

CS312 Spring 2023 – Midterm 2

Name: _____

Date: _____ Start time: _____ End time: _____

Honor Code: _____

Signature: _____

This exam is open course web page, open Ed, open notes, open slides, open your assignment solutions and open calculator, but closed everything else (e.g., consulting with others and searching online are not permitted). **You have 3 hours in a single sitting to complete the exam.** Read the problem descriptions carefully and write your answers clearly and *legibly* in the space provided. Circle or otherwise indicate your answer if it might not be easily identified. You may use extra sheets of paper, stapled to your exam, if you need more room, as long as the problem number is clearly labeled and your name is on the paper. If you attached extra sheets indicate on your main exam paper to look for the extra sheets for that problem.



Learning Target	Assessment
1	
2	
3	
4	
5	
6	
7	
8	

Question 1. User stories

You are developing a web application for managing the queue for in-person office hours appointments. When interviewing stakeholders, multiple respondents described e-mail notifications for individuals as they approached the front of the queue. Write two I.N.V.E.S.T. user stories for this feature, one from the perspective of a student in the queue for office hours, the other from the perspective of an instructor holding office hours. Your user stories will be evaluated on format and quality.

(a) Student:

(b) Instructor:

Question 2. Javascript

Assume `wait(sec)` returns a promise that resolves after `sec` seconds have elapsed:

```
1 function first() {  
2   return wait(3).then(() => console.log(1));  
3 }  
4  
5 function second() {  
6   return wait(2).then(() => console.log(2));  
7 }  
8  
9 function third() {  
10  return wait(1).then(() => console.log(3));  
11 }
```

(a) What will the following code print?

```
first();  
second();  
third();
```

(b) Using only the three functions above and either the Promise API, e.g., `then` methods, or `await`, write code that prints 1, 2, 3 in that order. You must use the 3 functions above without modification. No additional `console.log` statements or other code that isn't `then` with a callback or `await` is allowed.

Question 3. Testing

Imagine you are developing a reminder application, where each reminder object has a **description** string and **dueDate** date. You have implemented a component, **ReminderCreator**, containing input elements and a “Create” button to create new reminders. When the user clicks “Create”, the component invokes a callback with the reminder object. If the user doesn’t provide an item description or enters a date in the past, the “Create” button is disabled and the form provides error feedback. Using the skeleton below, implement **pseudo-code** for F.I.R.S.T. integration tests to verify that creating a valid reminder invokes the callback, and that a reminder with a past date disables the “Create” button and results in error feedback. We measure error feedback as the date input element having the attribute **"aria-invalid"** set to **"true"**. You do **not** need to provide executable Javascript, instead describe the steps of your test as pseudo-code. For example, one of the steps in your answer might be:

Assert date input element has attribute "aria-invalid" with value "true"

```
describe("Reminder creation", () => {
  test("Creating valid item invokes callback", () => {
```

```
  });
  test("Past date shows error feedback", () => {
```

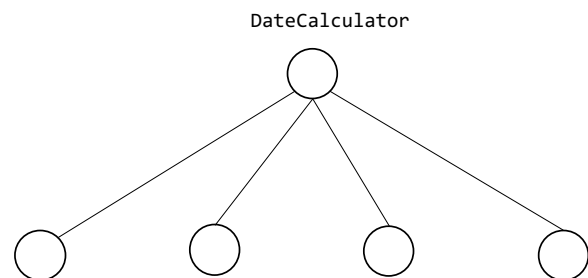
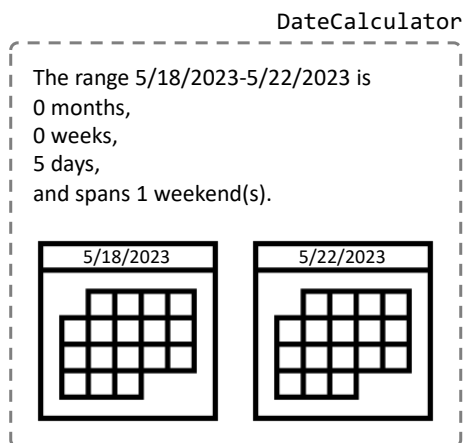
```
    });
  });
```

Question 4. Scenarios

Imagine you are writing a React component to display a list of products in an e-commerce store. Your component has a dropdown to display the items sorted in ascending order of cost (low to high) or descending order of cost (high to low). Write a Gherkin-style test scenario that covers this behavior. You do not need to provide the implementation details of the tests, just describe the scenario for the test.

Question 5. React

You are implementing an application for displaying statistics about date ranges in React. Outline and label the wireframe (below, left) with a possible set of components. Label the tree (below, right) with components to show the hierarchy, *including* labeling the tree nodes with state implemented in that component and labeling the tree edges with props passed to each component (similar to the figure in programming assignment 2). The top-level component **DateCalculator** is labeled for you. Assume you have a Javascript library that can compute statistics for arbitrary date ranges and that your component library already provides a controlled calendar-based **DatePicker** component (like shown in the wire-frame). Any implementation reflecting good React practices will be accepted. You may not need all the nodes in the tree; cross out any unused nodes. Your component, state and prop names should be clear.



Question 6. REST

For each of the following pages in a NextJS-based web application, provide an appropriate RESTful front-end (browser) URL for that page and, where relevant, an appropriate RESTful server API endpoint (HTTP verb and URL) that component would interact with. An example is provided is below.

Page	Page URL	API HTTP verb and URL
Add a new article to Simplepedia	/edit	POST /api/articles
View a patron's loaned (checked-out) items in a library management application	_____	_____
Renew a patron's loaned item to extend the due date in a library management application	_____	_____
Search results for items in a library collection with "fire" in the title	_____	_____

Question 7. Data modeling

Assume you are developing an application for managing class rosters at an institution where students take multiple classes and faculty members teach multiple classes, but each class is only taught by one faculty member.

- (a) Identify the models you would define in your server backend to implement the following user story:
- As a faculty member, I want to be able to view my current class roster, so that I know who is enrolled in my class
- (b) Choose ONE answer. In a normalized schema designed for a relational database (RDBMS), how would you best store link between faculty members and the classes they teach?
- ☐ An array of class IDs stored in a **Faculty** table row
 - ☐ An array Faculty IDs stored in the **Class** table row
 - ☐ A single **Class** ID stored in a **Faculty** table row
 - ☐ A single **Faculty** ID stored in the **Class** table row
- (c) How would your answer to the part above change if a single class could be taught by multiple faculty members with different factions of responsibility (i.e., faculty member one is 75% responsible and faculty member two is 25% responsible)? Describe the association that models the relationship *and* note any attributes needed to support this feature. Your answer should use the association vocabulary from class.

Question 8. Development processes

For each of the following series of commands, indicate whether it is consistent with our in-class development and deployment processes or not. If not, briefly explain why those actions would be problematic.

<hr/>	<input type="radio"/> Consistent with class practices
<code>git checkout -b edit_article</code>	<input type="radio"/> <i>Not</i> consistent with class practices
<code>...</code>	
<code>git add .</code>	
<code>git commit -m "New page ..."</code>	
<code>git push origin edit_article</code>	
<hr/>	<input type="radio"/> Consistent with class practices
<code>git checkout main</code>	<input type="radio"/> <i>Not</i> consistent with class practices
<code>...</code>	
<code>git add .</code>	
<code>git commit -m "New page ..."</code>	
<code>git push origin main</code>	
<hr/>	<input type="radio"/> Consistent with class practices
<code>git checkout -b edit_article</code>	<input type="radio"/> <i>Not</i> consistent with class practices
<code>...</code>	
<code>git add .</code>	
<code># --no-verify bypasses pre-commit hooks</code>	
<code>git commit --no-verify -m "New page ..."</code>	
<hr/>	<input type="radio"/> Consistent with class practices
<code>git checkout -b edit_article</code>	<input type="radio"/> <i>Not</i> consistent with class practices
<code>...</code>	
<code>git add .</code>	
<code>git commit -m "New page ..."</code>	
<code>git fetch</code>	
<code>git merge origin/main</code>	
<code>git push origin edit_article</code>	
<hr/>	

Page intentionally blank.