CS 142 Final Examination

Spring Quarter 2018

You have 3 hours (180 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During the examination you may consult two double-sided pages of notes; all other sources of information, including laptops, cell phones, etc. are prohibited.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination.

_____

(Signature)

_____

(Print your name, legibly!)

_____@stanford.edu

(SUID - stanford email account for grading database key)

Problem	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11
Score											
Max	10	10	10	8	8	8	8	10	8	8	8
Problem	#12	#13	#14	#15	#16	#17	#18	#19	#20	Total	
Score											
Max	8	8	10	12	10	8	8	10	10	180	

Problem #1 (10 points)

The CS142 Photo Sharing application you built used Express.js session management software `express-session` configured to use the default server-side session storage, `MemoryStore` which stores the session state in Node.js memory. Consider an alternative session store called `unsafe-cookie-session` that works by storing the session state object as a cookie with the object encoded as a JSON string. This cookie is attached to the session much in the same way that `express-session` attaches its cookie.

A. Describe the key advantage of `unsafe-cookie-session` compared to `express-session`.
B. Assume a threat model that includes an untrustworthy user of our application. Describe the damaging attacks the user could do with `unsafe-cookie-session` where the session state is stored in a cookie. Describe what kind of changes we could make to produce a `safe-cookie-session` that defeats the attacks yet still gets the benefit listed in A.

Problem #2 (10 points)

The CS142 Photo Sharing application session management (`express-session` described in problem #1) had an unfortunate behavior in that a browser refresh resulted in the user being logged out.

A. Describe what happened to the web application that resulted in the user no longer being logged in.
B. Sketch out changes you would need to make in order for your Photo Sharing web application to handle a browser refresh without the user having to log in again.

Problem #3 (10 points)

In order for our CS142 Photo Sharing application to run using its MVC pattern we need to have model data from the Node.js backend shipped to our browser-based frontend. In general there are two ways this shipping of model data can occur. The code running in the browser can "pull" the model data from the server or the code running in the server can "push" model data to the browser.

A. Which of the two ways ("push" or "pull") would you say our Photo Sharing application used? Briefly explain your answer.
B. If we could magically have the other way that is not the supplied answer in Part A available for our application, what might that other way be useful for?

Problem #4 (8 points)

Our discussion of full stack web applications involved mention of two types of data: *model data* and *session state*, each of which we end up treating very differently. For example, we used different storage systems for them. Is there ever a scenario when some application data could be both *model data* and *session state*? If so, give a plausible example. If no, briefly explain why not.

Problem #5 (8 points)

Your web application has become popular over in Europe which has just passed the Model Data Protection Act (MDPA). Under the MDPA all model data of web applications has to be encrypted when transferred over the Internet. Sketch out a change we could make to our Photo Sharing application to conform with the MDPA.

Problem #6 (8 points)

When your web browser connects to a web server, one of its first steps is to communicate with a DNS resolver, which tells your browser the IP address associated with a host name in the URL (for instance, the IP address of the machine you should connect to to access www.google.com might be 216.58.194.164). Your browser then attempts to connect to the server with that IP address. Unfortunately, DNS lookups are not particularly secure, so an attacker might be able to trick your browser into connecting to an evil server instead of a real Google server.

As a savvy student of CS 142, you're aware of the dangers of the Internet and are careful to always connect to your bank's website (www.mybank.com) over HTTPS. Unfortunately, an attacker has managed to gain control of your local DNS resolver and directs you to a server under his control (10.0.0.2) instead of the real bank server (10.0.0.1). Will HTTPS protect you from this attack? Please take two or three sentences to justify your answer.

Problem #7 (8 points)

If you receive an email and click on a link for `https://www.bankofthevvest.com` (note: vvest, not west.) Assume that www.bankofthevvest.com is under the attacker's complete control. Will your browser provide indication that the site you are visiting is not legitimate? If not, explain why. If so, how would it likely show up?

Problem #8 (10 points)

In class we talked about a ORM (Object Relational Mapping) layer that allowed objects of web application data to be mapped into a relational data model. You could also imagine a ROM (Relational Object Mapping) layer that maps a relational data model into a object/document database like MongoDB. Using your knowledge of relational and object data models, describe how an ROM might work by sketching the mapping of the relational concepts of **tables** with **rows** and **columns**, and **primary** and **secondary indexes**. A mapping layer would take these relational model concepts and implement them using features of the object model. Describe this mapping for the listed relational concepts.

Problem #9 (8 points)

In Project #5 we introduced the `FetchModel` function in your controllers that fetched data from the server. We had you implement the fetching using `XMLHttpRequest`. In order to prevent unwanted behavior in Angular, we suggested that you use `$scope.$apply`. Had the implementation used a AngularJS service `$resource` and `$http` the suggestion to use `$scope.$apply` wouldn't have been needed. Explain the problem that necessitated `$scope.$apply` and why it wasn't needed with the AngularJS model fetching services.

Problem #10 (8 points)

The following is an Express.js handling code for a particular URL with an "id" parameter.

A. What will go wrong in the following code, and why?
B. How would you fix it?

```
var user_photos = [];
Photo.find({user_id: request.params.id}, function (err, photos) {
    if (err) {
      response.status(400).send(JSON.stringify(err));
      return;
    }
    // process photos...
    user_photos = photos;
});

response.status(200).send(JSON.stringify(user_photos));
```

Problem #11 (8 points)

In the context of the photo sharing web app that you worked on, give an example of user input validation that we need only do on the frontend but not on the backend and an example of input validation that we could get away with doing only on the backend.

Problem #12 (8 points)

A software engineering course at Stanford had students build a web server with a focus on a clean modular decomposition where each module was free of code with knowledge belonging to another module. One group proposed the following processing pipeline for HTTP requests:

1. Read request - Read the full HTTP request from the data coming into the TCP socket
2. Parse HTTP request - Extract out the header properties and body of the HTTP request
3. Dispatch request - Call the handler function based on the URL and HTTP method

Using your knowledge of HTTP request headers, describe why this decomposition lost points for not being clean with the same work needing to be done in multiple steps.

Problem #13 (8 points)

In the class projects we used a simple Node.js web server program (webServer.js) running in the local environment to allow the browser to fetch the project files from the local file system. Browsers are perfectly capable of fetching files from the local file system using URLs specifying the "file:" protocol. Explain the reason we couldn't just use the "file:" protocol to fetch the various pieces of our web application given that everything fetched was coming from the local machine.

Problem #14 (10 points)

A. Describe the basic approach of a Cross-Site Request Forgery (CSRF) attack.
B. The request forgery part of the CSRF has limitations on the possible requests that can be forged. In particular only a relatively small set of the request types can be used in the attack. Describe this limitation and the reason behind it.

Problem #15 (12 points)

Consider the following Express.js program:

```
var express = require('express');
var app = express();

app.use(function(request, response, next) {
  request.value = 'foo';
  next();
});

app.get('/test', function (request, response) {
  response.status(200).send(request.value);
});

app.use(function(request, response, next) {
  request.value = 'bar';
  next();
});

app.get('/test2/:test3', function (request, response) {
  var paramValue = request.params.test3;
  var queryValue = request.query.test3;
  response.status(200).send((paramValue === '4' && queryValue ===
'5')
      ? 'baz' : 'qux');
});

app.use(function (request, response, next) {
  response.status(404).send('N/A');
});

app.listen(3000, function () {});
```

Question continued on next page …

…. continued from previous page.

Answer the following questions below. Hint: When processing requests, ExpressJS executes `app.use` and matching `app.get` callbacks in the order in which the `app.*` statements are executed.

A. Write down the response that the web server sends back for a `GET /test` request.

B. Write down the response that the web server sends back for a `GET /test3` request.

C. Write down the type of request (specify verb + url) that should be made to get a 'qux' response. Please include any necessary url path components or query strings. (e.g., sample (incorrect) answer: GET /test?q=hi)

D. Write down the type of request (specify verb + url) that should be made to get a 'baz' response. Please include any necessary url path components or query strings. (see sample answer above)

Problem #16 (10 points)

The Node.js runtime includes a function `setImmediate` that allows the call to directly add a callback to the Node.js event loop. It is like the DOM's `setTimeout` function except there is no delay.

Consider the following Node.js program:

```
for (var i = 0; i < 2; i++) {
  setImmediate(function(err) {
      console.log(i);
  });
  console.log(4);
}
```

Write down what is printed after the for block code above is executed (Order matters).

Suppose the `setImmediate` call is wrapped in an immediately invoked function expression like:

```
for (var i = 0; i < 2; i++) {
  (function (i) {
    setImmediate(function(err) {
      console.log(i);
    });
  })(i);
  console.log(4);
}
```

Does this affect the order/contents of what is printed to the console? If yes, state so and write down what is now printed after the for block above is executed (Order matters). If no, state so and briefly justify your answer.

Problem #17 (8 points)

REST and GraphQL are two different protocols used to fetch model data for web applications. Assume you have a web application with users located in countries where connections to the web app's backend servers use **low bandwidth networks** with **long round trip times**. Is either REST or GraphQL advantageous over the other under these communication characteristics? Justify your answer.

Problem #18 (8 points)

Which of the components in a MVC pattern would be inappropriate to put on a Content Distribution Network? Justify your answer.

Problem #19 (10 points)

While surfing the web while procrastinating on working on your CS142 project, you run across a news article that claims a web application enabled a *Cross Site Scripting Attack* that in turn launched a *SQL Injection Attack*.

A. Does a Cross Site Scripting Attack launching a SQL Injection Attack make any sense? Justify your answer.
B. Would reversing the terms with a SQL Injection Attack launching a Cross Site Scripting Attack make sense? Justify your answer.

Problem #20 (10 points)

The reminder emails from the Stanford Axess system used to contain useful hyperlinks for taking the reader to the right place in the Axess web application. At some point in the recent past Stanford IT removed the hyperlinks from the email and included the note:

> Note: Hyperlinks are purposefully excluded from this email notification as a deterrent to _____. Referenced sites in bold may be accessed by adding '.stanford.edu' to the site name or by searching from the Stanford home page.

State what the missing word in the above note is and how this change is intended to help as an deterrent.