
Think about how difficult it would be to work on your car if you had to buy a whole new tool set for each car, because the tools were not interchangeable. Or worse, you have to build the tools from scratch, because there wasn't a factory to create them. Both of these concepts (universal plug ability and object factories) are present in well-written modules and frameworks.

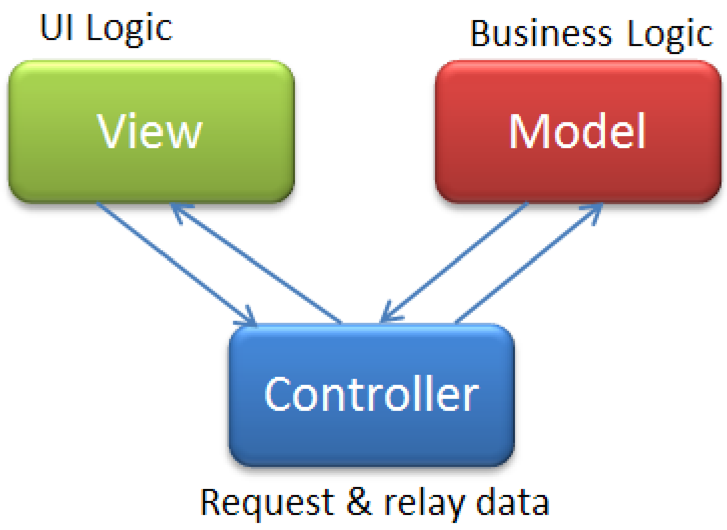
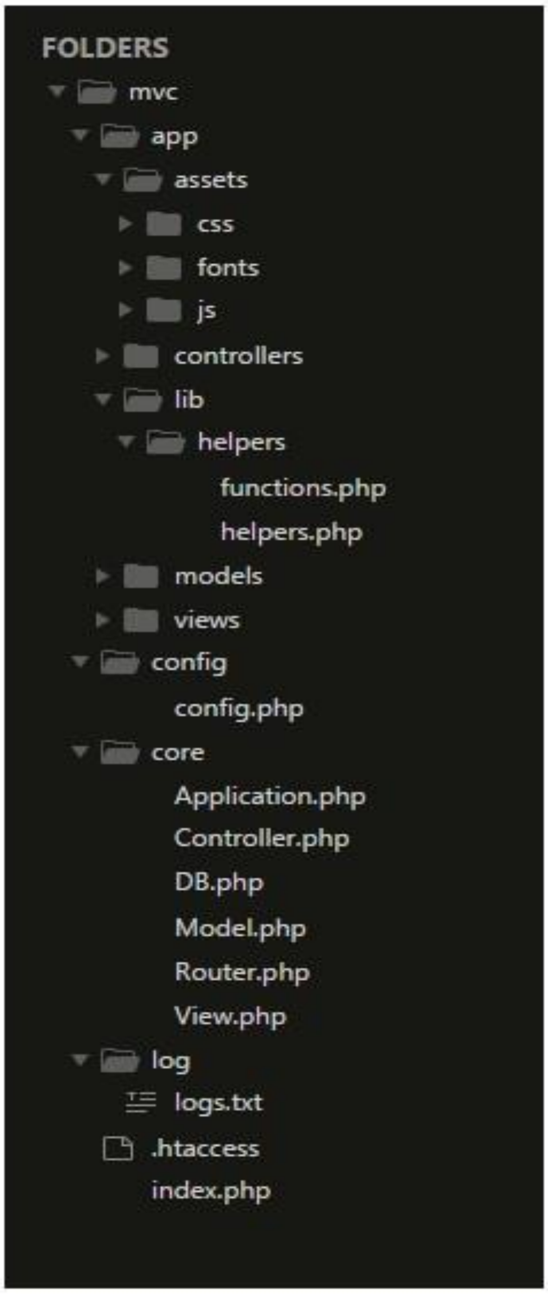


FIGURE 2: MVC CYCLE

MVC is a pattern for the architecture of a software application. It separates an application into the following components:

- **Models** for handling data and business logic.
- **Controllers** for handling the user interface and application.
- **Views** for handling graphical user interface objects and presentation.

2.2.1 Directory Structure



```
FOLDERS
▼ mvc
  ▼ app
    ▼ assets
      ▶ css
      ▶ fonts
      ▶ js
    ▶ controllers
    ▼ lib
      ▼ helpers
          functions.php
          helpers.php
    ▶ models
    ▶ views
  ▼ config
      config.php
  ▼ core
      Application.php
      Controller.php
      DB.php
      Model.php
      Router.php
      View.php
  ▼ log
      logs.txt
    .htaccess
    index.php
```

Figure 3: DIRECTORY STRUCTURE OF THE FRAMEWORK

The file structure of the framework consists of 4 folders and some files:-

- Application
- Config
- Core
- Logs
- Htaccess file
- Index.php file

2.2.2 Application Folder

As the name suggests the application folder or the app folder consists of all the essential files, the code and assets files of the application. This is the core directory of your entire app and most of the application-specific code will go into this directory. This means the application is separated into parts per its purpose in the Model, View, or Controller. These three sections goes inside this directory. Similarly all the essential static files associated with the application of the user goes in this directory. The basic cascading stylesheets and all JavaScript files needed for the projects are saved in this folder. This application folder further consists more directories namely:-

- **Assets**
- **Controllers**
- **Lib**
- **Models**
- **Views**

2.2.2.1 Application/Assets

The assets directory in this folder will consist of the static files that need to be served for every user using the web application. The three major type’s files that need to be served for a web application are the css, js and fonts. These files help in the presentation layer of the application and improve the user interface of the application. The files can also be served from a reverse proxy server and can further improve the time while serving a file in the production server. Thus improving the overall efficiency of the user application, but that need only depends on the user and its application requirements. This directory further consists of three folders:-

1. **Css**
2. **Js**
3. **Fonts**

As the name suggests the css folder will save all the cascading stylesheets of the application and the js folder will save all the frontend code or the JavaScript files of the application and same goes with the fonts folder if new type of fonts are required.

Whereas further new directories can be created in this folder to hold further static files of the application, for example some websites have their own set of glyph icon’s, this framework has the ability to store and serve them as well. The ability of the framework to have a separate frontend application and exchange data with backend with the help JSON objects can be done in this framework. The user can have different frontend and backend application to combine together and make a same application together. Most of the application these days are made by separate type of experts, developers with expertise in frontend logic and developers with expert knowledge in backend logic.

2.2.2.2 Application/controllers

A controller is a class, and like the model and view classes, is part of the MVC architectural design pattern. The job of the controller is to act as a data intermediary between the view and model classes, but it also determines how HTTP requests should be handled. When the user interacts with an application, they will be accessing the controller, so controllers are also thought of as the access point of an application. To expand upon the concept of an access point, in the URLs and routing sub chapter, it was explained what a URI segment was. To reiterate, a URI segment is simply a part of a URL that can call a method (with optional arguments) defined inside of the controller class. This class method is usually written to load specific libraries and helpers, orchestrate calls from the model, and load a particular view while passing data to it, which is ultimately returned to the user. This directory holds all the controllers of the application and can be used as an intermediary between models and views. Controller is often referred to as the application layer of the website.

```
Class SampleController extends Controller {}
```

This is how a basic controller should be created, so that the class inherits all the basic functionalities from the core controller class. The Controller component is basically the code that processes data, writes out pages, gets data, logs, creates events and so on. Essentially this is the active part of the site & system which interface between the databases, assets, template set, generating a result which the end user can see. Whenever the user sends a request for something then it always go through the Controller. The Controller is responsible for intercepting the requests from View and passes it to the model for the appropriate action.

2.2.2.3 Application/lib

Libraries are classes that extend the functionality of the framework. As mentioned earlier, a full stack web application framework is not just an architectural pattern, but also its libraries. They can be thought of as a toolkit, or as add-ons, and are essentially features that will aid in the rapid development of a web application. Frameworks libraries reside in the application/lib directory, while user defined, or third party libraries reside in application/libraries. Libraries are initialised in the controller, for example:

```
$this->load->library('library_class_name');
```

Once loaded, the developer can use its methods as documented. This libraries once included can improve the functionality of the application. Various libraries could be provided to the user as an additional module of code that could be loaded later on in the development of the project and as per the requirements. Authentication library and forms library are some famous examples of this. Authentication library helps the user encrypt the requests that are being sent to the user and later on that can be decrypted as well. Authentication library can be useful while saving password and useful information of the user in the database.

2.2.2.4 Application/models

Models, like views are an integral part of the MVC architectural pattern. In essence, a model is a class designed to work with information in a database. Due to the flexibility of the framework, it is not compulsory to use them. However, developers following the traditional MVC approach typically will make use of models. One can then access them by their class name. Although the model is usually for working with data access logic, it is also common for validation rules to be defined in the model and then called by a controller. The Model is where business logic is stored. Business logic is loosely defined as database connections or connections to data sources, and provides the data to the controller. The Model object knows about all the data that need to be displayed. It is Model who is aware about all the operations that can be applied to transform that object. It only represents the data of an application.

```
Class SampleModel extends Model {}
```

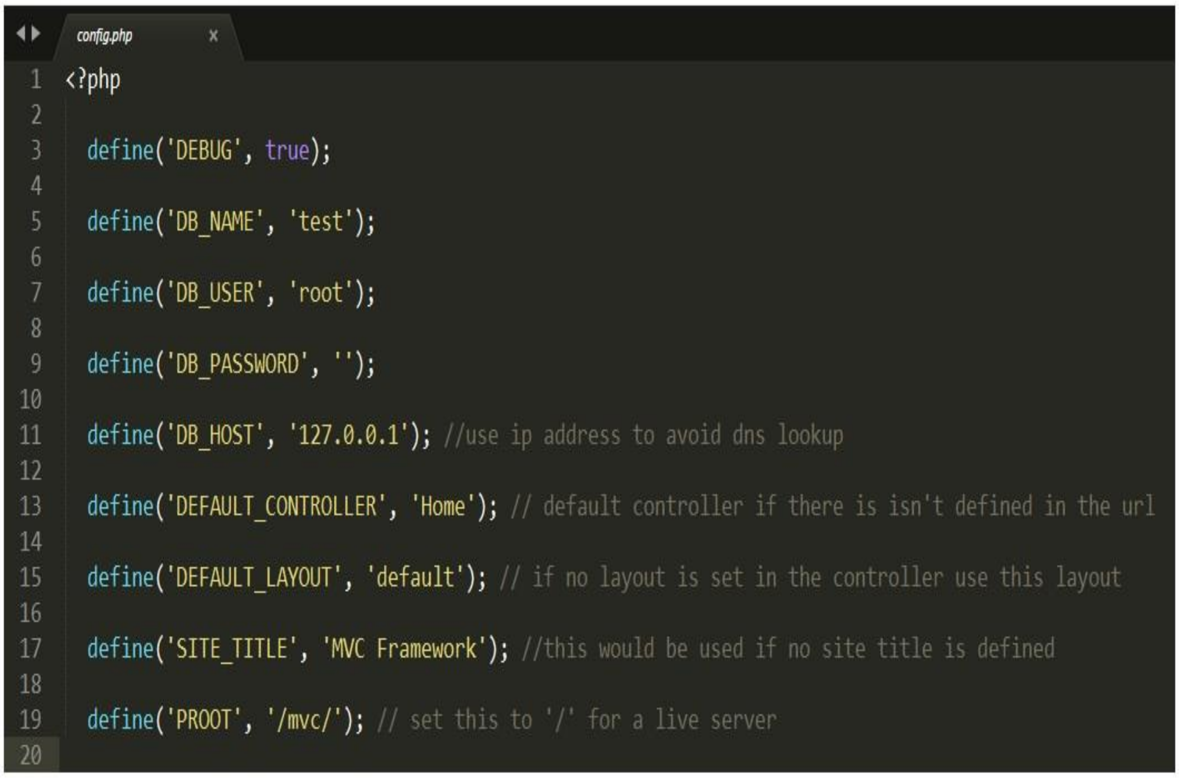
This should be the way of creating a model for the application, extending from the model class will give the database access to this class and the model can then perform all the database transactions that it needs after this. The Model represents enterprise data and the business rules that govern access to and updates of this data. Model is not aware about the presentation data and how the data will be display to the browser. The models created by the user are stored in this directory and should extend core model present. This extending of core model file will provide the basic functionalities the model class would be needing.

2.2.2.5 Application/views

Views are an integral part of the MVC architectural pattern. They play their part in separating application logic (business logic, data access logic, and validation logic) from presentation. Views can be simply thought of as web pages, or fragments of a web page (header, footer, menu etc...). However, the view itself (unlike traditional web applications), is not directly called by the end user. They are always loaded by the controller as instructed by the developer allowing for greater flexibility of presentation types. As mentioned earlier, if a view is fragmented, Framework will intelligently handle multiple calls to load each view fragment by appending them in the order each fragment was called. When variable data is required inside the view, it is defined and passed from the controller by way of an associative array or an object. When passed as an associative array, the key values are directly used as variables, or as arrays (if they were an array type). Framework views support its own special templating syntax which is meant to condense or ‘prettify’ regular PHP, which is already a HTML templating syntax. However, framework’s templating syntax requires parsing which makes applications slower, and of course, it also requires time for the developer to learn. The view folder further contains a **layouts** folder that can have different templates for different web pages and different modules of code. In layouts folder even different themes can be saved in above a view the layout should be declared and the corresponding theme would be enabled on that page.

2.2.3 Config folder

The Config folder consists of a single config.php file that contains all the application specific defaults. These constants depend on the application type, the developers and the environment of hosting or the development environment.



```php
<?php

  define('DEBUG', true);

  define('DB_NAME', 'test');

  define('DB_USER', 'root');

  define('DB_PASSWORD', '');

  define('DB_HOST', '127.0.0.1'); //use ip address to avoid dns lookup

  define('DEFAULT_CONTROLLER', 'Home'); // default controller if there is isn't defined in the url

  define('DEFAULT_LAYOUT', 'default'); // if no layout is set in the controller use this layout

  define('SITE_TITLE', 'MVC Framework'); //this would be used if no site title is defined

  define('PROOT', '/mvc/'); // set this to '/' for a live server

```

Figure 4: Configuration File

Debug: -	Sets the error reporting on or off in the environment, usually it is set to off in production environment and on in development environment.
Db_name: -	Sets the database name of the application and further connections and transactions are done on the same database.
Db_user: -	Usually every application on the internet has different user for every application for security purposes, we can set the user for logging in database.
Db_pass: -	The password needed to login into the database in provided here and can be left blank if not needed.
Db_host: -	The URL of the application or the address of the database server is provided, database can be saved to a different server to that of a application.
Default_controller: -	The most basic controller name of the application is saved here and default calls are routed here.

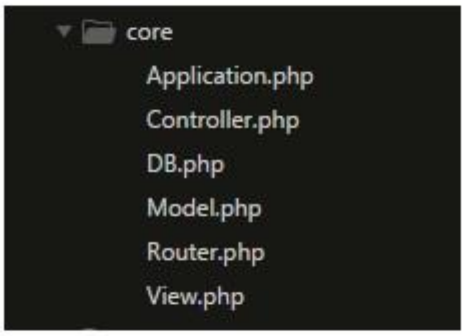
Default_layout: - Every application on the internet needs a proper layout on its user interface and default is saved here.

Site_title: - A default site title is provided here and in case of not individual page title is set, this title would be displayed and no error would be thrown.

Proot: - Project root directory address is saved here, all the redirects on the application are based on this route.

Similarly further more constants can be created in the same file as per the requirements of the user. But one thing should be noted in this case that this file contains all the database passwords and configuration information, and should always be kept in safe condition away from normal users. If needed this information should always be in encrypted before sending to the outside world, so that no one exploit the information.

2.2.4 Core Folder

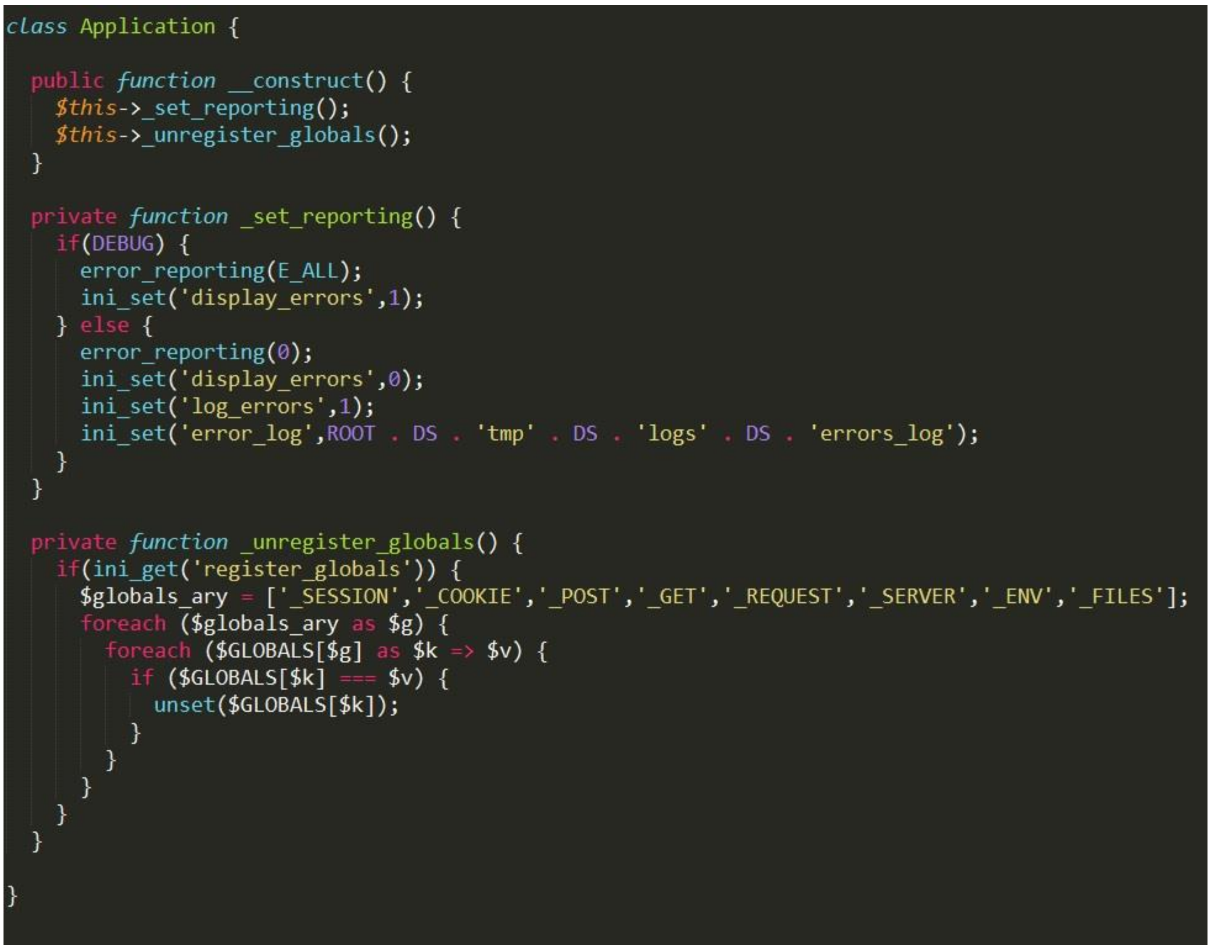


```
core
    Application.php
    Controller.php
    DB.php
    Model.php
    Router.php
    View.php
```

Figure 5: Core File Directory

The core directory of the framework consists of five main and important files. These five files Contain all the important functions and all the functionality of the framework. These files indicate modularization in the framework and less coupling levels in the framework and high cohesion levels in the framework. For the security and efficiency purposes these files are kept to a minimum number, so that the function of the framework serves the files at a faster rate than normal, but for needs of the users these files can be increased even. Like an authentication file can be added that encrypts all the requests of the user before sending out and decrypts if always before performing any operation.

2.2.4.1 Core/application File



```php
class Application {

  public function __construct() {
    $this->_set_reporting();
    $this->_unregister_globals();
  }

  private function _set_reporting() {
    if(DEBUG) {
      error_reporting(E_ALL);
      ini_set('display_errors',1);
    } else {
      error_reporting(0);
      ini_set('display_errors',0);
      ini_set('log_errors',1);
      ini_set('error_log',ROOT . DS . 'tmp' . DS . 'logs' . DS . 'errors_log');
    }
  }

  private function _unregister_globals() {
    if(ini_get('register_globals')) {
      $globals_ary = ['_SESSION','_COOKIE','_POST','_GET','_REQUEST','_SERVER','_ENV','_FILES'];
      foreach ($globals_ary as $g) {
        foreach ($GLOBALS[$g] as $k => $v) {
          if ($GLOBALS[$k] === $v) {
            unset($GLOBALS[$k]);
          }
        }
      }
    }
  }

}
```

Figure 6: Application Core file

This is the application core class, this class is responsible for performing two basic and most important functions to keep the user application safe from any security threats. The unregister global functions unset all the values of the global variables. These global variables include GET, POST, PATCH, SERVER etc. These variables hold the important information of users and it is necessary to empty these values after use. This bug was detected in PHP version 5.4 and later on all the developers were told to add on this functionality to their application.

The other function according to the user specification either sets the error reporting to the application browser or logs the error in the text file saved in logs folder. This is also an important feature since in production server the error should not be shown to the user rather it should be logged down in a file so further action on those errors could be done later and improved by developer. This way there is always a record in place of what errors came and the situation as well of the errors. This is an important feature for debugging and improving the application purposes.

2.2.4.2 Core/Controller File

The core controller file is the base of all controllers present in the application, all controllers In application inherits from this core controller file only. This file is responsible for a new instance of every controller file. This itself extends from the application file.

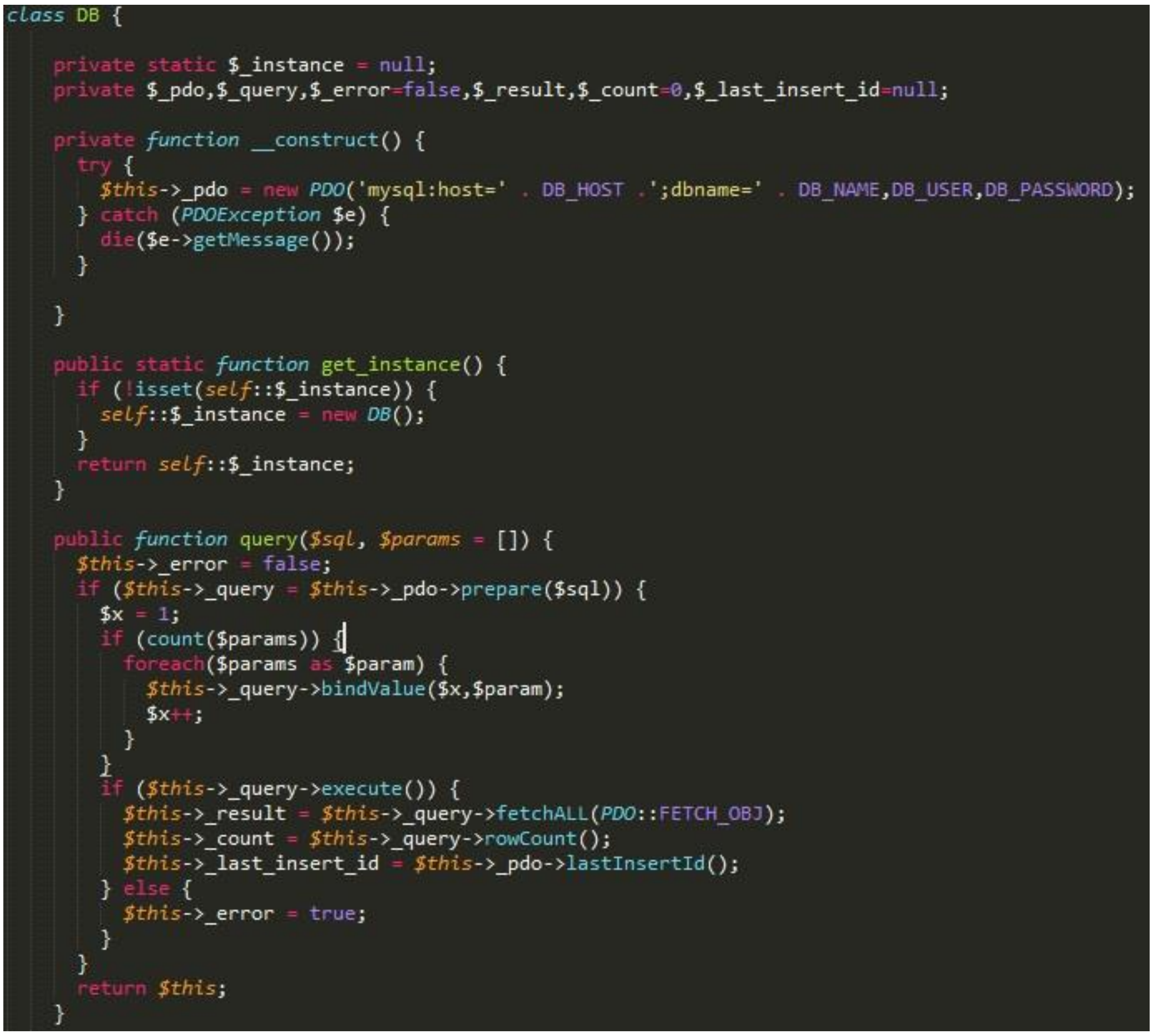
```
class Controller extends Application {

        protected $_controller, $_action;
        public $view;

        public function __construct($controller,$action) {
                        parent::__construct();
                  $this->_controller = $controller;
                      $this->_action = $action;
                    $this->view = new View();
                                   }
}
```

This is code of the core controller file that performs all the essential function all calling the controller class and calling its view later on in the application. Controllers are actually classes that extends a base class somewhere in the code and the methods will be matched without route properties. The Controller serves as an intermediary between the Model, the View, and any other resources needed to process the HTTP request and generate a web page, segments of the URL are mapped to a class and function. These classes are controllers stored in the "application/controller" directory.

Furthermore caching and privileges can also be added to the controller class to even further improve the security of the application, so that no unauthorized user can access or modify data and only authenticated users have the access to functions that add or retrieve data from the database.

2.2.4.3 Core/DB File

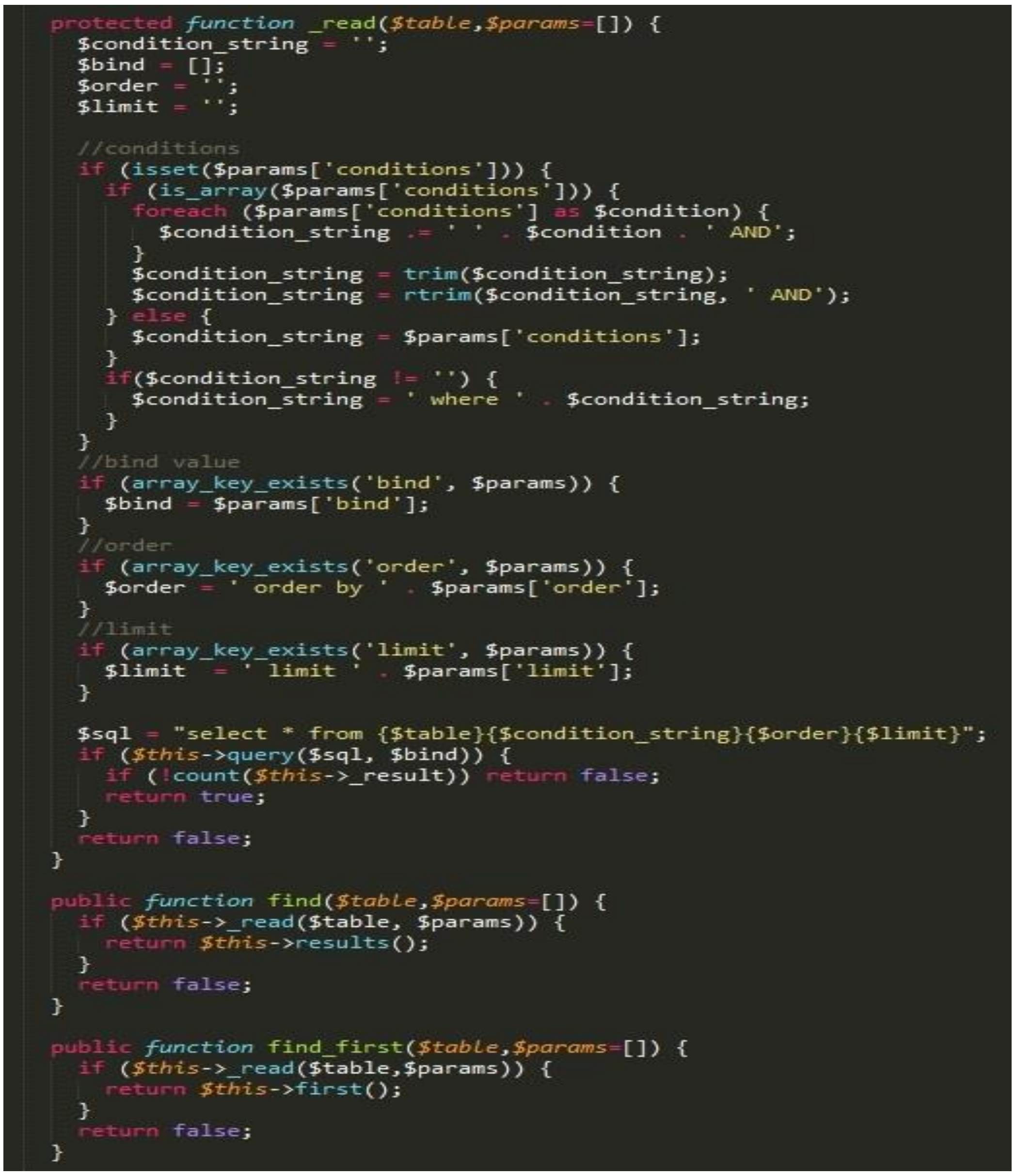
This is the biggest of the core files and any database transaction that needs to be handles is done by this class only and any inserting to data in the database as well. This class keeps check that only one instance of database class is present in the whole application so there in no race condition. This instance ensures that at one moment only of the database transaction occurs and multiple transactions cannot overwrite the same data. This core files holds the method of the basic CRUD functions ie create, read, update and delete. All the models in the application would perform their crud functions form this file only so that the code always remain DRY(Don’t repeat yourself), so there is no repeation in the code.



```php
class DB {

    private static $_instance = null;
    private $_pdo,$_query,$_error=false,$_result,$_count=0,$_last_insert_id=null;

    private function __construct() {
      try {
        $this->_pdo = new PDO('mysql:host=' . DB_HOST .';dbname=' . DB_NAME,DB_USER,DB_PASSWORD);
      } catch (PDOException $e) {
        die($e->getMessage());
      }

    }

    public static function get_instance() {
      if (!isset(self::$_instance)) {
        self::$_instance = new DB();
      }
      return self::$_instance;
    }

    public function query($sql, $params = []) {
      $this->_error = false;
      if ($this->_query = $this->_pdo->prepare($sql)) {
        $x = 1;
        if (count($params)) {
          foreach($params as $param) {
            $this->_query->bindValue($x,$param);
            $x++;
          }
        }
        if ($this->_query->execute()) {
          $this->_result = $this->_query->fetchALL(PDO::FETCH_OBJ);
          $this->_count = $this->_query->rowCount();
          $this->_last_insert_id = $this->_pdo->lastInsertId();
        } else {
          $this->_error = true;
        }
      }
      return $this;
    }
```

Figure 7: Database Core File – 1

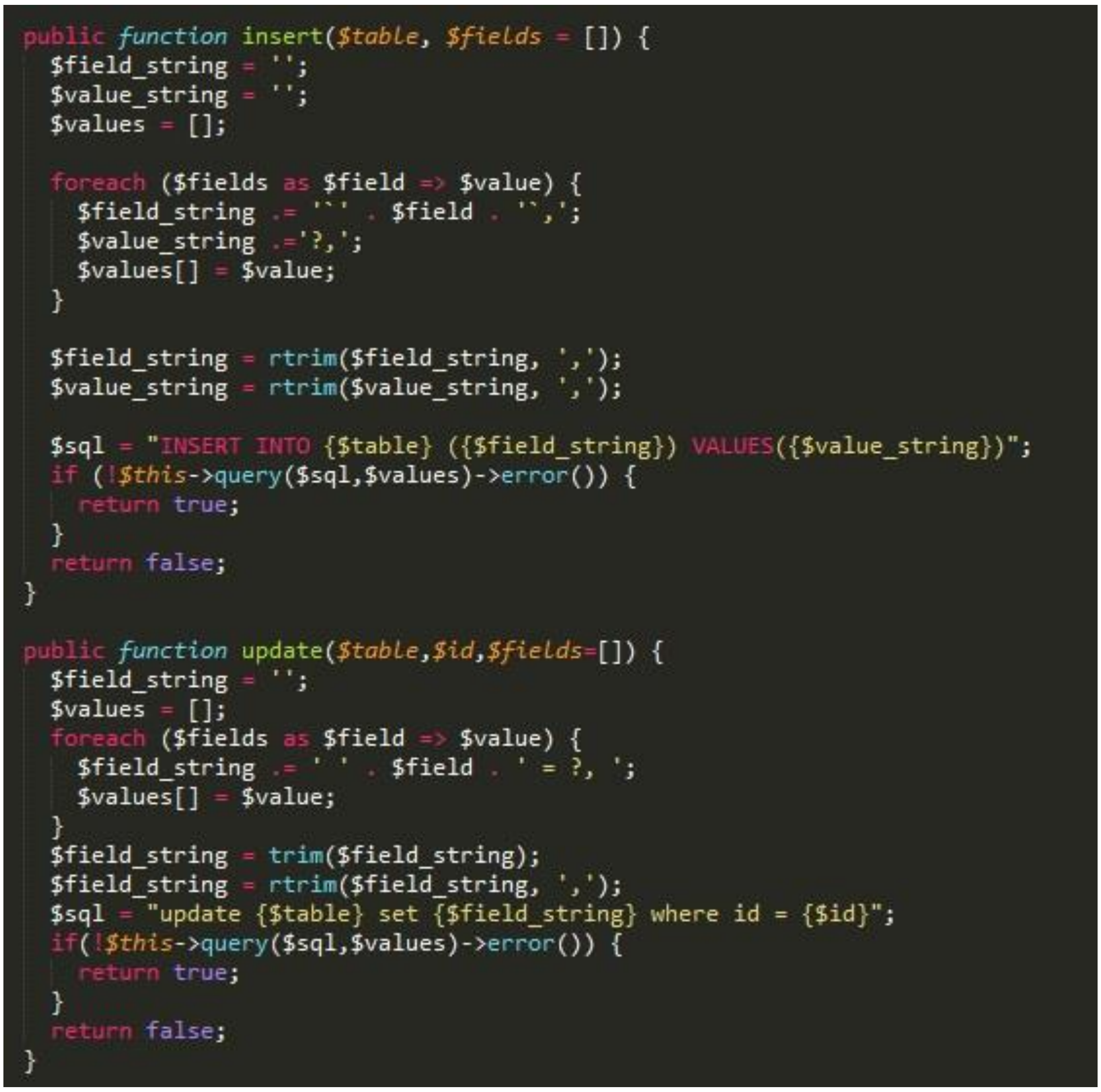
In the above figure the database core file contains, the first function is the constructor function and it establishes the connection between the application and the database stores this connection variable in a class variable called instance. The second function in the file returns the only database instance that the application has, this was present in the class variable named instance and would be returned in this function. The third function perform special queries on the database that the user wants, this function takes two arguments the sql text and the params array. This query is executed and the function returns false if the query false and sets the class variable error as true, or if the query is executed then the results are stored in the class variable named result which can then be accessed by the user in their views. This function also stores the last inserted id if there was any and the number of rows returned are stored in a class variable named count, usually the default values of these variables are set to null or 0 in the case of last inserted id, but comes in handy while creating a user and saving its id onto session variables for logging him in the application.



```
protected function _read($table,$params=[]) {
  $condition_string = '';
  $bind = [];
  $order = '';
  $limit = '';

  //conditions
  if (isset($params['conditions'])) {
    if (is_array($params['conditions'])) {
      foreach ($params['conditions'] as $condition) {
        $condition_string .= ' ' . $condition . ' AND';
      }
      $condition_string = trim($condition_string);
      $condition_string = rtrim($condition_string, ' AND');
    } else {
      $condition_string = $params['conditions'];
    }
    if($condition_string != '') {
      $condition_string = ' where ' . $condition_string;
    }
  }
  //bind value
  if (array_key_exists('bind', $params)) {
    $bind = $params['bind'];
  }
  //order
  if (array_key_exists('order', $params)) {
    $order = ' order by ' . $params['order'];
  }
  //limit
  if (array_key_exists('limit', $params)) {
    $limit  = ' limit ' . $params['limit'];
  }

  $sql = "select * from {$table}{$condition_string}{$order}{$limit}";
  if ($this->query($sql, $bind)) {
    if (!count($this->_result)) return false;
    return true;
  }
  return false;
}

public function find($table,$params=[]) {
  if ($this->_read($table, $params)) {
    return $this->results();
  }
  return false;
}

public function find_first($table,$params=[]) {
  if ($this->_read($table,$params)) {
    return $this->first();
  }
  return false;
}
```

Figure 8: Database Core File – 2

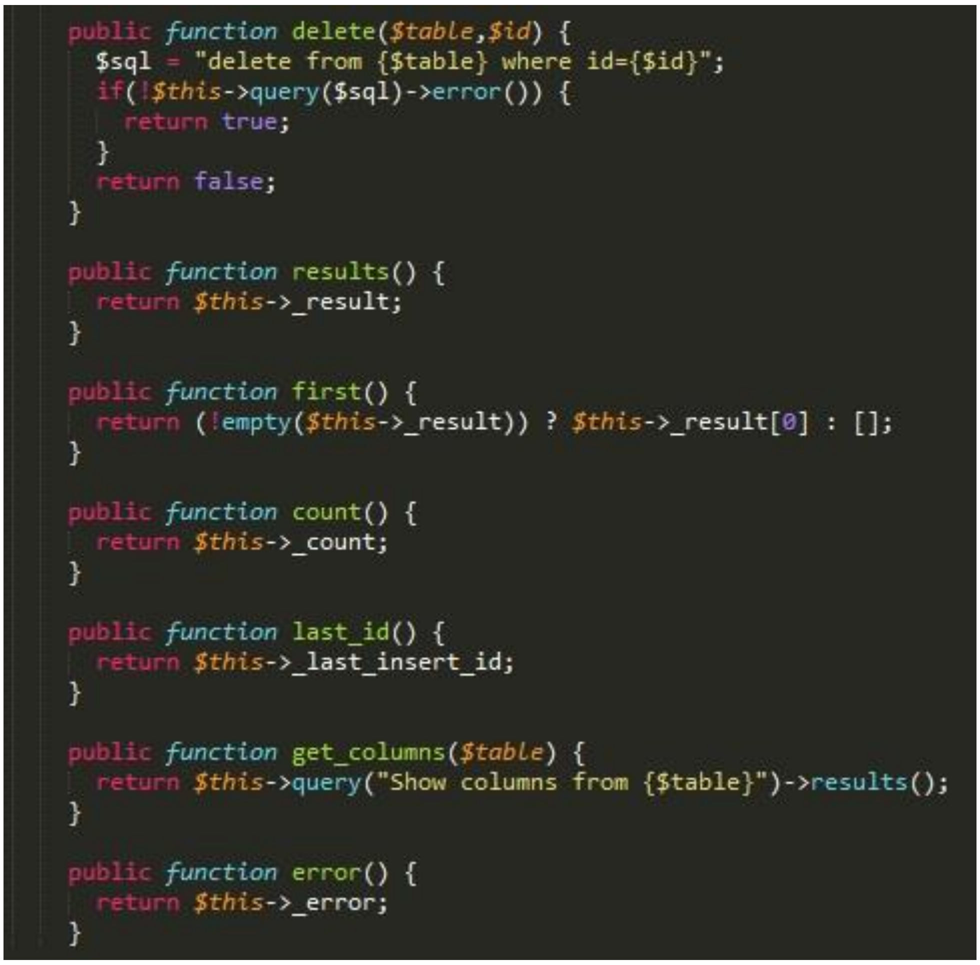
The above shown read function is the select function of the database and is in protected scope, this function takes the table name and parameters to perform the SELECT query on to the database and return results to the user, the parameters of the where condition can be supplied to this function the type of arrays and this would be broken down into the form of string and query would be executed then. Similarly the find function makes use of the read function and returns the results to the user in proper form of associative array. Similarly the find first will only return the first matching result rather than returning all the result rows to the user, this is mostly used in login functions since only one user is needed at that time and returning only one result from the database could come in handy at that moment. These are the SELECT functions of the database and perform the actions very well. These functions can also be modified by the user according to the needs of the application.



```
public function insert($table, $fields = []) {
  $field_string = '';
  $value_string = '';
  $values = [];

  foreach ($fields as $field => $value) {
    $field_string .= '`' . $field . '`,';
    $value_string .='?,';
    $values[] = $value;
  }

  $field_string = rtrim($field_string, ',');
  $value_string = rtrim($value_string, ',');

  $sql = "INSERT INTO {$table} ({$field_string}) VALUES({$value_string})";
  if (!$this->query($sql,$values)->error()) {
    return true;
  }
  return false;
}

public function update($table,$id,$fields=[]) {
  $field_string = '';
  $values = [];
  foreach ($fields as $field => $value) {
    $field_string .= ' ' . $field . ' = ?, ';
    $values[] = $value;
  }
  $field_string = trim($field_string);
  $field_string = rtrim($field_string, ',');
  $sql = "update {$table} set {$field_string} where id = {$id}";
  if(!$this->query($sql,$values)->error()) {
    return true;
  }
  return false;
}
```

Figure 9: Database Core File – 3

The create function of a database transaction is made in the name of insert, this function takes two variables, first one as the table name as to in which table the data is to be inserted and the parameter as an associative array, this associative array would be broken into key value pairs and all the them would be concatenated into one single string of sql query that would be executed. If the query is executed successfully a status of true condition would be returned otherwise the class variables of error would be set to true and a status of false would be returned from this function. The other function in the figure is of update, this update function is used to modify information already present in database. It takes three parameters the first one as table name as to which in which table the record is present, the second parameter as the id of the record that needs to be updated and the third one is associative array of the fields that needs to be updated on that array. It return true or false on whether the record has been updated or not accordingly.



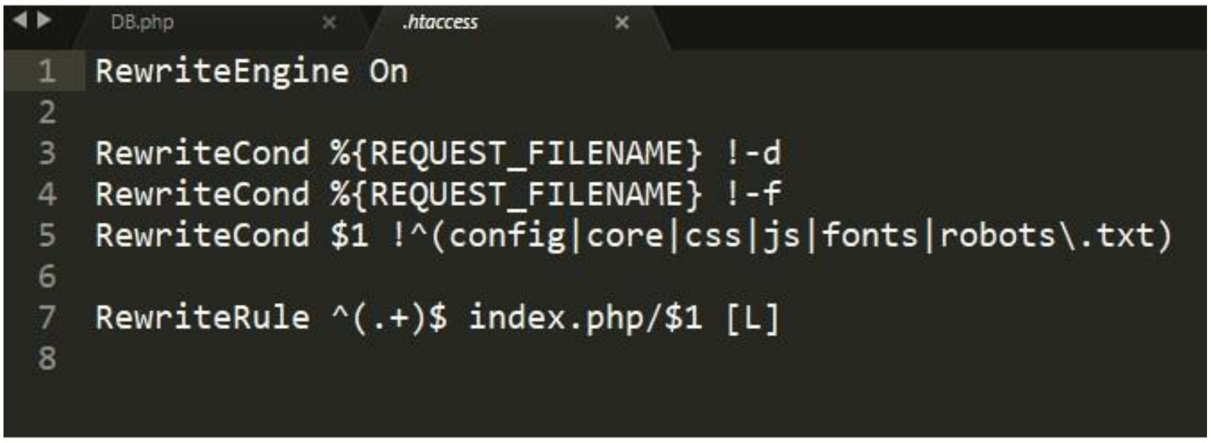
```php
public function delete($table,$id) {
  $sql = "delete from {$table} where id={$id}";
  if(!$this->query($sql)->error()) {
    return true;
  }
  return false;
}

public function results() {
  return $this->_result;
}

public function first() {
  return (!empty($this->_result)) ? $this->_result[0] : [];
}

public function count() {
  return $this->_count;
}

public function last_id() {
  return $this->_last_insert_id;
}

public function get_columns($table) {
  return $this->query("Show columns from {$table}")->results();
}

public function error() {
  return $this->_error;
}
```

Figure 10: Database Core File – 4

The fourth part of the CRUD functionality is the delete method, the delete function takes two parameters and the first one is the table name as to in which table the data would be deleted and the next one is id of the record that needs to be deleted. It is a fairly simple needed that just performs the delete query on the records and return true or false in the same fashion of the query. Then the rest of the functions present are mostly the setters method of the class, they usually set the values of the class variables present and return them. These setters method are necessary since all the class variables are present in protected scope of the class and cannot be accessed outside of class directly by any user. The get columns functions returns the information about a table about its columns so as to the user should know about a table before saving anything here.

The first method just returns the first element in the array and calls the same result function which was called earlier to return the possible sets of rows to a query. It is usually responsible for finding out a single user while logging the user in the application. The last id function gets the last inserted id from the database sets the class variable the value, so as to take advantage of this while creating a new record, and that new records id is needed to perform further operations on the application. That's why these functions are also important to be present in a framework to provide the basic database functionalities to a application.

2.2.5 Htaccess File



```
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond $1 !^(config|core|css|js|fonts|robots\.txt)

RewriteRule ^(.+)$ index.php/$1 [L]

```

Figure 11: Htaccess File

In the above figure an htaccess file is shown, .htaccess is a configuration file for use on web servers running the Apache Web Server software. When a .htaccess file is placed in a directory which is in turn 'loaded via the Apache Web Server', then the .htaccess file is detected and executed by the Apache Web Server software. These .htaccess files can be used to alter the configuration of the Apache Web Server software to enable/disable additional functionality and features that the Apache Web Server software has to offer. These facilities include basic redirect functionality, for instance if a 404 file not found error occurs, or for more advanced functions such as content password protection or image hot link prevention.

'.htaccess' is the filename in full, it is not a file extension. For instance, you would not create a file called, 'file.htaccess', it is simply called, '.htaccess'. This file will take effect when placed in any directory which is then in turn loaded via the Apache Web Server software. The file will take effect over the entire directory it is placed in and all files and subdirectories within the specified directory. If you have a directory containing PHP includes, that you do not wish to be accessed directly from the browser, there is a way of disabling the directory using Mod_Rewrite.

When uploading your .htaccess files, it is very important you upload the file in 'ASCII' mode. 'ASCII' and 'BINARY' are different methods of transferring data and it is important .htaccess files are transferred in 'ASCII' mode and not 'BINARY'. It is likely your FTP software will default to 'BINARY' so look for a 'Transfer Mode' or 'Transfer Type' option in the menus. Upload the .htaccess file to the directory you would like it to take effect over. Now visit this directory using your web browser as you would for any other document on your web site and check it has worked correctly.

2.2.6 URL and Routing in Framework

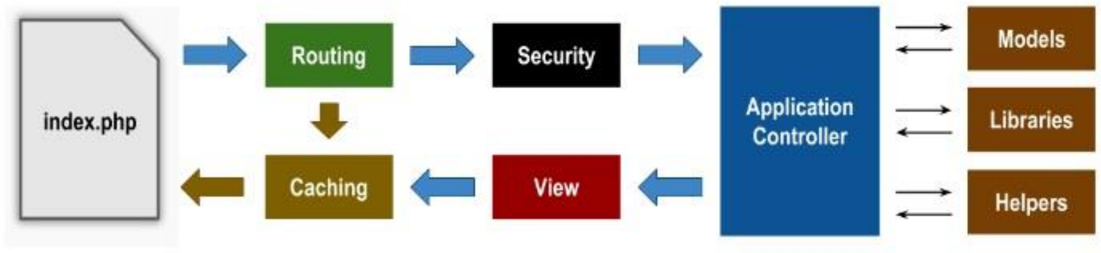


Figure 12: Routing in the framework

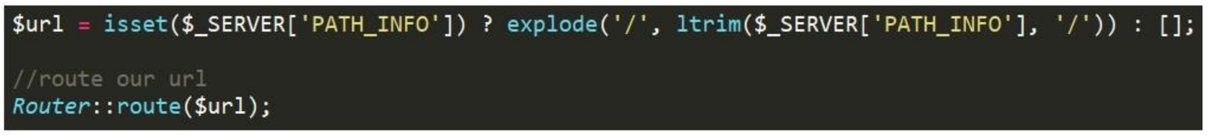
In traditional web applications, Uniform Resource Locators (URLs) usually used standard query string parameters to ‘get’ and ‘post’ data. For example:

http://www.example.com/Blog/Posts.php?Year=2015&Month=10&Day=11

In MVC frameworks, the segment-based approach is mainly used. The segments, also known as Uniform Resource Identifier (URI) segments, are added to the domain name (including any directory path) and represent the controller class, its method, and any arguments belonging to that class method. For example:

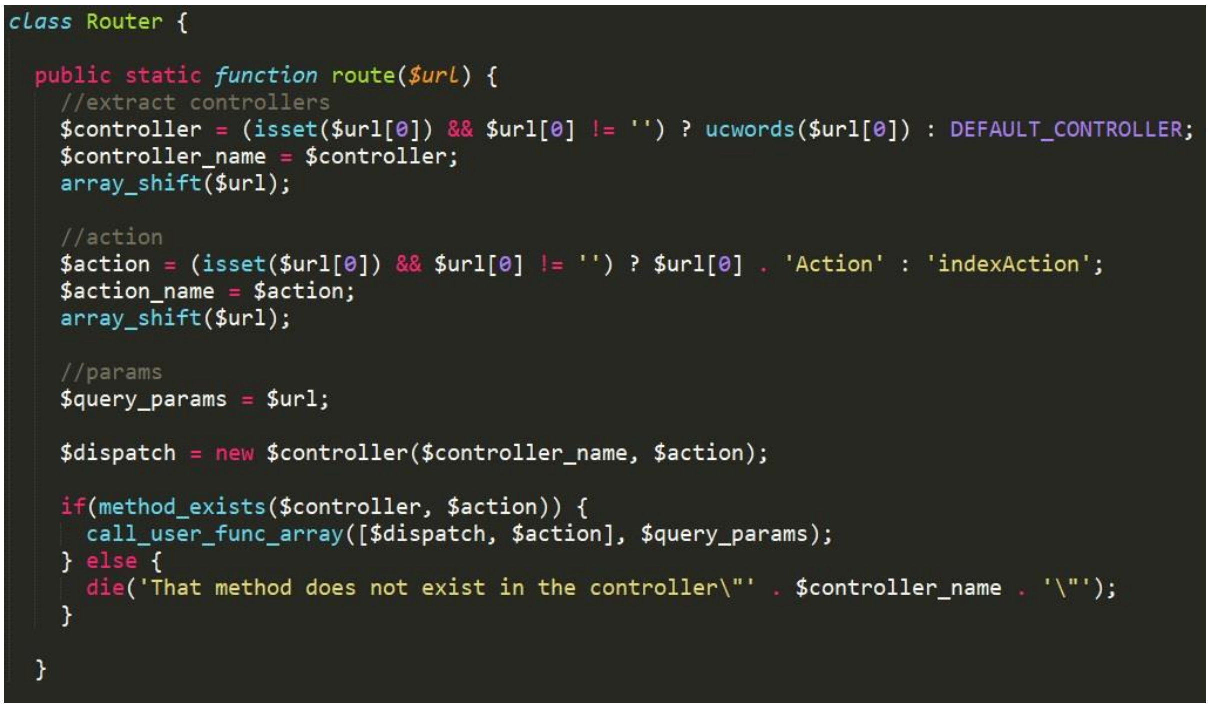
http://www.example.com/blog/posts/2015/10/11

In the above URL, ‘blog’ is the controller class, ‘post’ is its method, and ‘year/month/day’ are the method’s arguments. The arrangement of the segment approach allows URLs to be more human reader, and search engine friendly. When a framework performs routing on a URL, it modifies or rewrites the default action it has with the URI segment. Routing can be set up in the configuration file.



```
$url = isset($_SERVER['PATH_INFO']) ? explode('/', ltrim($_SERVER['PATH_INFO'], '/')) : [];

//route our url
Router::route($url);
```

Figure 13: Route breaking



```php
class Router {

  public static function route($url) {
    //extract controllers
    $controller = (isset($url[0]) && $url[0] != '') ? ucwords($url[0]) : DEFAULT_CONTROLLER;
    $controller_name = $controller;
    array_shift($url);

    //action
    $action = (isset($url[0]) && $url[0] != '') ? $url[0] . 'Action' : 'indexAction';
    $action_name = $action;
    array_shift($url);

    //params
    $query_params = $url;

    $dispatch = new $controller($controller_name, $action);

    if(method_exists($controller, $action)) {
      call_user_func_array([$dispatch, $action], $query_params);
    } else {
      die('That method does not exist in the controller\"' . $controller_name . '\"');
    }

  }
```

Figure 14: Router class and Route Function

In this framework all the requests coming to application folder are routed towards the main index.php file in the main directory, now this files takes all the query parameters ie everything ahead of its host name. Then this query string is sliced into an array by slashes, now the framework knows that the first element in the array is the controller name and the second element in the array is the action name of that controller and everything ahead of that is the query parameters that could be needed by the controller action for its functioning. The action dispatch method in php calls the individual class and the call user func array call the action of the controller method and even provides the parameters to that function. That helps in creating dynamic routing to the controller classes which can then interact with appropriate models and views as per the requirement. Every time an element from the url array is used it is removed from the starting so as there is no confusion in mind of the developers and even makes it easier to develop web applications.

Since this routing is standard among most of the mvc frameworks, it is also easier for experiences developers to move to this framework. Caching or web page caching is a method of storing part of a page (specifically view output) that is not usually subject to a change during refreshes, thus allowing faster subsequent reloads of an app. Caching is done on a page by page basis. Caching is optional and not set by default, and therefore requires setting in a controller’s method using a number of minutes as the expiration argument to its method call.

2.3 Security

To protect web applications against malicious users, the programmer must be aware of numerous kinds of possible vulnerabilities and countermeasures. Among the most popular guidelines for programming safe web applications are those in the OWASP. Top 10 report that covers “the 10 most critical web application security risks”. In the OWASP Report, vulnerabilities are not necessarily the result of an incorrect program. Incorrect setup of SSL certificates, failure to use encrypted transport protocols, and insecure data storage are also included in the report. While such issues are in general important to consider, this dissertation focuses on program analysis and we will therefore only discuss the vulnerabilities that are directly related to program errors. The three categories injection, cross-site scripting, and insecure direct object references are of particular interest to this dissertation. The two OWASP categories injection and cross-site scripting have been in particular been subject to research. A third OWASP category insecure direct object references is related to the two other categories in that all three categories of vulnerabilities are caused by trust in data sent from the client. In all three cases, the vulnerability could be avoided by inspecting the value before use. The class of client-state manipulation vulnerabilities are a superset of insecure direct object references. This category is not limited to values that act as database identifiers. It includes all values that are stored on the client side and returned to the server in a subsequent request. Changing such a value could lead to unexpected behaviour on the server.

2.3.1 Injection

According to OWASP Top 10 report, injection vulnerabilities can occur when untrusted data is sent to an interpreter as part of a command or query. The malicious client can exploit such a vulnerability to change the effect of the query arbitrarily to gain access to or to change data. An example of an injection vulnerability is shown in Figure 2.2. The programmer intends to query the database to validate the username and password of a client. He inserts the parameter strings from the client into the SQL query and runs the query on the database to see whether a matching user exists. A malicious user can provide the strings admin';-- for the username parameter and the empty string for the password parameter to generate the query string SELECT * FROM user WHERE username='admin';--' AND password=''. In SQL starts a comment and in effect the malicious client will be logged in as admin without a password check.

The problem can be avoided by not inserting the untrusted string into the query directly. There are multiple solutions to this. One solution is to use prepared statements where the programmer writes the database SQL query with place-holders instead of values and lets the database framework be responsible for inserting the escaped values correctly into the queries.

2.3.2 Cross-site scripting

Cross-site scripting vulnerabilities occur when untrusted data is send to a web browser without proper validation or escaping. In a vulnerable application, the malicious client can gain control over the generator by constructing data that contains HTML tags. If such data is inserted into the document, it will be interpreted by the client in a way that was not intended by the application programmer. Effectively, such a vulnerability allows the malicious client to change the structure of the document and manipulate the way the page is presented to other clients. A malicious user can furthermore exploit a cross-site scripting vulnerability to execute JavaScript in the browser.

2.3.3 Client-state manipulation

Client-state manipulation vulnerabilities occur when the server stores trusted data as part of the document on the client side and expects this data to remain unchanged from one request to another. In Section 4.3 we furthermore argue that vulnerabilities are a result of updating or querying the store with untrusted data. Such vulnerabilities are similar to injection and cross-site scripting vulnerabilities in that untrusted data is used on the server without checking its contents. Client-state manipulation vulnerabilities are distinct in that the untrusted data is neither sent to a query as in injection vulnerabilities or to the output to a web browser as in cross-site scripting vulnerabilities. Rather, the vulnerability allows the malicious to change or read unauthorized data on the server. In client-state manipulation attacks, this is done by making the web application accept a modified value of a parameter rather than changing the structure of a command issued by the web server. Vulnerability detecting involves both an analysis of the output to determine which parameters contain client-state and an information flow analysis to track the values to where they are used in the application.

2.3.4 Safety in PHP applications

PHP offers only very limited support for avoiding common web application errors. It is up to the programmer to ensure that cross-site scripting attacks are not possible. The PHP framework is therefore not safe by default against such vulnerabilities. Similarly, the SQL API of PHP is typically used in conjunction with SQL queries that are generated through string concatenation . The programmer must remember to escape client provided values that become part of such query strings. PHP does provide an API for prepared statements. If the programmer uses this API exclusively then the application is free of SQL injection vulnerabilities. PHP has no support for avoiding client-state manipulation vulnerabilities.

CHAPTER 3: SAMPLE APPLICATION ON FRAMEWORK

The sample application based on the framework is a series application, in the application the user would come in signup or login and can watch his favourite series or movie online. The class diagram for the application is as follows.

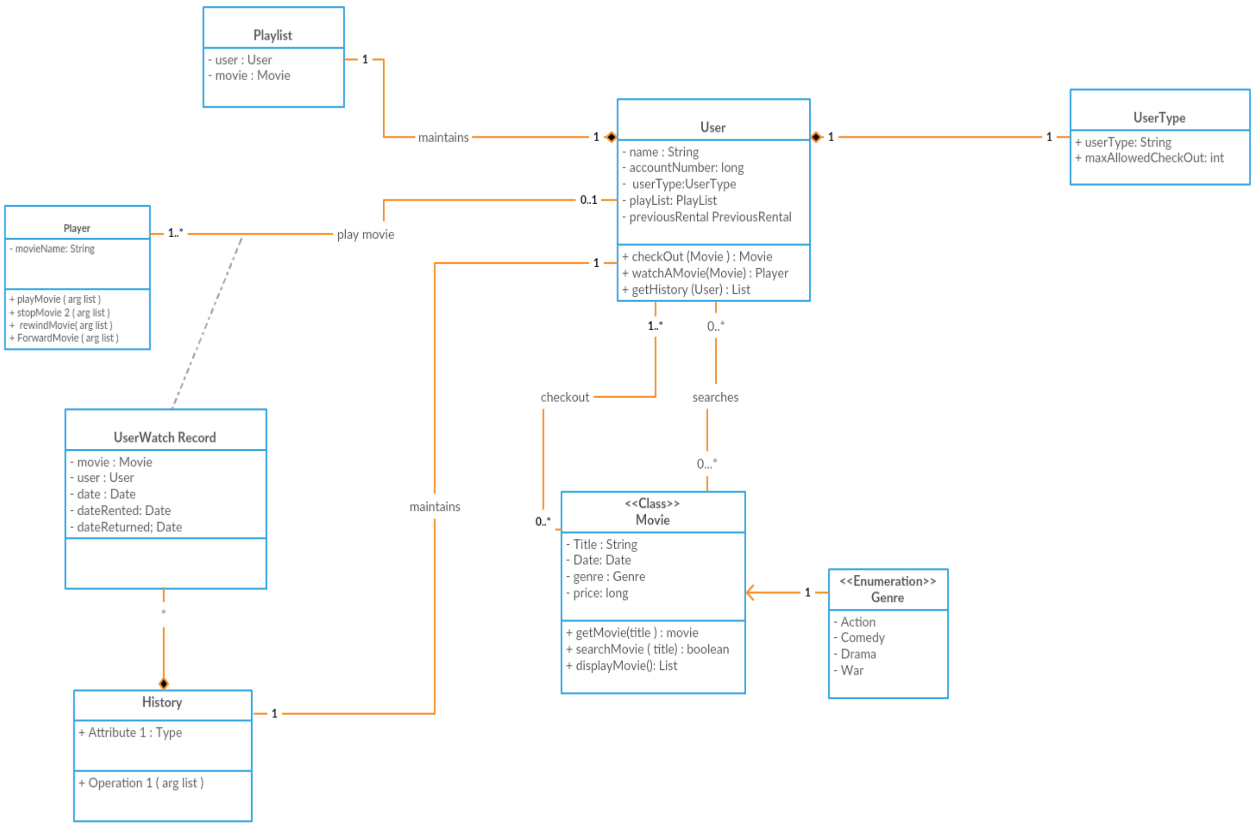


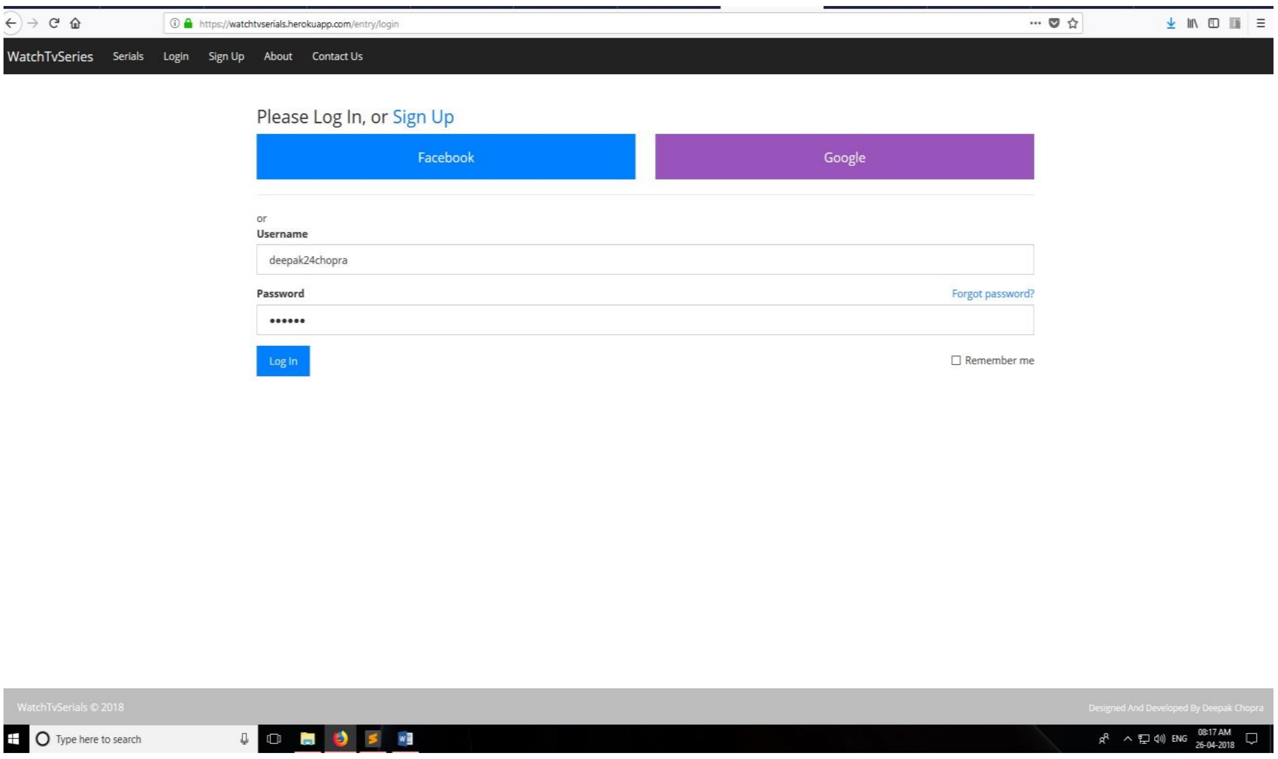
Figure 15: Class Diagram

The above application has seven different modules in the application, all of them having different requirements from the database and different needs. Each of this module has its own table in the database, these modules has different models so as to lower the coupling in the application, for increasing the cohesion they are mostly self-dependent. The models in the application are:-

1. Serials/Movies
2. Seasons
3. Episodes
4. Admin User
5. Comment
6. Contact
7. User

3.1 Admin User Module

The admin user module is responsible for managing the admin level users of the application, these users on the application manage the data that needs to be added, the movies or the series that needs to be added on the user.


https://watchtvserials.herokuapp.com/entry/login

WatchTvSeries Serials Login Sign Up About Contact Us

Please Log In, or Sign Up

Facebook Google

or

Username: deepak24chopra

Password: •••••• Forgot password?

Log In ☐ Remember me

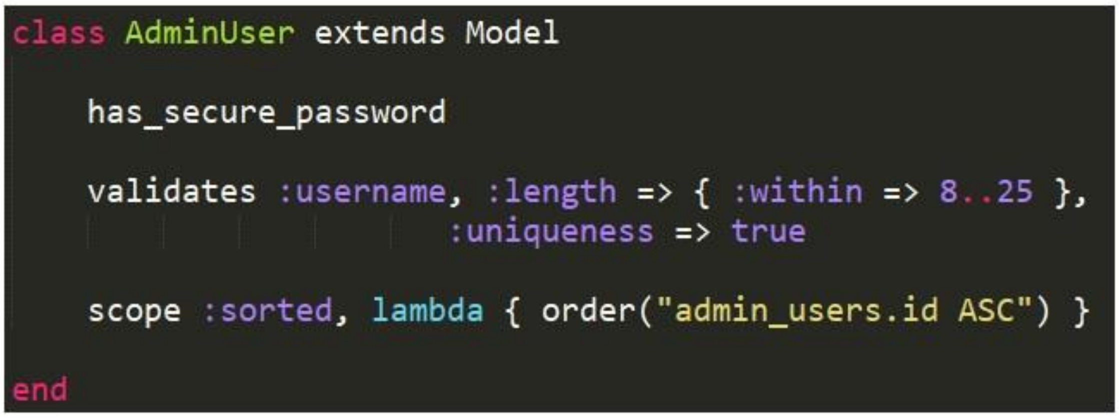
WatchTvSerials © 2018 Designed And Developed By Deepak Chopra

Figure 16: Admin User Login View

The above shown is view file of the admin login, this file either logs in the admin into the backend logic of the application or not, this admin user module has its own controller and a seprate set of views. The admin_user table in the database consists of five main fields:-

• Id	-	Integer
• Username	-	String
• Email	-	String
• Password	-	String
• Created_at	-	DateTime
• Updated_at	-	DateTime

The password of the users in the database are stored as an encrypted string, so even if someone has access to the database won't know the passwords of other users. After logging into the backend of the application the admin reaches the next page where it can see the lists of the movies, series and can easily manage them accordingly. All the new entries to this table in the database is managed by admin controller, it is not necessary to name same as the model that the controller is associated with, but it is good practice to name the controller something same or related to the model. The admin model consist of following code:-



```
class AdminUser extends Model

    has_secure_password

    validates :username, :length => { :within => 8..25 },
                        :uniqueness => true

    scope :sorted, lambda { order("admin_users.id ASC") }

end
```

Figure 17: Admin User Model Code

The above code includes a function named “has_secure_password”, this function always checks the password string and encrypts it before saving it to database, this function would always be called before the create method or as well as before the update method. Then there is validations in the model so that there is inconsistency in database. It validates whether the username has the length between 8 to 25 characters, and each and every user should had a unique username. A unique username in database will ensure that two users with the same id cannot login in backend logic of the application. Admin after login reaches a page where he can manage content of the application with single click to any module.

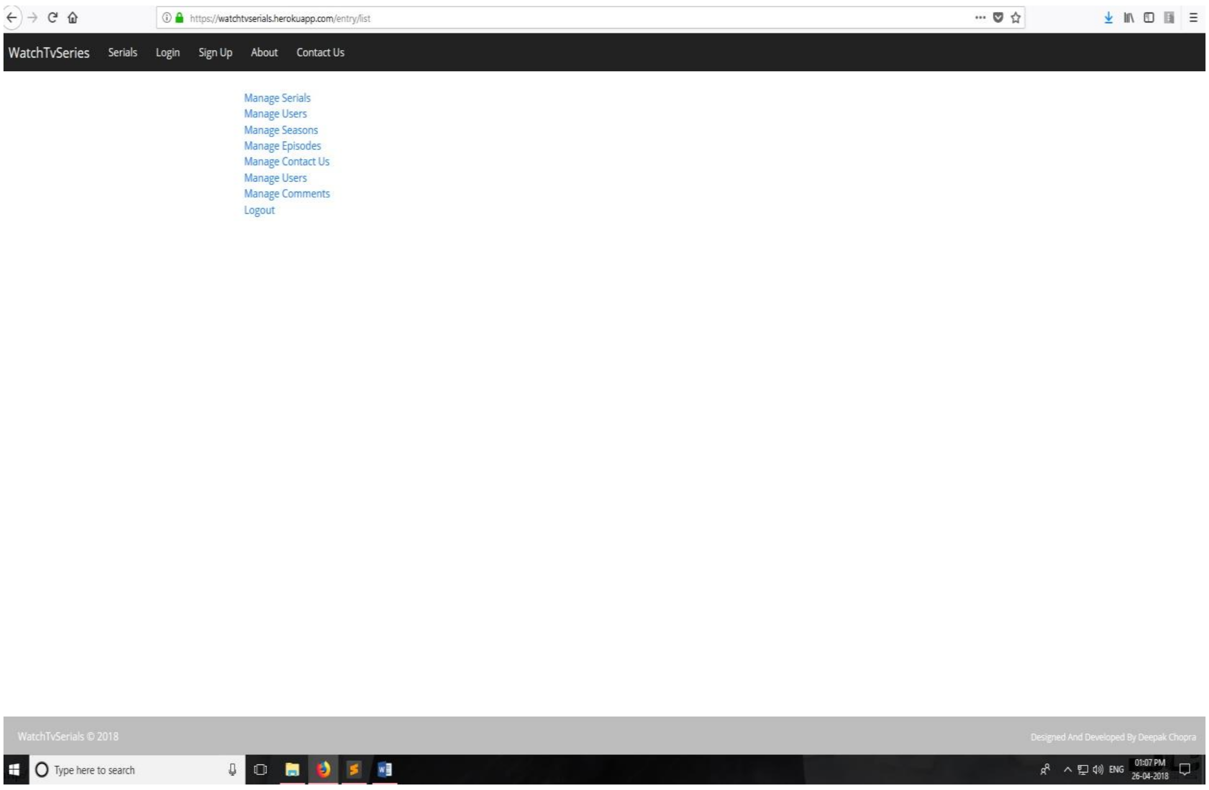
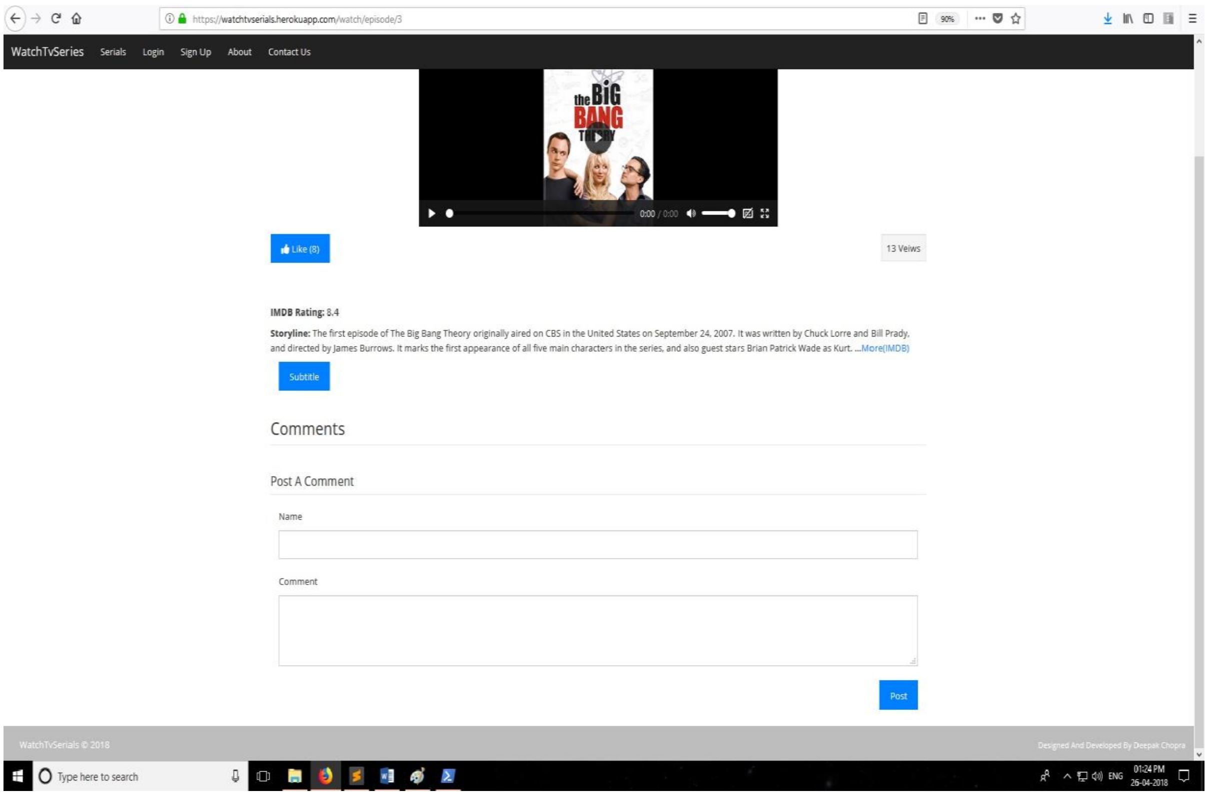


Figure 18: Admin User Dashboard Page

3.2 Comment Module

The comment module in the application is responsible for each and every comments on the episodes of the tv series or a movie file. A normal user of the application can come to the website and share his/her points, reviews or comments on a particular video file or on the whole series. The application is also open for overall suggestion for the data flow or suggestions by users to change the UI or in some change the functionality of the application, or to add or remove new features to the application. Since a new project always on its beta stage needs user reviews so it can improve or remove features. Since no application can be fully complete and there is always room for improvement, user reviews is an important part of the application. The comment module consists of following fields in database:

• Id	-	Integer
• Content	-	Text
• Episode_id	-	Integer
• User_id	-	Integer
• Created_at	-	DateTime
• Updated_at	-	DateTime

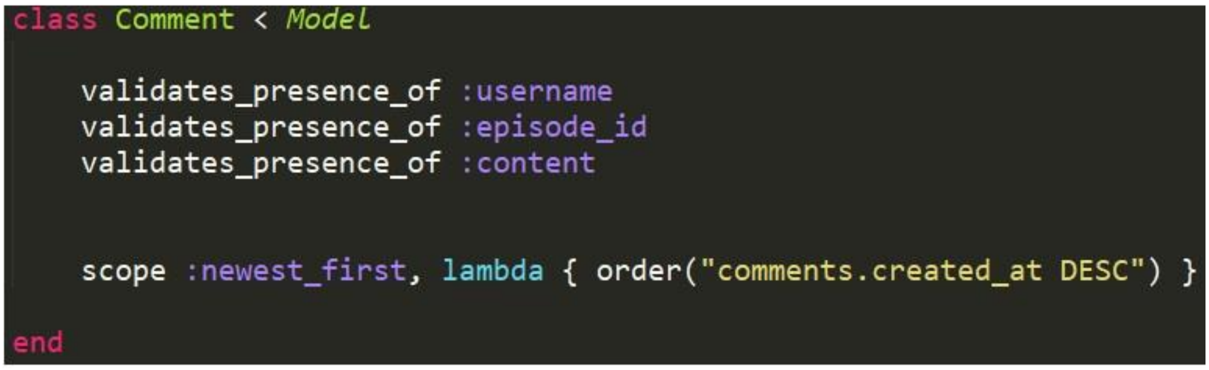


IMDB Rating: 8.4

Storyline: The first episode of The Big Bang Theory originally aired on CBS in the United States on September 24, 2007. It was written by Chuck Lorre and Bill Prady, and directed by James Burrows. It marks the first appearance of all five main characters in the series, and also guest stars Brian Patrick Wade as Kurt. ...More(IMDB)

Figure 19: Comment Form

This comment form is related to each and every episode in the application. So this has 1:n relationship among them, a single episode on the web application can have multiple comments. These comments also have a 1:n relationship with the user like a single comment belongs to only one user but one user can have a lot of comments on different files. Although the comment module is not a large one but it still needs its own model and controller, since it is good practice to create models of entity that have a table in database.



```
class Comment < Model

    validates_presence_of :username
    validates_presence_of :episode_id
    validates_presence_of :content


    scope :newest_first, lambda { order("comments.created_at DESC") }

end
```

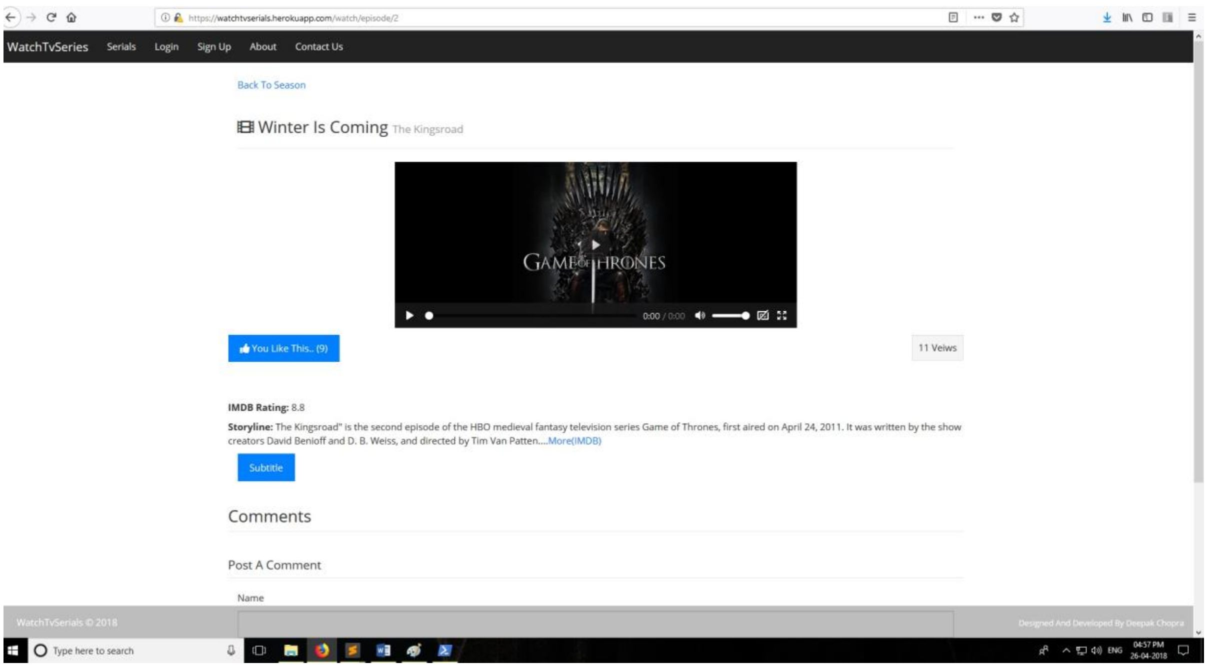
Figure 20: Comment Model Code

The comment model has a few validations before saving the entry to the database. The validates presence of method doesn't allow empty values to be saved in database. This method keeps check the entry of a comment does have its user, the episode it was commented on and the content of the comment. These fields cannot be sent empty to database. After the successful submission of a comment row the page is redirected to the same episode so that there is no inconsistency between the flow of application.

3.3 Episode Module

The episode modules is responsible for serving the individual video files to the browser of the user. This module has a lot of fields in the database since it contains a lot of information about a single video file which could be an episode of a series or a movie file. Episode module consists of following fields:-

• Id	-	Integer
• Video_url	-	String
• Torrent_url	-	String
• Subtitle_url	-	String
• Image_url	-	String
• Episode_number	-	Integer
• Name	-	String
• Description	-	Text
• Storyline	-	Text
• Likes	-	Integer
• Air_date	-	DateTime
• Created_at	-	DateTime
• Updated_at	-	DateTime



https://watchtvserials.herokuapp.com/watch/episode/2

WatchTvSeries Serials Login Sign Up About Contact Us

Back To Season

Winter Is Coming The Kingsroad

You Like This.. (9)

11 Veiws

IMDB Rating: 8.8

Storyline: The Kingsroad" is the second episode of the HBO medieval fantasy television series Game of Thrones, first aired on April 24, 2011. It was written by the show creators David Benioff and D. B. Weiss, and directed by Tim Van Patten....More(IMDB)

Subtitle

Comments

Post A Comment

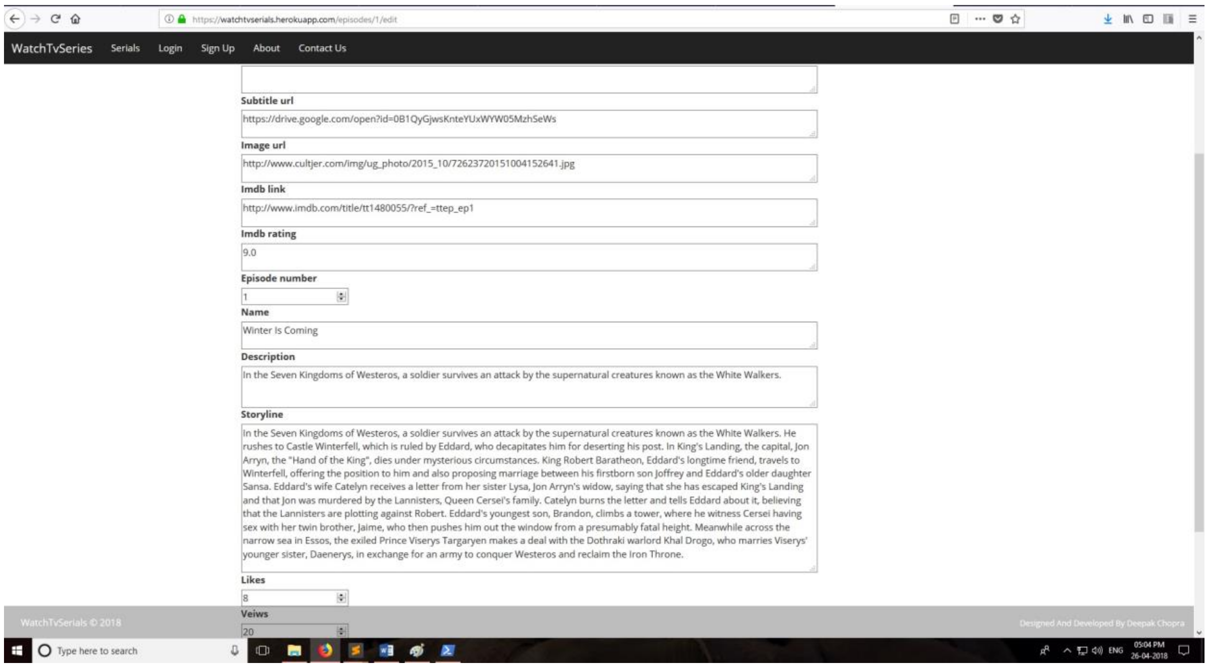
Name

WatchTvSerials © 2018

Designed And Developed By Deepak Chopra

Figure 21. Episode Frontend View

The frontend logic of the episode module, in here the user can even like the video files, then a record of number of likes a kept. The video file from the video url is being loaded into a flash media player to watch at this page. This module even has the ability to provide user with the download the subtitles of the video and torrent file download support. Further if you play the video file you can enter a full screen mode and even the matching subtitles to the video file can be straight loaded there.



https://watchtvserials.herokuapp.com/episodes/1/edit

WatchTvSeries Serials Login Sign Up About Contact Us

Subtitle url
https://drive.google.com/open?id=0B1QyGjwsKnteYUxWYW05MzhSeWs

Image url
http://www.cultjer.com/img/ug_photo/2015_10/72623720151004152641.jpg

Imdb link
http://www.imdb.com/title/tt1480055/?ref_=ttep_ep1

Imdb rating
9.0

Episode number
1

Name
Winter Is Coming

Description
In the Seven Kingdoms of Westeros, a soldier survives an attack by the supernatural creatures known as the White Walkers.

Storyline
In the Seven Kingdoms of Westeros, a soldier survives an attack by the supernatural creatures known as the White Walkers. He rushes to Castle Winterfell, which is ruled by Eddard, who decapitates him for deserting his post. In King's Landing, the capital, Jon Arryn, the "Hand of the King", dies under mysterious circumstances. King Robert Baratheon, Eddard's longtime friend, travels to Winterfell, offering the position to him and also proposing marriage between his firstborn son Joffrey and Eddard's older daughter Sansa. Eddard's wife Catelyn receives a letter from her sister Lysa, Jon Arryn's widow, saying that she has escaped King's Landing and that Jon was murdered by the Lannisters, Queen Cersei's family. Catelyn burns the letter and tells Eddard about it, believing that the Lannisters are plotting against Robert. Eddard's youngest son, Brandon, climbs a tower, where he witness Cersei having sex with her twin brother, Jaime, who then pushes him out the window from a presumably fatal height. Meanwhile across the narrow sea in Essos, the exiled Prince Viserys Targaryen makes a deal with the Dothraki warlord Khal Drogo, who marries Viserys' younger sister, Daenerys, in exchange for an army to conquer Westeros and reclaim the Iron Throne.

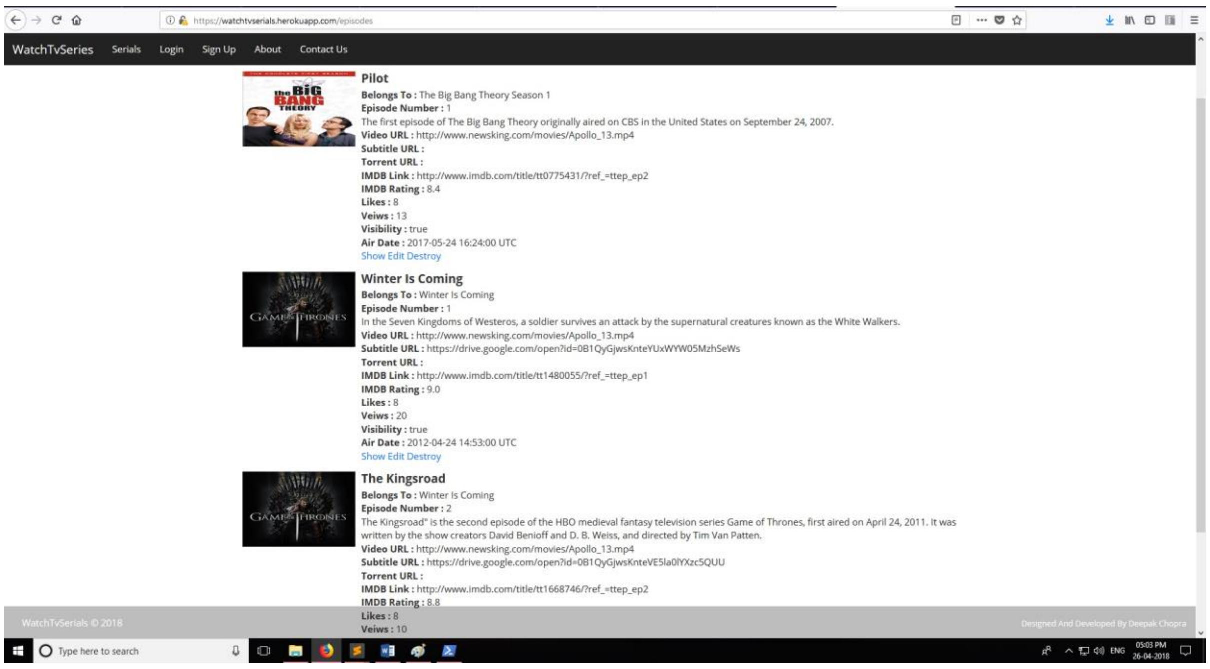
Likes
8

Veiws
20

WatchTvSerials © 2018

Designed And Developed By Deepak Chopra

Figure 22. Episode Adding and Update Form



Pilot
Belongs To : The Big Bang Theory Season 1
Episode Number : 1
The first episode of The Big Bang Theory originally aired on CBS in the United States on September 24, 2007.
Video URL : http://www.newsking.com/movies/Apollo_13.mp4
Subtitle URL :
Torrent URL :
IMDB Link : http://www.imdb.com/title/tt0775431/?ref_=ttep_ep2
IMDB Rating : 8.4
Likes : 8
Veiws : 13
Visibility : true
Air Date : 2017-05-24 16:24:00 UTC
Show Edit Destroy

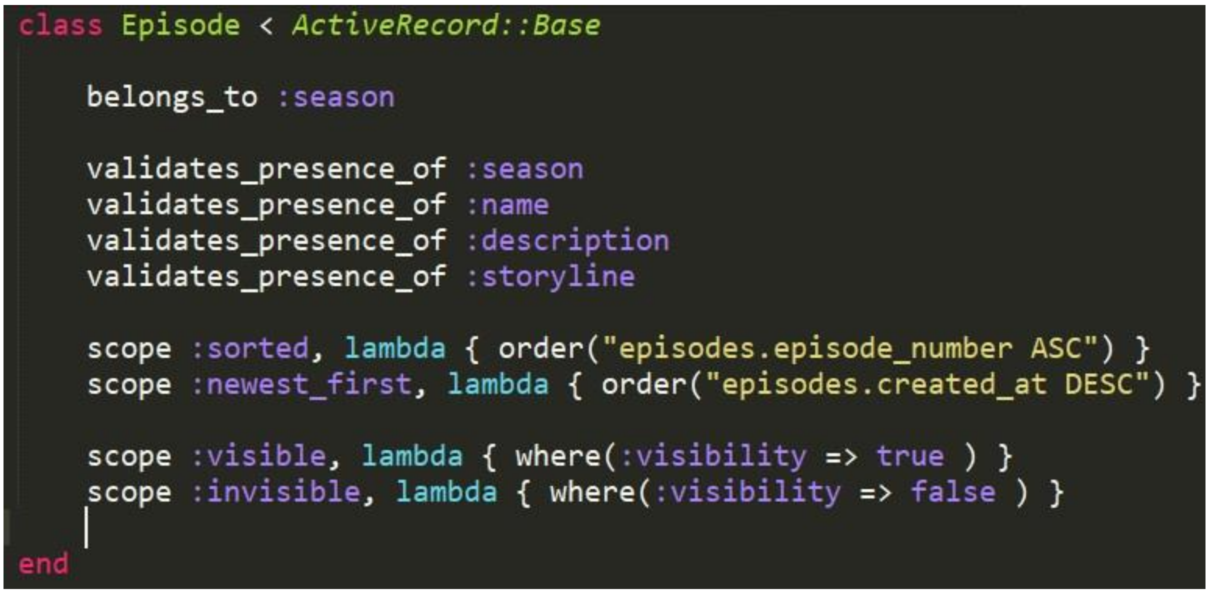
Winter Is Coming
Belongs To : Winter Is Coming
Episode Number : 1
In the Seven Kingdoms of Westeros, a soldier survives an attack by the supernatural creatures known as the White Walkers.
Video URL : http://www.newsking.com/movies/Apollo_13.mp4
Subtitle URL : https://drive.google.com/open?id=0B1QyGjwsKnteYUxWYW05MzhSeWs
Torrent URL :
IMDB Link : http://www.imdb.com/title/tt1480055/?ref_=ttep_ep1
IMDB Rating : 9.0
Likes : 8
Veiws : 20
Visibility : true
Air Date : 2012-04-24 14:53:00 UTC
Show Edit Destroy

The Kingsroad
Belongs To : Winter Is Coming
Episode Number : 2
The Kingsroad" is the second episode of the HBO medieval fantasy television series Game of Thrones, first aired on April 24, 2011. It was written by the show creators David Benioff and D. B. Weiss, and directed by Tim Van Patten.
Video URL : http://www.newsking.com/movies/Apollo_13.mp4
Subtitle URL : https://drive.google.com/open?id=0B1QyGjwsKnteVE5la0lYXzc5QUU
Torrent URL :
IMDB Link : http://www.imdb.com/title/tt1668746/?ref_=ttep_ep2
IMDB Rating : 8.8
Likes : 8
Veiws : 10

WatchTvSerials © 2018
Designed And Developed By Deepak Chopra

Figure 23. Backend Episode Management

There are two main backend view pages of episode module, the first one is the form where a new episode can be added or an old episode present in the database can be modified. This is a form page containing all the fields of the episode. The next main backend page is the episode management page where you can come and see a list of episodes and the details about them. A delete button is provided to each and every episode that kinds of deletes the episode from the database. This Episode module has a separate controller that performs all the action related to the episode. This controller is responsible for performing the CRUD functionality on the episode module. It even serves the views on the backend of the application.



```ruby
class Episode < ActiveRecord::Base

    belongs_to :season

    validates_presence_of :season
    validates_presence_of :name
    validates_presence_of :description
    validates_presence_of :storyline

    scope :sorted, lambda { order("episodes.episode_number ASC") }
    scope :newest_first, lambda { order("episodes.created_at DESC") }

    scope :visible, lambda { where(:visibility => true ) }
    scope :invisible, lambda { where(:visibility => false ) }

end
```

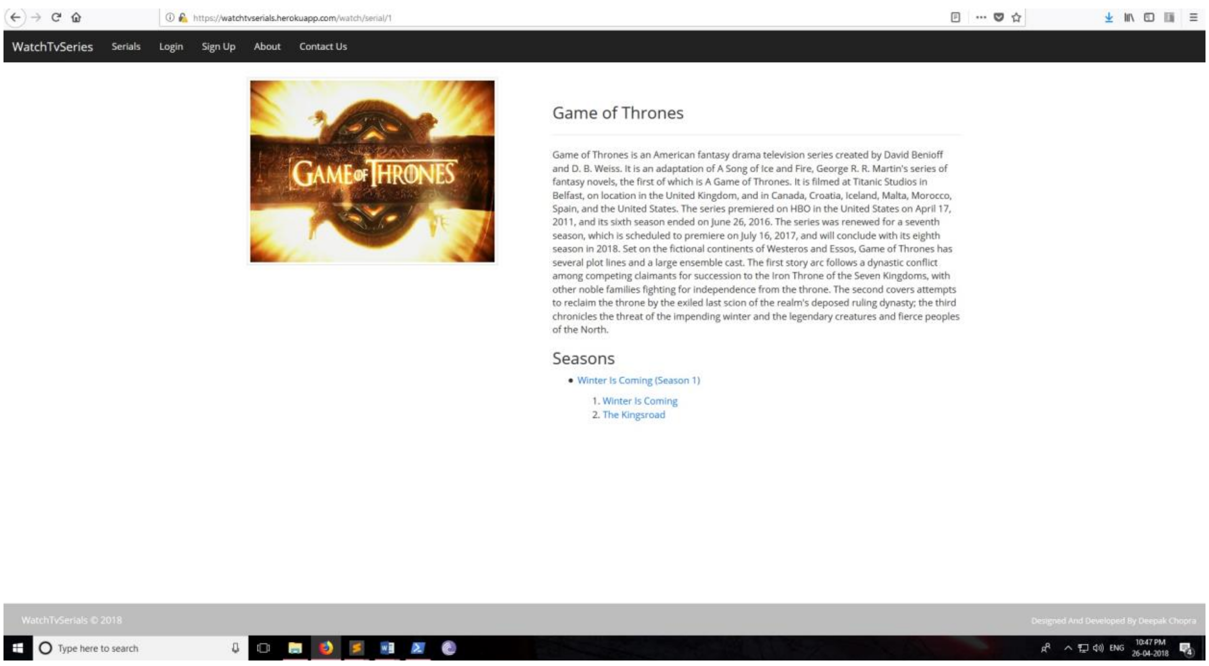
Figure 24. Episode model code

This is the code of the episode model and the function belongs_to in the line creates a relationship among a single episode and that episode belongs to the next module the season module. It performs that basic functionality of before saving the entry into the database that name string cannot be empty, or the description string cannot be sent empty and the storyline field of the episode cannot be empty. The scope functions defined above are used to sort the given results according to the requirements. Visible sorted and newest first are the scope functions defined for episodes in the model.

3.4 Seasons Module

Each and every web series or a tv series are always aired in series. So it was important to have a separate module for season that holds all the related information about a single season of a series. This module is related to serial module in 1:n relationship, one serial can have a lot of seasons and opposite goes for episodes, one season of a series can a lot of episodes related to it. Season module have its own controller and a set of views for the frontend logic of application. Season table in the database consist of following fields:-

• Id	-	Integer
• Serial_id	-	Integer
• Image_url	-	String
• Season_number	-	Integer
• Name	-	String
• Story	-	Text
• Created_at	-	DateTime
• Updated_at	-	DateTime



Game of Thrones

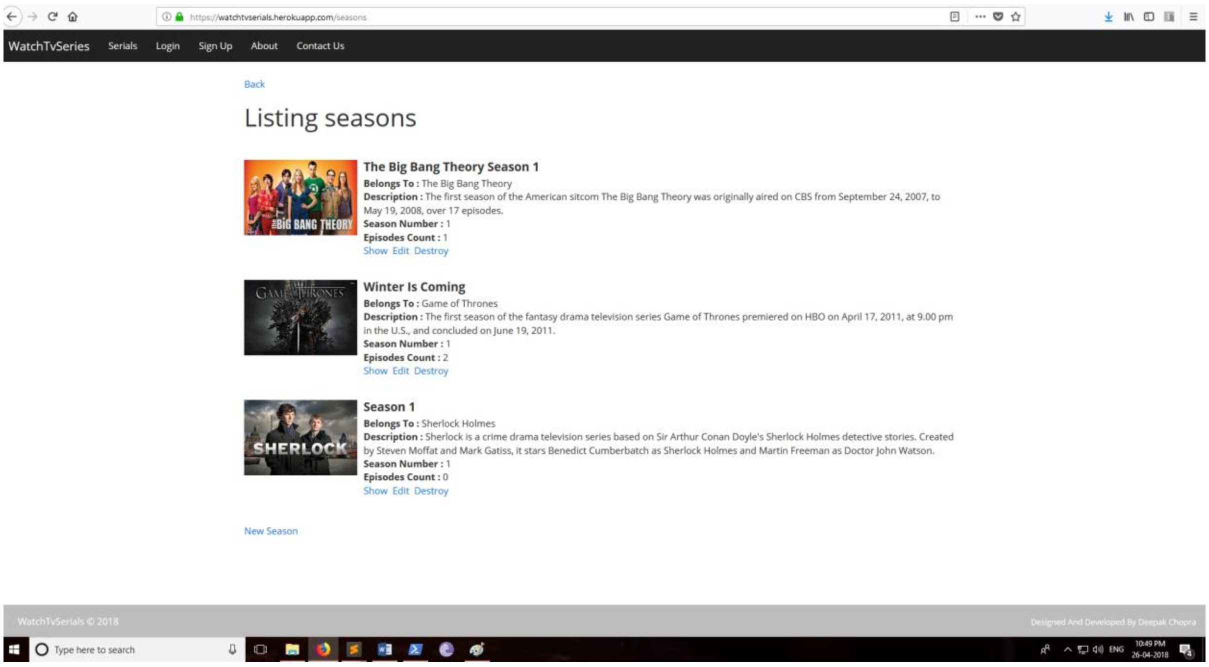
Game of Thrones is an American fantasy drama television series created by David Benioff and D. B. Weiss. It is an adaptation of A Song of Ice and Fire, George R. R. Martin's series of fantasy novels, the first of which is A Game of Thrones. It is filmed at Titanic Studios in Belfast, on location in the United Kingdom, and in Canada, Croatia, Iceland, Malta, Morocco, Spain, and the United States. The series premiered on HBO in the United States on April 17, 2011, and its sixth season ended on June 26, 2016. The series was renewed for a seventh season, which is scheduled to premiere on July 16, 2017, and will conclude with its eighth season in 2018. Set on the fictional continents of Westeros and Essos, Game of Thrones has several plot lines and a large ensemble cast. The first story arc follows a dynastic conflict among competing claimants for succession to the Iron Throne of the Seven Kingdoms, with other noble families fighting for independence from the throne. The second covers attempts to reclaim the throne by the exiled last scion of the realm's deposed ruling dynasty; the third chronicles the threat of the impending winter and the legendary creatures and fierce peoples of the North.

Seasons

- Winter Is Coming (Season 1)
  1. Winter Is Coming
  2. The Kingsroad

Figure 25. Season Frontend view

Season frontend logic consists of three main view files, the first one is for presenting the single season and its fields. In this view s single and all the episodes related to the season are presented in the list orders with the scope of newest first. This scope comes in handy since the latest upload will come first.



Listing seasons

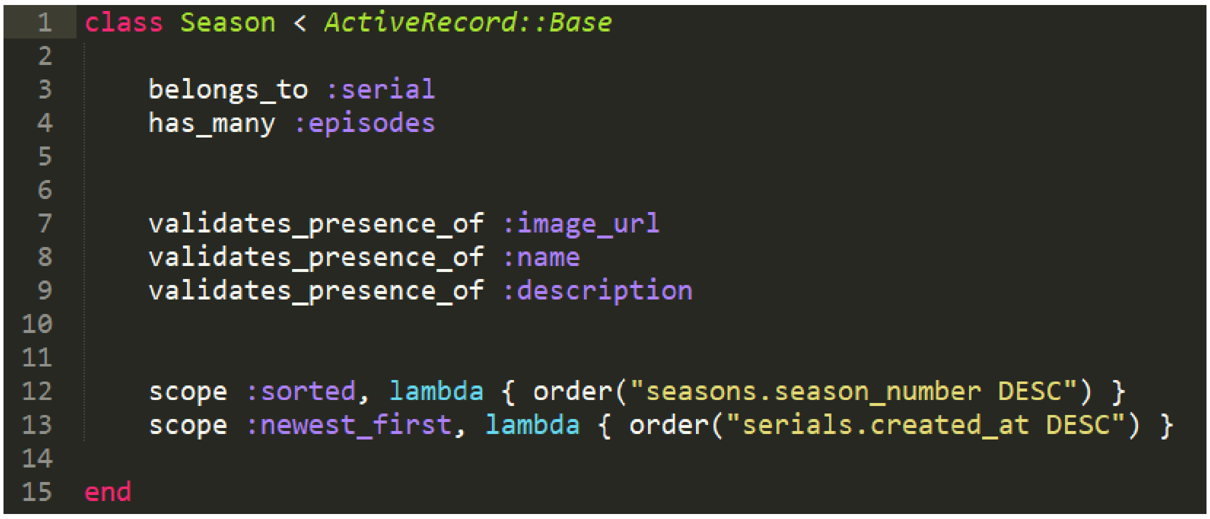
The Big Bang Theory Season 1
Belongs To : The Big Bang Theory
Description : The first season of the American sitcom The Big Bang Theory was originally aired on CBS from September 24, 2007, to May 19, 2008, over 17 episodes.
Season Number : 1
Episodes Count : 1
Show Edit Destroy

Winter Is Coming
Belongs To : Game of Thrones
Description : The first season of the fantasy drama television series Game of Thrones premiered on HBO on April 17, 2011, at 9.00 pm in the U.S., and concluded on June 19, 2011.
Season Number : 1
Episodes Count : 2
Show Edit Destroy

Season 1
Belongs To : Sherlock Holmes
Description : Sherlock is a crime drama television series based on Sir Arthur Conan Doyle's Sherlock Holmes detective stories. Created by Steven Moffat and Mark Gatiss, it stars Benedict Cumberbatch as Sherlock Holmes and Martin Freeman as Doctor John Watson.
Season Number : 1
Episodes Count : 0
Show Edit Destroy

Figure 26. List View of Seasons

The above view will present the seasons in a list order, the admin of the application has the ability the make a season with a Boolean field invisible. In above view the admin of the application can delete a season or edit the information about the season. Addition of a new season is also done from this page only. There is link at the end of a page that leads to the form of addition of new serial. There is pagination among this view, it means as soon as more than 10 serials would be present in the database it will start creating pages to next and previous lists.



```ruby
class Season < ActiveRecord::Base

    belongs_to :serial
    has_many :episodes


    validates_presence_of :image_url
    validates_presence_of :name
    validates_presence_of :description


    scope :sorted, lambda { order("seasons.season_number DESC") }
    scope :newest_first, lambda { order("serials.created_at DESC") }

end
```

Figure 27. Season Model Code

In the above shown image the season model code is present. The relationship to serial and among the episodes is shown in the initial two lines and they are responsible for relation each other. The other lines of code is from the validation library ensures that each value saved into the database does have some basic values and the fields like image, name and description of a season does not go empty into the database. Then there is scope present that is responsible for sorting out the rows that are returned by database during a query execution process.

CHAPTER 4: RESULT AND DISCUSSION

As the above screenshots depicts that applications made on this framework are better on time efficiency and requests made on this framework are completed a lot faster than that done on other frameworks. The whole reason that why these requests are completed a lot faster is that there are only a minimum number of classes present in the core framework. All the other classes are based on the requirement of the application, so that there is no un-necessary load on the memory of the server and the requests respond at a lot faster rates than normal. Any level of developer can kick-start an application on this framework with a little to no knowledge of PHP. The development and understanding of framework is a lot easier on this framework that's what makes it more accessible to other developers. The level of simplicity of this indicates that as per the requirements the modification of core classes is easy as well with reading a little of documentation so as to change in these files do not lead to the breakdown of the whole framework and leading to un necessary errors and reports. The sample application on this framework responds at a lot faster rate on a free server of heroku as well, that's what makes this framework better than others.

REFERENCES

Books:

- PHP 7 packt publishing
- Internetworking Technologies - An Engineering Perspective (2002)
- Apache Cookbook

Web Sites:

- http://www.lynda.com (For PHP tutorials)
- https://www.wamp.com (Windows apache mysql php)