CSCI 4350
Spring 2016
Final Exam

Last Name: _____

First Name: _____

During the exam, you may reference printouts of any source code you have written (HTML, CSS, Javascript, SQL, PHP, Codeigniter). Write your name on this exam sheet and write your answers on the answer sheets provided.

1. (4 pts) How are cookies and session variables the same? How are they different?
2. (2 pts) Name one benefit of using Ajax.
3. The following questions pertain to web frameworks.
   (a) (2 pts) Name two frameworks.
   (b) (4 pts) Name two benefits of using a framework.
   (c) (4 pts) What does MVC stand for and how is it relevant to frameworks?
4. (4 pts) Currently, some web bots do not execute Javascript. Explain one way in which this fact can produce the function of a CAPTCHA without involving extra work on the part of a legitimate user.
5. (4 pts) Name some differences between using the PHP `mail()` command and using HTML's `mailto:` directive to send email from a web application.
6. (4 pts) What advice would you give up-and-coming Web 2 students about using a version control system (hg or git) in a group environment?
7. The following questions assume you are writing a web application called `groups` that provides Twitter-like functionality. In particular, everyone in your "group" can see what others in the group have posted. The project has already been started using Codeigniter. Your task is to implement two pages as follows:

   - A post page that allows a user to submit a post.
   - A viewing page that allows a user to view posts.

   The Database
   Assume the PostgreSQL database is in place and the Codeigniter `database.php` file is properly set. The database has this structure:

```
CREATE TABLE users (
  user_id   SERIAL,
  email     VARCHAR(120) NOT NULL,
  password  VARCHAR(200) NOT NULL,
  lastcheck TIMESTAMP WITH TIME ZONE DEFAULT current_timestamp, --plaintext
    -- NOTE: stores when we last checked for posts (defaults to NOW)
  PRIMARY KEY (user_id)
);
```

```
CREATE TABLE posts (
  user_id INTEGER,
  message VARCHAR(100) NOT NULL,
  ts      TIMESTAMP WITH TIME ZONE DEFAULT current_timestamp
);
```

Existing Model Code

You may assume the following model code is already in place in a model called groupsdb:

```
// get posts from the last hour
// we also update "lastcheck"
// $this->session->userdata('user_id') gives us logged_in user's id number

function get_recent_posts()
{
  $this->db->query("UPDATE groups.users SET lastcheck=DEFAULT WHERE user_id=?",
      $this->session->userdata('user_id'));
  return $this->db->query("SELECT ts,email,message FROM groups.posts JOIN ".
                          "groups.users USING(user_id) WHERE ".
                          "ts>=(current_timestamp-interval '1 hours') ".
                          "ORDER BY ts DESC")->result();
}


// get posts since the current user's "lastcheck" time stamp
// $this->session->userdata('user_id') gives us logged_in user's id number

function get_posts_since_lastcheck()
{
  $lastcheck= $this->db->query("SELECT lastcheck FROM groups.users WHERE user_id=?",
      $this->session->userdata('user_id'))->row()->lastcheck;
  $this->db->query("UPDATE groups.users SET lastcheck=DEFAULT WHERE user_id=?",
      $this->session->userdata('user_id'));
  return $this->db->query("SELECT ts,email,message FROM groups.posts JOIN ".
                           "groups.users USING(user_id) WHERE ts>? ORDER BY ts ASC",
                           $lastcheck)->result();
}
```

Some Queries You May Find Helpful

- `SELECT password FROM groups.users WHERE user_id=? --` obtain the password of the specified user
- `INSERT INTO groups.posts (user_id,message) VALUES (?,?) --` create a new post for the specified user

Organizing Your Work
Since this question is in the context of Codeigniter, you should organize your answers by having a one answer sheet for the controller code, another sheet for the model code, and another sheet for the view code. Assume the controller is in `welcome.php` and the model is in `groupsdb.php`.

NOTE: Don't spend a lot of time trying to make the pages look pretty . . . just make them do what they are supposed to do.

NOTE: You don't need to show the constructors, or headers for models or constructors. Just write the functions that would go inside of them.

Creating the "Make a Post" Page (30 pts)
Write the appropriate commands separated among model, view, and controller to will provide a form that a logged in user can use to submit a post. We will require that a user enter their password every time they want to write a post. (While this "feature" helps prevent others from submitting posts while a user's back is turned, it might be the reason you have heard of "Twitter", but not of "Groups"!) The form, therefore, will have a text box for the post, a password box for the password, and a submit button.

The form should be validated to ensure that the typed password matches the user's stored password (which is not encrypted). It should also require that the post not be blank. Furthermore, the message should be sanitized to prevent cross-site scripting attacks. All validation and sanitization should be server-side.

Once the form passes validation, it should be saved in the database and the user should be redirected to the "View Posts" page described below. If validation fails the user should be returned to the "Make a Post" page and be presented with appropriate error messages.

NOTE: Assume the code `$this->session->userdata('user_id')` will give you the logged-in user's database id number.

Creating the "View Posts" Page (20 pts)
Write the appropriate commands separated among model, view, and controller to implement a page that when visited will display all posts with most recent listed first (see `get_recent_posts()` in model code above). The elements should be stored in HTML using a `<ul>`. For each post display the time stamp, the email address of the person who made the post, and, of course, the message.

After the posts are displayed, have a Javascript function that one time per minute issues an Ajax request to retrieve any posts that have been added to the database since the stored `lastcheck` time in the database. Of course, you'll update the value

for `lastcheck` after each call. If one or more new posts are found in the database, add the new entries at the top of the `<ul>`. See

You may assume that the JQuery library is loaded on the page. Also, you may use the model function `get_posts_since_lastcheck()` as defined above.

NOTE: The following Javascript code will cause an alert box to be display every minute:

```
setInterval(function() {
        alert("Hello");
}, 60*1000);
```