The Development of Accreditation App with MVC Architectural Pattern

Maksy Sendiang
Information Technology Dept
Manado State Polytechnic
North Sulawesi - Indonesia

Maryke Alelo
Tourism Dept
Manado State Polytechnic
North Sulawesi - Indonesia

Venny Ponggawa
Electrical Engineering Dept
Manado State Polytechnic
North Sulawesi - Indonesia

ABSTRACT
Accreditation of educational institutions becomes a gateway for measuring the quality of education services run by educational institutions. It takes great energy to prepare accreditation documents, so the presence of software to organize accreditation documents, making initial assessments of existing documents becomes an inevitable requirement. This paper contains the development of accreditation applications using Model View Controller (MVC) pattern. MVC pattern is used with consideration of the number of entities and data that exist in the accreditation process so that required tools that can be developed and maintained flexibly. Applications are developed using RUP methods that support object-oriented concepts and their implementation using PHP Data object (PDO).

General Terms
Web technology, object oriented programming, database

Keywords
MVC, RUP, accreditation, PDO

1. INTRODUCTION
Data and information has become a very important asset in both government and private organizations nowadays. Every day large amounts of data stored using hardware or software . Usage and rapid progress of the Internet infrastructure , has spurred the increasing amount of data stored in a database lately . Increasing the number of users and the heavy reliance on digital information is one barometer of the importance of securing data or information [1].

Web applications have become the interface is widely used in presenting data and information. Most web applications using multitier design that consists of a presentation tier (front end) , application tier (middle tier) and a data tier (backend) [2]. Presentation tier is the leading layer that presents information regarding to services presented by the web application. Application layer is a layer that implements software functionality by performing the process in detail. . The data tier is a layer that stores data and provides a response to a request from the application tier and consists of a database server. These tiers shaping the architecture of client – server developed as separate modules that are generally known as a user interface module, functional process logic module and data storage module.

The accreditation application has the entity with a lot of data. Therefore in developing this application should be used architectural patterns that facilitate both in terms of development and maintenance. The Model View Controller (MVC) development pattern becomes the solution in the development of this accreditation application. MVC architecture separates three basic parts of the application : Model (data model), View (user interface) and Controller (control application logic). These three components are largely autonomous and therefore changing one of them does not basically influence the others.

2. LITERATURE REVIEW
2.1 MVC Concept
MVC (Model-View-Controller) is a design framework which decreases the coupling between the objects by separating business object (Model), user interface (View) and business logic (Controller)[3]. MVC is not specific to any particular language and can be implemented in different languages. It is easy to implement MVC in the language which supports OBJECT.

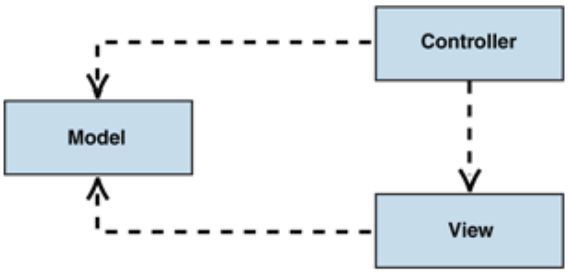

Fig 1. MVC Concept

1. Model; in its simplest form model stores data which is to be accessed by the view and written to by the controller. The model is the most complex of all the parts of the system and will contain all the logic which specific to the application and where domain entities that relate to real concepts [4].

   Model is the part of the application which takes data (from any source) and processes it. The model also handles all data access and storage. It will handle any operations regarding users, saving/loading records, validating registrations. The model is not (common mistakes made by those misinterpreting the pattern) :

   - A simple data access point
   - A class called “model” represents a single database table

2. View; the view contains all the display logic. In PHP it will be the part of the application which generates the HTML. It has direct access to the Model and can query the model to get its data. The view can create callbacks to its controller (for example a clicking button in the view would trigger an action in the controller). A lot of MVC examples state that the view is decoupled from everything else fed data by the controller. This is entirely inaccurate. In MVC the view queries the model to request its own data. The view is not (common mistakes made by those misinterpreting the pattern) :

- Absent of logic
- Given data by the controller

3. Controller; the controller takes user input and updates the model where required. Where there is no user interaction (e.g. where a static data set is displayed and will be the same every time), no controller should be necessary. It is important to note that the controller is not a mediator or a gateway between the view and the model. The view gets its own data from its model. The controller accesses the model but does not contain any display logic itself. All the controller does it respond to user input [4].

   It is important to note that the controller is not in charge of instantiating the model or the view. The controller knows of them but flexibility is heavily reduced in implementations that force the controller to select which view and model is being used. Each controller is linked to a single instance of a view and a single instance of a model. The controller is not (common mistakes made by those misinterpreting the pattern) :

   - A gateway / mediator between the model and the view
   - The place where views get initialized based on the state of a model. The controller is linked to a single view class (although it could be assigned to multiple instances) and responds to actions on it.

2.2 Program Flow

The typical program flow in MVC is [5] :

- The model, view and controller are initialized
- The view is displayed to the user, reading data from the model
- The user interacts with the view (e.g. presses a button) which calls a specified controller action
- The controller updates the model in some way
- The view is refreshed (fetching the updated data from the model).

2.3 PHP Data Object

PDO extension has become one of the trends in developing dynamic web applications and connect to the database . PDO is the PHP5 extensions written in C / C ++ and has several advantages including the system supports a number of databases supported by PHP , faster because it is written with a compiled language , and easy installation . In short PDO needed when we needed a portable application that supports a number of database systems and faster execution[6].

PDO provides database abstraction layer that can use the same functions to execute SQL commands on any database [6] . The main reason to use PDO is security and flexibility when connected to the system database. Through the use of prepared statements (utility not in php_mysql *) then the PDO can prevent SQL Injection [7] .

3. METHODOLOGY

This study uses research and development method that includes four phases: analysis, design, implementation and testing. Implementation PDO parameterized query to prevent SQL Injection in this paper is applied to the scheduling application for vocational high school in North Sulawesi Province of Indonesia. This object -oriented applications in development using Rational Unified Process (RUP) . This method is used because the time needed in application development is relatively short and this application will undergo repairs during the development process

Rational Unified Process (RUP) is a software development approach that is done iteratively, focusing on architecture (architecture- centric) and is directed by use cases. RUP is a software engineering process of good defining and structuring . RUP provides a good structure for defining workflow software project life [7] .

RUP has four stages or phases that can be done iteratively. In this methodology, there are four stages of software development, eg:

1. Inception is a stage model the business processes required and defines the need for the system to be created
2. Elaboration is more focused on planning the system architecture. This stage can also be made to determine whether the desired system architecture can be made or not. This stage also gives emphasis on the analysis of the system design and system implementation and expected results of this phase is to fulfill the Lifecycle Architecture Milestone
3. Construction, this stage is more focused on the development of a component or system features
4. Transition, this stage is the deployment or installation of the system in order to be understood by the user. Activities at this stage includes user training, maintenance and testing of the system to meet user expectations

4. RESULT AND DISCUSSION

4.1 System's Model

The system is modeled using the Unified Modeling Language (UML) as follows:

1. Use Case Diagram; based on the use case ilustrated can be seen the users and responsibity of the new system.

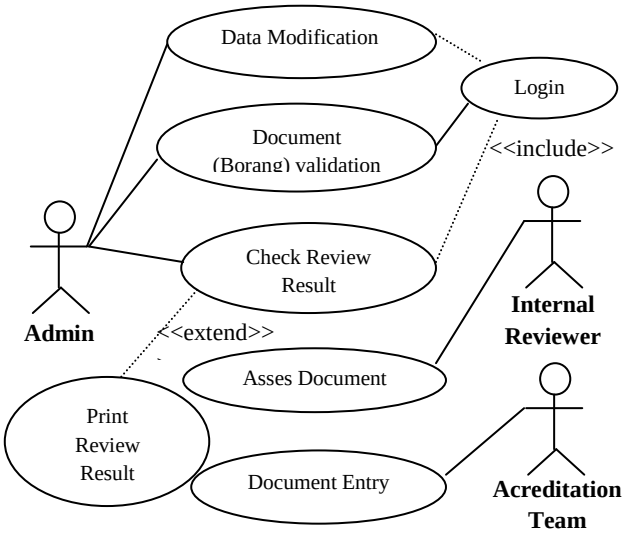

Fig 2. Use Case Diagram

2. Activity Diagram; serves to describe the activities that occur in the application to be developed

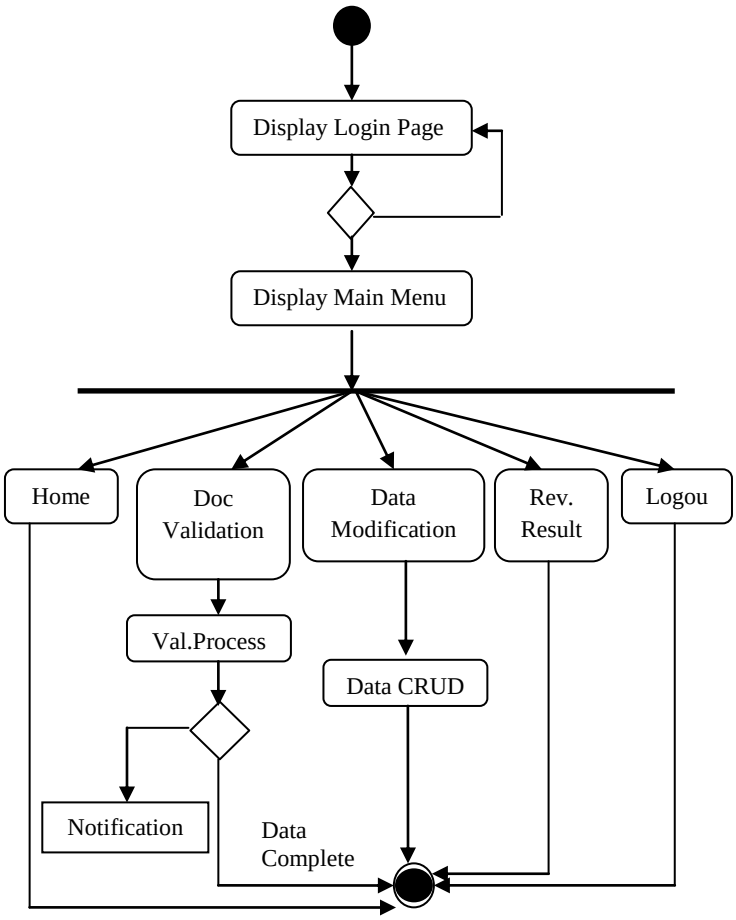

Fig 3. Activity Diagram

3. Class Diagram; used to describe the entities relationship.

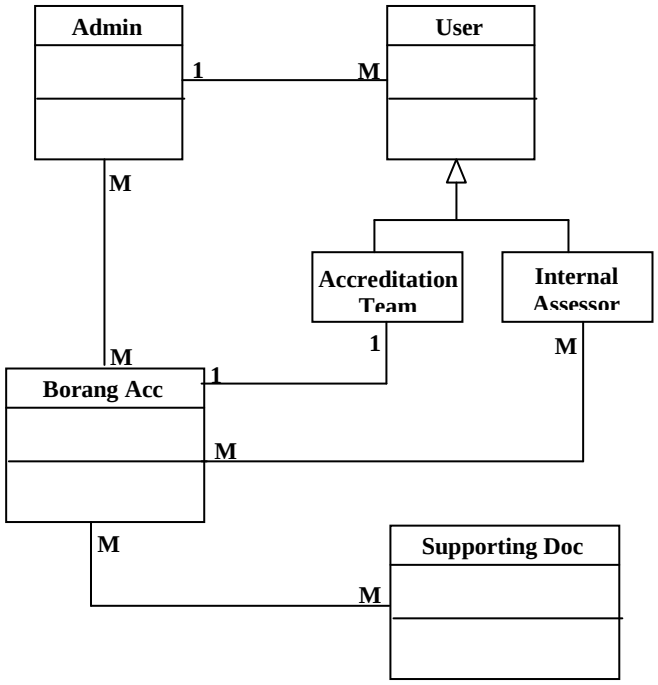

Fig 4. Class Diagram

4.2 Accreditation App MVC Structure

The core concept of MVC is to separate business logic from displaying (the View part). Based on this concept and by simplifying the structure of PHP framework such as Zend, YII, CodeIgniter then the structure of the accreditation application is as in the following figure :

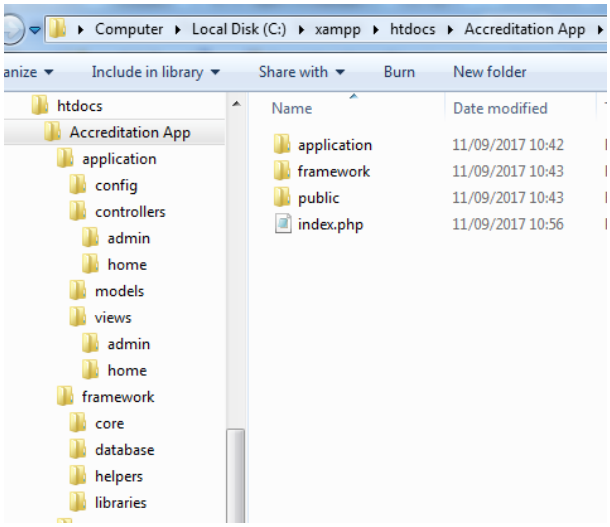

Fig 5. App MVC Structure

There are three main directories of this application is the application directory which is the web app's directory, framework director and public directory to store all public static resources like html, css and js files. This structure is created following the general structure that exists in the php framework. The application directory itself has subfolders ie config (stores the app's configuration files), controllers (this is for all app's Controller classes), models (this is for all app's model classes) and views (this is for all app's View classes). For controllers and views folders a subfolder is created for frontend and backend platforms. In this application frontend and backend are two different systems, but they are CRUD the same database. That is why the Models directory does not have a sub folder frontend and backend.

Framework directory has some subfolders eg core (to store framework's core classes), database (database related classes, such as database driver classes), helpers (help/assistand functions) and libraries (for class libraries). While public directory has some subfolders like css (for css files), images (for images files), js (for javascript files) and uploads (for uploaded filesm such as uploaded images).

The implementation of this application simplifies the common MVC implementation pattern of PHP frameworks such as Zend, YII, CodeIgniter, and so on. The common functions found in the MVC PHP Framework are simplified and developed as in the following listing code :

```
    // Autoloading
      private static function autoload()
      {
          spl_autoload_register(array(__CLASS__,'load'));
      }
      // Define a custom load method
      private static function load($classname)
      {
           // Here simply autoload app's controller and model classes
          if (substr($classname, -10) == "Controller")
          {
              // Controller
              require_once CURR_CONTROLLER_PATH . "$classname.class.php";
          }
          elseif (substr($classname, -5) == "Model")
          {
              // Model
              require_once  MODEL_PATH . "$classname.class.php";
          }
      }

    // Routing and dispatching
      private static function dispatch()
      {
           // Instantiate the controller class and call its action method
            $controller_name = CONTROLLER . "Controller";
            $action_name = ACTION . "Action";
            $controller = new $controller_name;
            $controller->$action_name();
      }
```

The PDO approach is used for connectivity and CRUD manipulation with MySQL as a SQL database.

```
<?php
    class CekLogin
    {
        var $dbh;
        function __construct()
        {
            $hostname = 'localhost';
            $username = 'root';
            $password = '';
            try
            {
                $dbh = new PDO("mysql:host=$hostname;dbname=drpm", $username, $password);
                $this->dbh = $dbh;
            }
            catch(PDOException $e)
            {
                echo $e->getMessage();
            }
        }

        function getDatabase()
        {
            return $this->dbh;
        }


function tambahBerkas($nama,$kategori,$thn,$sem,$tgl,$point,$nip,$keterangan)
{
    $log = new CekLogin();
    $lokasi_file    = $_FILES['fupload']['tmp_name'];
    $tipe_file      = $_FILES['fupload']['type'];
    $nama_file    = $_FILES['fupload']['name'];
    $acak           = rand(1,99);
    $nama_file_unik = $acak.$nama_file;
    if (!empty($lokasi_file))
    {
        move_uploaded_file($lokasi_file,"arsipBerkas/$nama_file_unik");
        try
        {
            $sth = $log->getDatabase()->prepare ("INSERT INTO berkas (nama,kategori,tahun,
                point,arsip,nip,keterangan) VALUES (?,?,?,?,?,?,?,?,?)");
        # use bindValue() to bind data values $nama,$kategori,$thn,$sem,$tgl,$point,$nip)
            $sth->bindValue (1,$nama);
            $sth->bindValue (2,$kategori);
            $sth->bindValue (3,$thn);
            $sth->bindValue (4,$sem);
            $sth->bindValue (5,$tgl);
            $sth->bindValue (6,$point);
            $sth->bindValue (7,$nama_file_unik);
            $sth->bindValue (8,$nip);
            $sth->bindValue (9,$keterangan);
            $sth->execute ();
        //echo "Data berhasil ditambahkan";
        }
        catch(PDOException $e)
        {
            echo $e->getMessage();
        }
    }
   else
    {
        //echo "Data gagal diinsert";
    }
}
```

4.3 Application Testing

An effective testing strategy includes automated, manual and exploratory tests to efficiently reduce risk and tighten release cycles. Test come in several flavors :

- Unit tests validate the smallest components of the system, ensuring they handle known input and output correctly. Unit test individual classes in an application to verify they work under expected, boundary and negative cases.
- Integration tests exercise an entire subsystem and ensure that a set of components play nicely together.
- Functional tests verify end-to-end scenarios that your users will engage in.

This accreditation app use functional test to verify all functions and test cases that are used to verify particular features are as follows :

Table 1. Test Cases

Test Scenario	Test Steps	Expected result	Actual Result
Check user login with valid data	• Go to site • Enter UserId • Enter Password • Click Submit	User should Login into application	As Expected
Check user login with invalid data		User should not Login into application	As Expected
Insert, update and delete database with valid data	• Go to site • Enter data, select key (for update and delete) • Click Submit	Data for each entity can be inserted, updated and deleted	As Expected
Insert and update with invalid data	• Go to site • Enter data for inserting and updating • Click Submit	Data for an entity can not be inserted or updated.	As Expected
Input data and upload accreditation documents	• Go to site and select accreditation document menu • Enter data including upload the document • Click Submit	Data can be stored in database and accreditation document can be moved into temp folder.	As Expected
Input invalid data and upload accreditation documents		Data can not be stored in database and accreditation document can not be moved into temp folder.	As Expected
Validate accreditation documents	• Go to site and select validate document menu • Select document based on its category. • Click Submit	Accreditation document can be moved to permanent folder if it is fulfill the standart and vice versa.	As Expected
Input data as result of accreditation document assessment	• Go to site and select assessment menu. • Enter document result assesment's data • Click Submit	Data can be stored and assessment report can be displayed	As Expected
Input invalid data as result of accreditation document assessment		Data can not be stored and assessment report can not be displayed	

5. CONCLUSION

Model View Controller (MVC) architectural pattern is very suitable to be used as application development architecture for application that has many entities like in accreditation application development. MVC separates the business logic from the presentation view and controls the process mechanism through the component controller. This mechanism provides convenience to developers to develop and maintain the system, even the ease of combining with other systems. Future scope for this paper is how to combine this MVC architecture with big data and cloud computing technology for data store, fast processing and security.

6. REFERENCES

[1] Yash Tiwari, Mallika Tiwari, "A study of SQL of injection techniques and their prevention methods", International Journal of Computer Applications (0975-8887), vol 114, no. 17, March 2015.

[2] Bojken Shehu, Aleksander Xhuvani, "A literature review and comparative analyses on SQL injection : vulnerabilities, attcks and their preventation and detection techniques", IJCSI International Journal of Computer Science Issues, vol 11, issue 4, no. 1, July 2014.

[3] David J.Anderson, 2011, Using MVC Pattern in Web Interactions, http://www.uidesign.net/Articles/Papers/UsingMVCPatterninWebInter.html

[4] P.Simek, J.Vanek, "New approaches to presenting information in agrarian sector and country areas – Technological solution of agris web portal", Agris on-line papers in economics and informatics, vol.II, no.4, 2012

[5] Nutaro, A and Hammonds,P, "Combining the Model / View / Control Design Pattern with the DEVS Formalism to Rigor and Reusability in Distributed Simulation", JDMS, Vol. 1, Issue 1, April 2010"

[6] Sendiang, Maksy, Anritsu Polii, and Jusuf Mappadang. "Minimization of SQL injection in scheduling application development." Knowledge Creation and Intelligent Computing (KCIC), International Conference on. IEEE, 2016..

[7] Utami E, Raharjo S,"Database Security Model in the Academic Information System", International Journal of Security and Its Applications. 8:170. 2014

[8] Chen Q, "Compare and study about owing to the three kinds important softwaresdevelop process", Proceeding of the International Conference on Education Technology and Economic Management (ICETEM). 450-451. 2015