Project Description:
In this project, you will create a simple web application for storing and retrieving data about movies. The project has the following goals:

1. Get experience with using Amazon EC2 to host your web application in the cloud.
2. Use one of the most popular NoSQL databases (MongoDB) for schema-free storage of data.
3. Get experience with developing dynamic web sites using PHP.

You can either do the project alone, in which case you will be required to implement Part 1 only, or in teams of two, which will require implementing both parts of the project.

Preliminaries:

Amazon EC2:

Your application in this project will be residing in a virtual machine on Amazon's Elastic Compute Cloud (EC2). The virtual machine you will be provided with will have the necessary components for your application (including MongoDB and the web server), therefore the only thing you will need to do is to write the server side code in PHP and put your PHP files in the correct directory (/var/www/html/) and you will be able to access your application by pointing your web browser to the global IP address of your machine. You will be able to connect to your machine instance with an ssh client (you can just use the ssh command in Unix), using the following syntax:

```
ssh -i keyfile.pem ubuntu@XX.XXX.XXX.XX
```

where keyfile.pem is the name of your private key file (which we will send to you) and XX.XXX.XXX.XX is the address of your machine instance. The key file should be in the directory in which you execute the command. You can use the scp command to copy files to your machine instance, although vim is also available if you prefer to edit your files on the machine instance. We will create machine instances and send you the address and key file once you decide whether you will work alone or in a team.
**MongoDB & PHP:**

MongoDB is a schema-less database, which stores data as document collections in BSON format. You do not need to know details of all features of MongoDB for this project. The tutorial at [http://devzone.zend.com/1730/getting-started-with-mongodb-and-php/](http://devzone.zend.com/1730/getting-started-with-mongodb-and-php/) should provide sufficient information to learn how to run MongoDB, insert records into the database, retrieve records from the database and connect your PHP website with MongoDB. Note that MongoDB will be setup under the directory *mongodb* on your EC2 machine instance and the data will be stored in the directory *mongodb/data*. The database will come pre-populated with some records for movies, actors, cast members and users.

Note that, to be able to access the data in your database from within your application, the database server should be running (see the relevant section of the tutorial).

**Requirements:**

Note that the user interface for this project does not have very strict requirements in terms of look and feel. However, you should include all elements required to achieve the desired functionality for each web page of the application as explained below. Whenever a requirement states the inclusion of a hyperlink, you can also use a button instead of the hyperlink if that will make the implementation easier for you.

**Part 1:**

In this part of the project, you will create the user side of the movie database, which will allow regular users to register on the website, and view information about movies and actors.

A page-by-page description of the website is provided below. Note that in this project, you don't need to do any error checking for data types in text fields or null values in text fields. You can assume that all input will be in the correct format.

All pages of the website should provide hyperlinks for the following items:

- **Movies**: Clicking on this link should take the user to the Movies page
- **Actors**: Clicking on this link should take the user to the Actors page
- **Login/Register (or Logout)**: Clicking on this link should take the user to the Login/Registration page if the user is not already logged in and should log the user out if he is already logged in.

1. **Welcome page**: This page should have the static text "Welcome to the CS34800 movie database!". The users should be able to navigate to other pages by clicking on the appropriate links on this page.
2. **Login/Registration page:** This page will be reached by clicking on the Login/Register link. The page should provide a text field for username, a text field for password and a login button at the top part of the page, and a text field for username, a text field for password, a text field for first name, a text field for last name, and a register button at the bottom part of the page. If the user clicks on the login button, you should check whether the username and password provided in the login text fields have a match in the database. If not, you should take the user to an error page, which has the text “Invalid login” at the top, in addition to the same fields as on the login/registration page at the bottom. If the login is successful, you should take the user to the welcome page. Note that as long as the user is logged in to the website from this point on, all pages you display to the user should have the text “Hi first_name last_name!” at the top of the page and the “logout” link next to that text (first_name last_name should be the first name and last name for that user found in the database). If the user clicks on the logout link at any point, you should not display the “Hi first_name last_name!” text until the user logs in again (Hint: you will need to use the session variables in PHP to achieve this).

If the user clicks on the register button, you should insert a new user record into the database, with the given username, password, first name and last name. You do not need to check whether this user already exists. You should still take the user to the welcome page and display the text “Hi first_name last_name!” at the top of the page as in the case for login.

The user data will be stored in the following BSON format:

```json
{
    "username": "pangin",
    "password": "pass",
    "firstName": "Pelin",
    "lastName": "Angin"
}
```

3. **Movies page:** This page should contain a text field for entering the movie title and a submit button. Once the user clicks the submit button, you should take the user to the information page for the movie whose title you will get from the text field.

4. **Actors page:** This page should contain a text field for entering the actor/actress name and a submit button. Once the user clicks the submit button, you should take the user to the information page for the actor whose name you will get from the text field.

5. **Movie information page:** You should display the following information on this page:
   a. Movie title
   b. Genre
   c. Release year
   d. Plot summary
e. Cast members (the list of all actors/actresses in this movie)
f. Average rating for the movie (note that your program should calculate the average rating)

Movie information will be stored in the database in the following format:
{
  "title": "American Hustle",
  "genre": "Comedy",
  "releaseYear": 2013,
  "plotSummary": "A con man, Irving Rosenfeld, along with his seductive partner Sydney Prosser, is forced to work for a wild FBI agent, Richie DiMaso, who pushes them into a world of Jersey powerbrokers and mafia."
}

Movie rating information will be stored in the following format:
{
  "movieTitle": "American Hustle",
  "user": "pangin",
  "rating": 8
}

Cast member information will be stored in the following format:
{
  "movie": "American Hustle",
  "castMember": "Amy Adams"
}

6. Actor information page: You should display the following information on this page:
   a. Actor name
   b. Birth date
   c. List of movies (movie titles) in which this actor is a cast member
   d. Pictures of the actor/actress (image files will be available in the /var/www/html/images directory, the number of pictures for every actor will be 5 and the file names will be stored in the following format:

   ```
   {
     "actormap": "Brad Pitt",
     "fileNumber": 2,
     "fileName": "BradPitt2.jpeg"
   }
   ```

   Actor information will be stored in the following format:
   {
     "actorName": "Brad Pitt",
     "birthdate": "December 18, 1963"
   }
Part2:

In this part of the project, you will create the administrator side of the movie database, which distinguishes between regular users and the administrator and provides all functionality from the first part to regular users, while allowing the administrator to perform additional functions including adding movie, actor and cast member information to the database. You will also add functionality to allow users to rate movies.

In addition to the functionality in part 1, you should provide the following functionality:

1. **Changes to the links on all pages:** If the administrator logs in to the website (by providing the username “admin” and password “pass”), in addition to the hyperlinks in the first part, you should display the links for “Add a movie”, “Add an actor” and “Add a cast member” as long as the admin is logged in (these links should not be displayed if a regular user is logged in or if nobody is logged in).

2. **Changes to the movie information page:** This page should now also include a button “Rate this movie” and a group of radio buttons representing numbers from 1 to 5. If the user clicks on the “Rate this movie” button, you should check that the user is logged in and insert the rating (the value of the selected radio button) to the database for this movie by the current user. If the user is not logged in, you should show the same page content (the movie information) and the error message “Please login to rate this movie”.

3. **Movie insertion page:** This page will be reached upon clicking on the “Add a movie” link. It should contain text fields for movie title, release year and plot summary and a drop-down menu for the genre, which provides the following list (genre options):
   
   - Comedy
   - Adventure
   - Drama
   - Biography
   - Science fiction

   It should also contain an “Add movie” button. When the button is clicked, a movie with the provided information should be inserted into the database (**Hint:** You can see an example of how to create drop-down menus at http://www.html-form-guide.com/php-form/php-form-select.html)

4. **Actor insertion page:** This page will be reached upon clicking on the “Add an actor” link. It should contain text fields for actor name, birthdate and five text fields for picture file names. It should also contain an “Add actor” button. When
the button is clicked, an actor with the provided information should be inserted into the database.

5. **Cast member insertion page:** This page will be reached upon clicking on the “Add a cast member” link. It should contain a drop-down menu for actor names and a drop-down menu for all movies (You should populate the drop-down menus with all actor names and movie titles in the database). It should also contain an “Add cast member” button. When the button is clicked, the selected actor-movie pair should be inserted into the database.

**Submission Instructions:**

Your main PHP file should be named MovieDB.php.

Please create a README file containing identifying information. For example:

CS348 - Project 4  
Author(s): First Member & Second Member  
Login(s): firstlogin, seconlogin  
Email(s): firstlogin@purdue.edu, seconlogin@purdue.edu  
Application site: http://xx.xx.xx.xx/MovieDB.php

Include here anything you might want us to know when grading your project.

To turn in your project, ssh to one of the lab machines, create a folder named project4 in your home directory and copy your PHP files and your README file to that folder.

After copying your files in the folder project4, execute the following command in your home directory:

```
turnin -c cs348 -p proj4 project4
```

To verify the contents of your submission, execute the following command right after submission:

```
turnin -c cs348 -p proj4 -v
```

Note that only one submission is required per group. You can use either member’s account for the submission.