CS 142 Final Examination

Fall Quarter 2017

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	8	8	8	8	8	8	8	9	8	6	8
Problem	#12	#13	#14	#15	#16	#17	#18	#19	#20	#21	Total
Score											
Max	6	10	16	12	8	12	12	8	8	8	187

Problem #1 (8 points)

Mobile devices offer some challenges for web applications beyond the typical desktop computer. Among the challenges are:

A. Smaller screen sizes which can display less information in a view.
B. Much higher latency and lower bandwidth Internet connections.

Describe the advantages that GraphQL has over REST APIs for dealing with these mobile challenges.

Problem #2 (8 points)

(A) Give one advantage that scale-out architecture has over a scale-up architecture and one advantage that scale-up architecture has over a scale-out architecture.

(B) If you wanted to scale your photo app to work for more users, which architecture, scale-up or scale-out, would you choose? Justify your answer.

Problem #3 (8 points)

When doing direct DOM programming in JavaScript we could either assign to a DOM node's `innerHTML` property or explicitly do DOM calls to modify the existing DOM tree or create and insert a new subtree of DOM nodes.

Later in class we saw the storing into the innerHTML property approach used as part of a **Cross Site Scripting Attack** (XSS). Is the explicit DOM modification alternative approach also risky from an XSS attack point of view? Provide justification for your answer.

Problem #4 (8 points)

What did we get from using Mongoose as an intermediary between our Express.js handlers and the MongoDB database that we wouldn't have gotten if we talked to MongoDB directly?

Problem #5 (8 points)

Assume you have joined a hot Silicon Valley web application startup that has Photo Sharing App headed to a billion users and growing. One complaint of the app users is the user registration process is painful since you have a try a bunch of different user names before finding one that hasn't been taken already. The user interface designers have come up with a fix that makes the user registration view update an indication of the availability of the user name as the user name is inputted key-by-key into the view.

(A) Your job is to implement the backend API for the key-by-key checking if an user name is available. Your initial implementation which runs a query of the User collection to see if the name exists gives the right answer but the query consumes significant resources and is growing the the number of users. Describe what you could do to the database to make this query not scale directly with the number of users.

(B) Although you are now a hero for fixing that user check problem, you hear complaints that another API has got much slower. The API allows for bulk loading of new accounts that are known to be unique has slowed down significantly with your change. Describe what is happening here.

Problem #6 (8 points)

Wireshark software allows anyone with a network adapter to capture and examine wireless packets. This means an attacker could use their laptop near a user of your photo sharing app and see all the requests the user sends along with the cookie attached to that request.

(Part A)
This is a problem because the attacker is able to see all of this information. Using cryptography we can encrypt the connection to the web server, but to do so we need both the user's environment and our web server to know a shared secret (i.e. encryption key). Describe in detail how we could setup the shared secret even with the attacker running Wireshark on our network?

(Part B) Since the attacker sees our HTTP request/response traffic they will also see the Express session cookies on the requests. Describe in detail why we need to worry about the attacker using the session cookie to impersonate the current user, but wouldn't need to worry about the attacker being able to modify a session cookie to impersonate a different user.

Problem #7 (8 points)

Imagine you put your Project 8 Photo App into production. Your site's fresh style and innovative user experience starts drawing all the millennials away from current photo sharing applications. They become upset that your site is taking business away from them and causing them to lose on advertising revenue, so they rent a botnet to conduct a DDOS attack on your site.

Provide **two** ways that the botnet could be configured to send requests to attack your site. Be specific about what API is targeted and why that would be an effective DDOS attack. For each way you list describe something you could do to mitigate or guard against the attack.

Problem #8 (9 points)

Angular.js includes multiple services to communicate between the app running the browser and a web server. Two of the services we introduced in class were `$http` and `$resource`. Describe the relationship between the two services.

Is it possible to implement the services on top of each other (i.e. implement `$http` on `$resource` and/or implement `$resource` on `$http`)? Explain your answer.

What guideline of RESTful API design allows you to easily add more web servers to your API to service requests?

Problem #9 (8 points)

Describe the advantages and disadvantages of having your web server returns the view components of your web application using the HTTP response header line of:

```
Cache-Control: max-age=7200
```

You discover a bug in your web server that a particular Angular JavaScript controller has a bug and was mistakenly being served with a near infinitely long `Cache-Control` setting. What change could you make to your application to ensure that fixes to the web app will be seen immediately?

Problem #10 (6 points)

Explain in detail what the browser will do when it finds this line in an HTML document:

```
<script type=”text/javascript” src=”/main.js”></script>
```

Problem #11 (8 points)

One of the CAs once used the Instagram REST API for a research project. The CA used it for the following tasks:

(1) search for very popular photos by keyword
(2) automatically post questions about the photos the API returned

Part A)
Both of these are REST API calls on the Instagram domain. For each of tasks ((1) and (2)) list:

- What type of request was made (GET, POST, PUT, DELETE)?
- What types of systems were likely involved the request (e.g. load balancer, web server, database server, memcache)?

Provide a brief justification for your answer:

(1) search for very popular photos by keyword

(2) automatically post questions about the photos the API returned

Part B)
Which of the two calls ((1) and (2)) would you expect to run faster? Why?

Problem #12 (6 points)

GAE is an example of something that has become known as **serverless computing**. Serverless computing appears to be an oxymoron in that the computing is most definitely done on servers. Explain how a term with such a contradiction could be adopted by the industry?

Problem #13 (10 points)

Google App Engine (GAE) is a service for building scalable web apps. The developer provides the code and the service provides the complicated load balancing, virtualization, etc. to make the app scalable. In spite of strong developer interest, GAE didn't initially support WebSockets and continues to strongly recommend against using them. Why do you think the GAE developers don't like WebSockets?

What are some advantages and disadvantages of cloud services like Google App Engine, relative to housing one's own servers?

Problem #14 (16 points)

The communication protocol between the frontend running the browser and the backend is frequently run over HTTP. That means information can be transmitted using the HTTP request and response by encoding in parts of the URL such as the **hierarchical part** and the **query params** or in the **body** of the HTTP request/response.

Given the purpose of the request made by your browser code in your photo app below, choose the most appropriate properties to populate in the request out of the three possible answers: (**hierarchical part**, **query params**, **body**). Briefly explain your answer.

Changing user password. (input: new password)

Searching all user with first name that starts with string "ken". (input: search string)

Modifying the current user occupation information. (input: new occupation)

Fetching a photo details in a dedicated page for one photo. (input: photo id)

Fetching all photos with date after 05/04/XXXX. (input: date)

Tagging a user face in an existing photo. (input: pixel bounding box)

Fetching all user tagged in a photo. (input: photo id)

Changing current user profile photo. (input: photo image)

Problem #15 (12 points)

(Part 1)
Consider the code fragment below. Suppose that there are a total of 5 users in the database. Given the endpoint below, the request to "/user/countusers" is supposed to return the number of users in the system but has a required line commented out.

```
var User = require('./schema/user.js');
var async = require('async');

app.get("/user/countusers", function(request, response) {
    var count = 0;
    User.find({}, function (err, users) {
        if (err) {
            response.status(400).send(JSON.stringify(err));
            return;
        }
        //(A) response.status(200).send(JSON.stringify(count));
        async.each(users, function(user, callback) {
            count += 1;
           //(B) response.status(200).send(JSON.stringify(count));
           callback();
        }, function(err) {
            //(C) response.status(200).send(JSON.stringify(count));
        });
    })
    //(D) response.status(200).send(JSON.stringify(count));
});
```

There are four commented-out Express response-sending statements labeled (A)-(D). The statements either return the **correct count**, **return an incorrect count** but otherwise run successfully, or will use Express incorrectly and likely **generate an error**. For each of the statements say which of the above it will be if the line is uncommented. If the code isn't using Express incorrectly, state what value for the number of users is returned.

//(A)

Problem continued on next page…..

Problem 15 continued.

//(B)

//(C)

//(D)

(Part 2)
If the code is run as is, all the response lines commented out, describe the symptoms you would see in a web app attempting to use the endpoint.

Problem #16 (8 points)

The following Mongoose.js code returns the user list model from the CS142 Photo Sharing app.

```
User.find({}).select("_id first_name last_name").exec(
    function doneCallback(err, users) {
     if (err) {
         response.status(500).send(JSON.stringify(err));
     } else {
         response.end(JSON.stringify(users));
     }
    }
);
```

If you omit passing the function doneCallback to the exec method of the Mongoose query builder the method will return a promise like so:

```
var promise = User.find({}).select("_id first_name
last_name").exec();
```

Show the code you would need to write to convert the promise to send the Express.js response as in done in the callback approach shown above.

Problem #17 (12 points)

(A) Content distribution networks (CDNs) provide an interface where web application developers give them content and the CDN gives them an URL that will return the content if a HTTP GET request is issued to that URL.

For each the following parts of a web application, state if using a CDN to serve the part would be appropriate. Provide a short justification for your answer.

1. Company logo that is shown on every view.

2. Model data for popular views of the web application.

3. Controller code of popular views of the web application.

(B) Describe the mechanism used by the CDN so that the same URL (Universal Resource Locator) will use a different location (web service) depending on the physical location of the browser running the web application.

Problem #18 (12 points)

Express.js is a package for Node.js has a concept called **middleware** that allows functionality to be added to a web application backend. Pretty much any functionality that can be implemented using Express.js could be implemented as middleware but typically only certain type of functionality is done as middleware. For each of the following functionality you might add a web application like your CS142 Photo Sharing Website, state if implementing it as a middleware would good choice or not. Provide a brief justification for your answer.

A. Add a "read-only" user type to a web application were the user can run all the functionality that views information but cannot execute functionality that changes information.

B. Add resource quotas for users of the web application so that users can not create more of some resource like Comments in our Photo Sharing app.

C. Adding an objectionable world filter for all user generated content being added to the web application so that content like obscene or offensive words can be disallowed by the backend.

D. Add a debug tracing capability that if enabled allows the operations team to see all the backend REST calls made by a user.

Problem #19 (8 points)

When writing operating system code a technique called **busy waiting** is sometime used to pause program execution. Busy waiting looks something like:

```
while (!eventOfInterestHasHappened()) {
   continue;
}
```

and is frequently used to wait for some external entity such a device to do something.

JavaScript is also frequently used to interact with external entities (e.g. the user) yet busy waiting wouldn't be recommended. Explain why.

Problem #20 (8 points)

The code fragment below uses async and events module. Write out the console.log outputs in the order in which they will be printed for this code fragment.

```
var async = require('async');
var events = require('events');

var emitterOne = new events.EventEmitter();
var emitterTwo = new events.EventEmitter();

function sumWithOne(num, callback) {
    sum = 1 + num;
    emitterOne.emit('event 1');
    callback();
}

emitterOne.on('event 1', function() {
     console.log('A');
});
emitterTwo.on('event 1', function() {
     console.log('B');
});
emitterOne.on('event 2', function() {
     console.log('D');
});
emitterOne.on('event 2', function() {
     console.log('E');
});

console.log('Start');
async.each([1, 2], sumWithOne, function(err) {
    console.log('Async done');
    emitterOne.emit('event 2');
    emitterTwo.emit('event 2');
});
console.log('End');
```

Problem #21 (8 points)

With JavaScript objects it is possible to add method functions in two different ways as shown below for the class Rectangle:

Way (A):	Way (B):

`function Rectangle(width, height) {`
`  this.width = width;`
`  this.height = height;`
`  this.area = function() {`
`  return this.width *`
`  this.height;`
`  };`
`}`
	`function Rectangle(width, height) {`
	`  this.width = width;`
	`  this.height = height;`
	`}`
	`Rectangle.prototype.area = function(){`
	`  return this.width*this.height;`
	`};`

Describe why Way (B) is preferred.