CS 142 Final Examination

Spring Quarter 2023

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
Points	10	10	10	8	8	8	12	16	8	12
Problem	#11	#12	#13	#14	#15	#16	#17	#18	#19	Total
Points	10	10	8	8	8	8	8	10	8	180

Only the front side of the exam pages will be scanned. Do not write answers on the back of the pages.

Problem #1 (10 points)

As is widely done in computer networking protocols, the HTTP protocol is layered on top of the TCP/IP protocol. There are further protocol layers both below TCP/IP and above HTTP. For example, below TCP/IP are network and data link layers like Ethernet and WiFi.

A) Describe one of the layers above HTTP that were presented in class.

B) Sometimes when you have layer A built on top of layer B, it is possible to implement layer B on top of layer A. For example, it won't be efficient but you could implement Ethernet by sending and receiving packets over a TCP/IP connection. Would it be possible to implement TCP/IP on HTTP? Explain your answer.

Problem #2 (10 points)

Referential integrity was a serious concern in the early HyperText system. Broken references to pages that were deleted or moved were viewed as unacceptable. How did the HTTP designers envision handling references that:

A) Move to another location.

B) Are deleted.

Problem #3 (10 points)

The DOM's `XMLHttpRequest` class allows JavaScript to perform HTTP requests. In lecture, we presented the basic usage pattern:

```
let xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("GET", url);
xhr.send();
```

A) The documentation for the `open` method reports an additional optional parameter named async (e.g. `open(method, URL, async)`). The documentation reads:

> `async` An optional Boolean parameter, defaulting to true, indicating whether or not to perform the operation asynchronously. If this value is false, the send() method does not return until the response is received.

The use of this feature of `XMLHttpRequest` is strongly discouraged. Describe the reason for this and describe what the problem would be if you modified your PhotoApp to use `async=False` rather than Axios to fetch model data.

B) Explain why URLs that would work if entered in the browser's location bar aren't guaranteed to work if passed to an `XMLHttpRequest` request.

Problem #4 (8 points)

Although our simple React.js Photo App had components individually fetch their model data, it is possible with a JavaScript framework using model/view/controller decomposition like React.js to collect together all the component model data fetches of the current view and process them together. Would such an extension to collect the fetches together be more advantageous for REST or GraphQL backend APIs? Explain your answer.

Problem #5 (8 points)

Assume you are given two JavaScript functions that return Promises named `f1` and `f2` and the following JavaScript function:

```
async function runIt(a) {
  let f1result = await f1(a);
  let b = await f2(f1result);
  console.log(b);
}
```

Your boss decides your code can no longer use the `async` and `await` keywords. Write a version of the function that doesn't use these keywords. Your code should print the same `console.log` statement when called with the same parameter.

Problem #6 (8 points)

Node.js embedded a JavaScript runtime. JavaScript has a full-featured `String` data type and supports garbage collection but, until recently, didn't have good support for arrays of bytes, which are widely used by the system call interfaces of modern operating systems. Encoding an array of bytes into a JavaScript string in the JavaScript heap is a resource-intensive operation.

Much of the processing of web servers involves moving arrays of bytes between system calls. For example, processing a GET request for a file involves invoking a read system call to get an array of bytes and passing that array of bytes to a socket write system call. Explain how a web server written in Express.js can service GET requests without paying the cost of encoding the arrays of bytes into JavaScript strings.

Problem #7 (12 points)

Our PhotoApp backend used multiple Express middleware modules. List three of the modules, including a description of what the module does and what functionality would break if the module wasn't used.

Problem #8 (16 points)

A)

```
// A
app.put('/update', function (req, res) {
  // B
  User.findOne({_id: req.params.user_id}, function (err, user)
{
    // C
    user[req.body.prop] = req.body.value;
    // D
    user.save();
    // E
    res.send('success');
  });
  //F
});
// G
```

Although JavaScript has a single thread of control, not all JavaScript code can assume no other JavaScript execution can be performed between two points in the code. For each of the following pairs of letters, state if other JavaScript execution could be performed between the letters.

A,B

B,C

C,D

D,E

A,G

B,F

Problem #8 continued….

B) If there are no other requests active in the system, what can the requestor in the browser assume about the state of the database when the response is received? Can the requestor assume the database has been updated or might the database still hold the old value of the User object?

Problem #9 (8 points)

Describe the issues with the SQL databases that lead to web applications preferring no-SQL databases like MongoDB.

Problem #10 (12 points)

The REST APIs allow CRUD operations on single resources named by the URL. The HTTP verb specifies which of the CRUD operations to perform. A REST CRUD operation can result in the resource being changed in the database and potentially database indexes being updated.

For each of the HTTP verbs below, state if the operations would never update an index, update a single index only, or potentially update multiple indexes.

A) GET

B) PUT

C) POST

D) DELETE

Problem #11 (10 points)

Both the UI design using the MVC structure and the Mongoose ODL used the same word, "model", to describe a collection of data. Even if the two parts (MVC and Mongoose) are referring to the same concept (e.g User in our PhotoApp) they might not have the same property.

A) Is there ever a possibility that the MVC model is a subset of the properties of the Mongoose model? If so, give an example.

B) Is there ever a possibility that the MVC model is a superset of the properties of the Mongoose model? If so, give an example.

Problem #12 (10 points)

Assume you download a new version of the Chrome browser and it has an unfortunate bug of sometimes forgetting to attach the origin's cookies to HTTP requests sent out. Assume this bug occurred infrequently but did happen at least once an hour.

A) Describe what kind of failures (you would see) if you ran your PhotoApp on this buggy browser.

B) Is there a simple workaround you could suggest to the user when they hit the bug that would allow them to continue using the application, or would they need to get a browser to get any use of the PhotoApp?

Problem #13 (8 points)

The base functionality of the PhotoApp didn't change the session state except during login and logout. Assume you do a Project 8 story that makes most Express handlers update the session state (e.g., keep a display of an estimated number of logged-in users). Would you expect Express session to have to respond with a `set-cookie` header on most responses? Explain your answer.

Problem #14 (8 points)

One of the exciting new features of React.js when it was introduced was rather than rendering fully in JavaScript in the browser, the developer has the option of rendering the view into HTML that is downloaded and displayed in the browser. This was called server-side rendering and it became a must-have capability for JavaScript frameworks. Describe what problems might occur if you used server-side rendering for views that take user input.

Problem #15 (8 points)

When using HTTPS to communicate between your browser and your backend web server, who knows the encrypt key being used? Choose from the list:

- Browser,
- Web server,
- Certificate authority, or
- Someone else.

Explain your answer.

Problem #16 (8 points)

Explain why cross-site request forgery (CSRF) attacks can happen on certain HTTP verbs that are used in REST APIs (like GET and POST) but not as likely for other verbs (like PUT and DELETE).

Problem #17 (8 points)

Describe how message authentication codes (MACs) can provide integrity and authentication but not confidentially.

Problem #18 (10 points)

Answer the following questions about Content Distribution Networks (CDNs).

A) Describe the issue that makes CDNs not suitable for rapidly changing content.

B) What would be the problem seen if you put content that was updating frequently?

Problem #19 (8 points)

Describe the attributes of a scale-out web server architecture that make it easier to handle large variability in load compared to scale-out data storage systems.