CS312 Spring 2023 – Midterm Solution

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 2 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 a library. When interviewing stakeholders, multiple individuals described presenting an alert in the user interface when a patron has books due within the next week. Write two I.N.V.E.S.T. user stories, one from the perspective of a library patron, the other from the perspective of a librarian working at the circulation desk. Your user stories will be evaluated on format and quality.

(a) Library patron:

Solution: As a library patron,
I want receive an alert when logged and visiting the library web page that I have books due
So that I remember to return books before they are overdue.

(b) Librarian working at the circulation desk:

Solution: As a librarian checking someone out,
I want receive an alert that they have library books due
So that I can remind them during the checkout process to return the books before they are overdue.

Question 2. Javascript

Assume `wait(sec)` returns a promise that resolves after `sec` have elapsed. Consider the following two implementations:

```
1  function first() {
2    console.log(1);
3  }
4
5  function second() {
6    wait(1).then(() => console.log(2));
7  }
8
9  function third() {
10   console.log(3);
11 }
12
13 first();
14 second();
15 third();
```

```
1  function first() {
2    console.log(1);
3  }
4
5  async function second() {
6    await wait(1);
7    console.log(2);
8  }
9
10 function third() {
11   console.log(3);
12 }
13
14 first();
15 await second();
16 third();
```

Write the expected output for left-side code below on the left If the right-side code produces the same result indicate below, otherwise provide the expected output below on the right.

◯ Both snippets produce the same output

Solution:

```
1
3
2
```

Solution:

```
1
2
3
```

Question 3. Testing

In Simplepedia we largely ignored `fetch` errors. Imagine we are extending our `Editor` implementation (with a server) to report any errors sent by the server to the user so they can correct their submission. Using the skeleton below, implement **pseudo-code** for a F.I.R.S.T. integration test to verify that when adding an article with a duplicate title, an error response from the server results in error feedback. We measure error feedback as the title 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 title input element has attribute "aria-invalid" with value "true"

You may or may not need all of the functions below. You only need to include pseudo-code in bodies of the functions relevant to your answer.

```
describe("Error reporting when adding an article", () => {
  beforeEach(() => {
```

Solution:

```
Mock the POST to /api/articles to return an error (with duplicate title message)
```

```
  });
  afterEach(() => {
```

Solution:

```
Clear the mocks
```

```
  });
  test("Duplicate title shows error feedback, () => {
```

Solution: An integration test would need to involve both the `Editor` component and the parent component that performs the `fetch`.

```
Render the SimplepediaCreator component (containing Editor)
Find and fill the title and content input elements with sample values
Assert title input element does not have attribute "aria-invalid" with value "true".
Find "Save" button and simulate click.
Assert that Editor is still showing.
Assert that the URL is still pointing to the create article page
Find the title input.
Assert title input element has attribute "aria-invalid" with value "true".
Assert mock fetch was performed.
```

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

Question 4. Scenarios

Imagine you are writing a React implementation of a blog. The `Post` component will either show a short version of the post truncated after the first sentence, or the full post. In the truncated view the user can click on a "Show more" button to show the article and in the full view click on the "Show less" button to truncate the test. 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.

Solution:

We want to make sure to test both the change in view form truncated to full and back again.

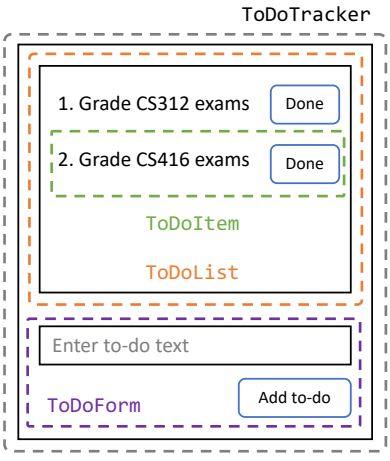
```
Given that the post text is "First sentence. Second sentence." and short mode is true
Then "First sentence." should be shown for the post body
When I click on the "Show more" button
Then "First sentence. Second sentence." should be shown for the post body
When I click on the "Show less" button
Then "First sentence." should be shown for the post body
```

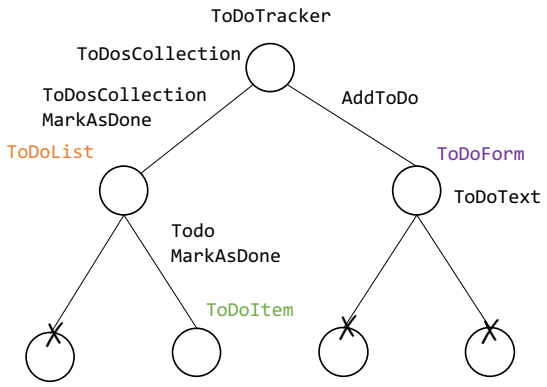
Question 5. React

You are implementing the following view for a simple To-Do tracking application with 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. Further label the tree nodes with state implemented in that component and label the tree edges with props passed to each component (similar to the figure in programming assignment 2). The top-level component `ToDoTracker` is labeled for you. 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 sufficiently descriptive that their role is clear.

Solution:

Common issues were not including a callback to effect "Done" and not including state for the controlled input element.





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 the main page for a course on a Canvas-like site	**/courses/1**	**GET /courses/1**
View an assignment as part of a course on a Canvas-like site	**/courses/1/assignments/1**	**GET /courses/1/assignments/1**
Edit the course main page on a Canvas-like site	**/courses/1/edit**	**PUT /courses/1**

Question 7. Data modeling

For the following question, you can assume the application serves a single library, that is your data model can assume a single library.

(a) Identify the models you would define in your server backend to implement the following user story: As a library visitor with or without an account, I want to be able to check if the library has a specific book by searching for the title, so I can easily learn if that library has it in their collection.

> **Solution:** The minimum model would be `Book`. The title would be an attribute of `Book`, not a separate model. Since the database only contains availability for a single library there would not need to be a `Library` model. Since availability is public and the visitor may not be logged in, this user story would not require a `User` model.

(b) Choose ONE answer. When creating an application for libraries to track their collection, you decided to create an `Author` model. Which association between `Author` and `Book` best models this relationship?

- ◯ An `Author` has one `Book`, an `Book` belongs to an `Author`.
- ◯ A `Book` has many `Author`s, an `Author` belongs to a `Book`.
- √ **An `Author` has many `Book`s, a `Book` has many `Author`s.**

> **Solution:** A book can have multiple authors and an author can have multiple books and so we need a many-to-many relationship.

(c) You want to implement a features where patrons can put holds on books that are currently checked out, e.g., put themselves on a first-come-first-serve wait list. What association between `Patron` and `Book` would model this relationship. Your answer should be formatted similarly to the answers to the previous part. In your answer note any Model attributes needed to support this feature.

> **Solution:** There is a many-to-many relationship between the `Patron` and `Book` models, i.e., a patron have put a hold on many books and a book can have many holds. We would introduce `Hold` model that stored the date and time the hold was placed to enable us to find the oldest hold when a book is returned. Thus the association would be a `Patron` has many `Book`s through `Hold`, a `Book` has many `Patron`s through `Hold`.

Question 8. Development processes

In class we practice continuous delivery, in which we can deploy our applications on demand with a single command. An alternative is continuous deployment, in which each integration to the `main` branch is automatically deployed.

(a) What technical changes, if any, in our project workflow to enable continuous deployment?

> **Solution:** Add a step to the GitHub actions to automatically deploy main after a pull request is completed and the merged code passes the test suite.

(b) Would we need to make other changes to our processes to enable continuous deployment? Your answer should specifically address testing and the role of our different branches.

> **Solution:** No. Our stated process is that the `main` branch should always be deployable, so the only difference is when that deployment occurs.

(c) One challenge for continuous delivery is coordinating with specific events, e.g., our sprint demo or marketing push. How could we control and coordinate feature availability?

> **Solution:** We could use feature flags to coordinate and control feature availability (for selective rollout, to coordinate with specific events/marketing, etc.). We would deploy the change but not turn on the feature until the appointed time or for a specific user.

Page intentionally blank.