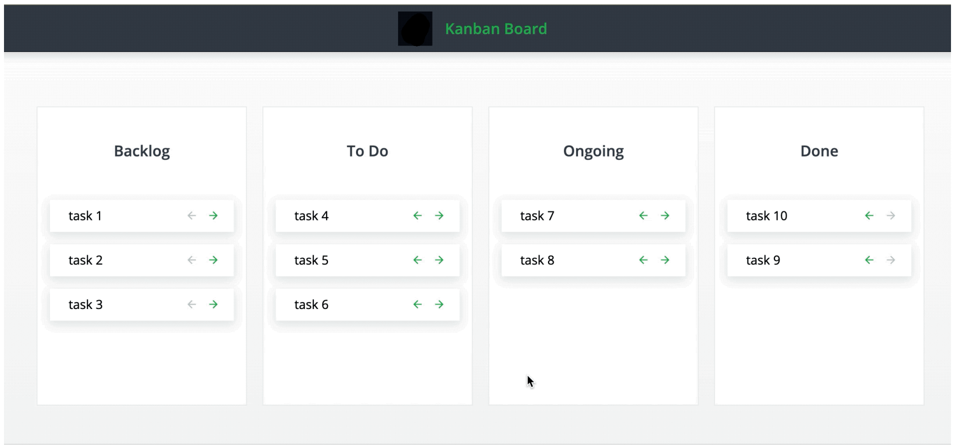
[redacted]

Question - 1
React: Kanban Board

Kanban is a popular workflow used in task management, project management, issue tracking, and other similar purposes. The workflow is usually visualized using a Kanban Board.

Create a Kanban Board component with tasks, where each task consists of a name only, as shown below:

Hide animation



The component must have the following functionalities:

- The component board contains 4 stages of tasks in the sequence 'Backlog', 'To Do', 'Ongoing', and 'Done'.
- An array of tasks is passed as a prop to the component.
- In every individual stage, the tasks are rendered as a list <ul>, where each task is a single list item <li> that displays the name of the task.
- Each task list item has 2 icon buttons on the right:
  1. Back button: This moves the task to the previous stage in the sequence, if any. This button is disabled if the task is in the first stage.
  2. Forward button: This moves the task to the next stage in the sequence, if any. This button is disabled if the task is in the last stage.
- Each task has 2 properties:
  - name: The name of task. This is the unique identification for every task. [STRING]
  - stage: The stage of the task. [NUMBER] (0 represents the 'Backlog' stage, 1 represents the 'To Do' stage, 2 represents the 'Ongoing' stage, and 3 represents the 'Done' stage)

The following data-testid attributes are required in the component for the tests to pass:

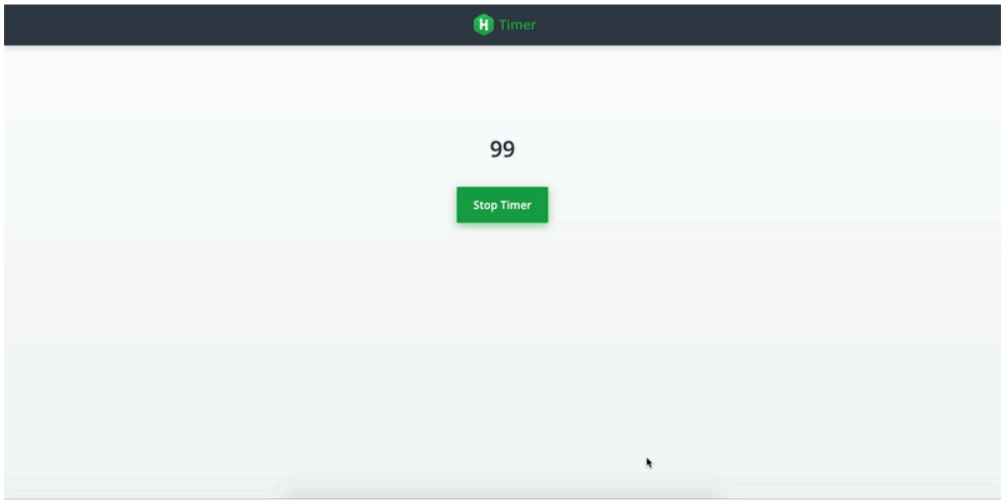
- The <ul> for the 'Backlog' stage should have the data-testid attribute 'stage-0'.
- The <ul> for the 'To Do' stage should have the data-testid attribute 'stage-1'.
- The <ul> for the 'Ongoing' stage should have the data-testid attribute 'stage-2'.
- The <ul> for the 'Done' stage should have the data-testid attribute 'stage-3'.
- Every <li> task should follow these guidelines:
  1. The <span> containing the name should have the data-testid attribute 'TASK_NAME-name', where TASK_NAME is the name of the task joined by a hyphen symbol. For example, for the task named 'task 0', it should be 'task-0-name'. For the task named 'abc', it should be 'abc-name'.
  2. The back button should have the data-testid attribute 'TASK_NAME-back', where TASK_NAME is the name of the task joined by a hyphen symbol. For example, for the task named 'task 0', it should be 'task-0-back'. For the task named 'abc', it should be 'abc-back'.
  3. The forward button should have the data-testid attribute 'TASK_NAME-forward', where TASK_NAME is the name of the task joined by a hyphen symbol. For example, for the task named 'task 0', it should be 'task-0-forward'. For the task named 'abc', it should be 'abc-forward'.

Please note that components have the above data-testid attributes for test cases and certain classes and ids for rendering purposes. It is advised not to change them.

Question - 2
React: Timer Component

Create a timer component like the one shown.

Hide animation



The component has the following functionalities:

- The timer value decreases by 1 every second. For example, if the initial value is 100, after 1 second it becomes 99, etc.
- The value starts to decrease when the component is mounted.
- The initial value of the timer is set by a prop, *initial*.
- Once the counter reaches 0, it should stop.
- The 'Stop Timer' button stops the timer at its current value.

The following data-testid attributes are required in the component for the tests to pass:

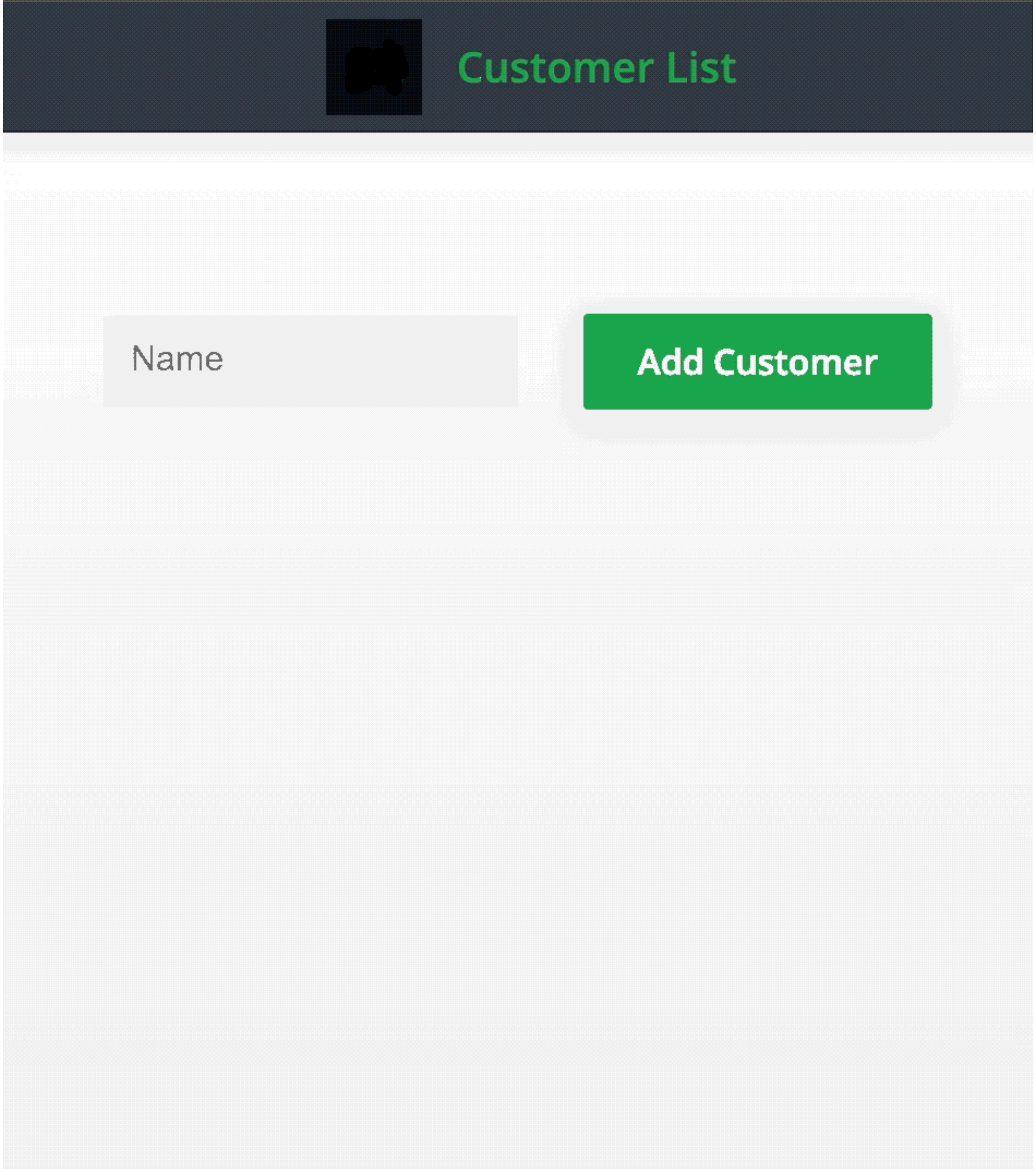
Component	Attribute
Title	*app-title*
Timer value	*timer-value*
Stop timer button	*stop-button*

Please note that components have these *data-testid* attributes for test cases and certain classes and ids for rendering purposes. They should not be changed.

Question - 3
React: Customer List

Create a customer list component as shown.

Hide animation



The component must have the following functionalities:

- The input should initially be empty.
- If no value is entered, clicking on the 'Add Customer' button should not do anything.
- Clicking on the 'Add Customer' button should add the input value to the list below. For this, add *<li>{input}</li>* to the <ul data-testid="customer-list"> element.
- After the value is added to the list, it should clear the value in the input box.
- Please note that the customer list <ul> element should only be rendered if it has at least one customer added, i.e. at least one <li> child. When the app is mounted, since no customers are added, the <ul> element should not be rendered.
- All the values added by the button should be rendered in the list below.

The following *data-testid* attributes are required in the component for the tests to pass.

Component	Attribute
Input	*app-input*

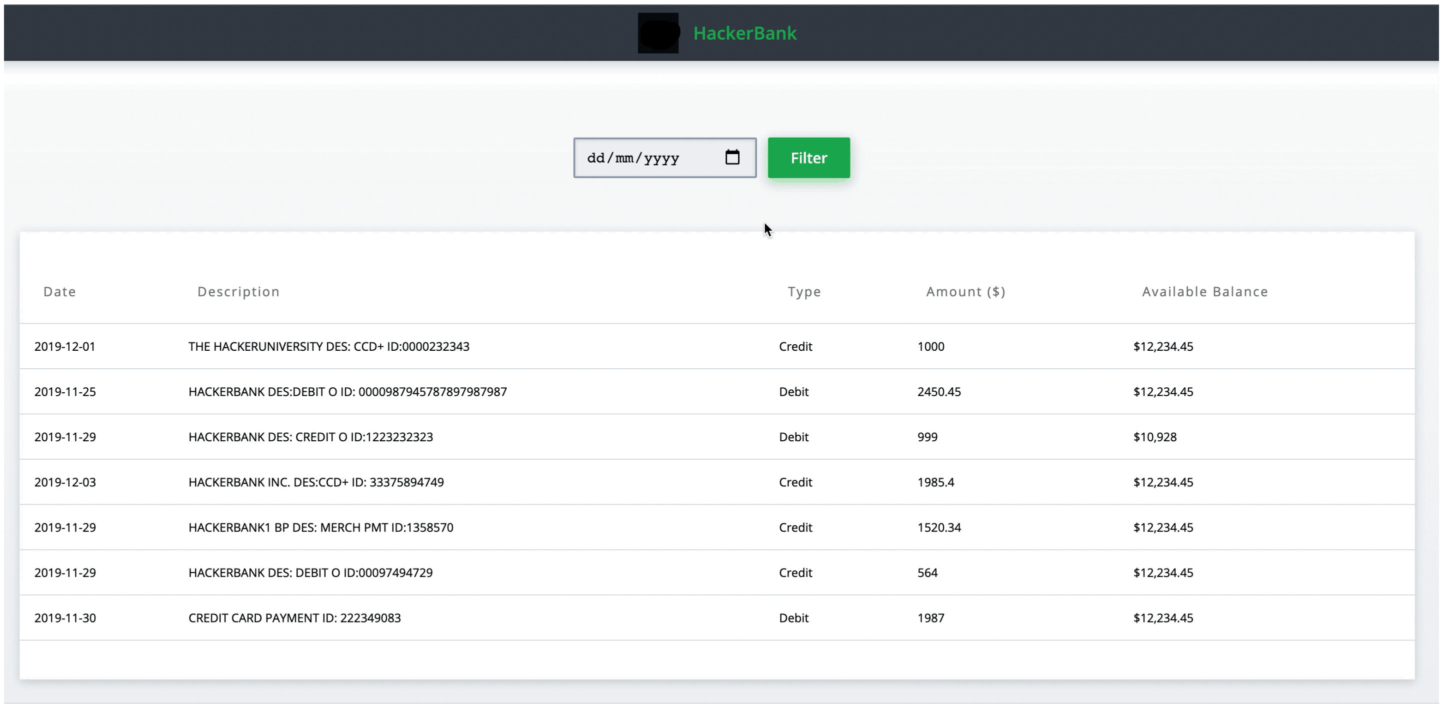
Button	*submit-button*
Customer list <ul>	*customer-list*
List elements	*list-item0, list-item1, ...*

Please note that components have these *data-testid* attributes for test cases, and certain classes and ids for rendering purposes. They should not be changed.

Question - 4
React: HackerBank

Create a banking application as shown below. Some core functionalities have already been implemented, but the application is not complete. Application requirements are given below, and the finished application must pass all of the unit tests.

Hide animation



HackerBank

dd/mm/yyyy Filter

Date	Description	Type	Amount ($)	Available Balance
2019-12-01	THE HACKERUNIVERSITY DES: CCD+ ID:0000232343	Credit	1000	$12,234.45
2019-11-25	HACKERBANK DES:DEBIT O ID: 00009879457878979879 87	Debit	2450.45	$12,234.45
2019-11-29	HACKERBANK DES: CREDIT O ID:1223232323	Debit	999	$10,928
2019-12-03	HACKERBANK INC. DES:CCD+ ID: 33375894749	Credit	1985.4	$12,234.45
2019-11-29	HACKERBANK1 BP DES: MERCH PMT ID:1358570	Credit	1520.34	$12,234.45
2019-11-29	HACKERBANK DES: DEBIT O ID:00097494729	Credit	564	$12,234.45
2019-11-30	CREDIT CARD PAYMENT ID: 222349083	Debit	1987	$12,234.45

Implement the following functionalities:

- All transactions are initially displayed inside the table in the order they are retrieved from the source. The source is passed down as '*txns*' prop to TransactionTable component in App.js.
- Picking the date from the date input and pressing the 'Filter' button should display all the records for that date in the table. If no date is chosen, the 'Filter' button should not do anything.
- Clicking on the 'Amount ($)' table header should sort the records in ascending order of amount. The behavior is the same for multiple clicks on 'Amount ($)'.

Each transaction object contains the following properties :

- *String* date: The date when the transaction took place in the format YYYY-MM-DD.
- *String* description: The description of the transaction.
- *Number* type: The type of transaction, where 0 denotes a credit transaction and 1 denotes a debit transaction.
- *Float* amount: The total amount of the transaction.
- *String* balance: The balance of the account after the transaction was completed, prefixed with a dollar sign ($).

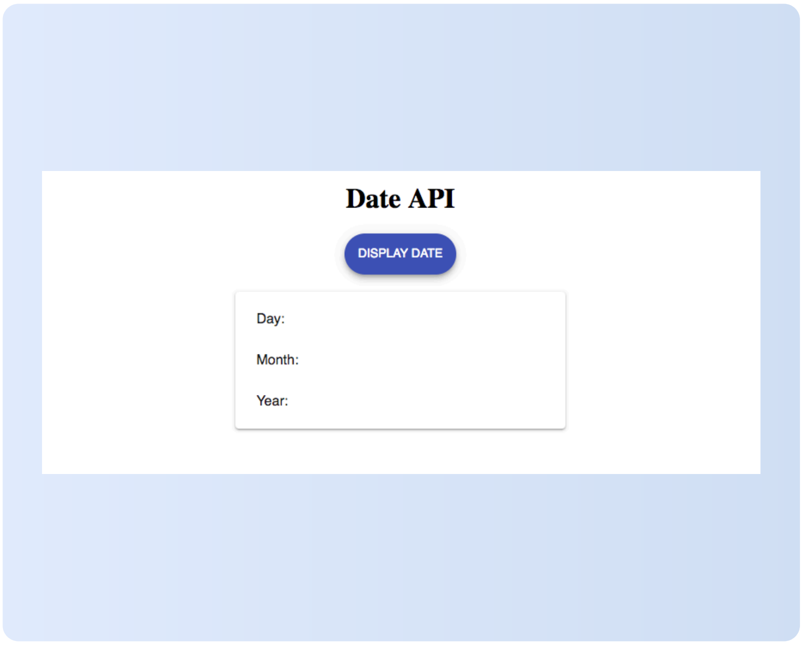
```
{
    "date": "2019-12-03",
    "description": "HACKERBANK INC. DES:CCD+ ID: 33375894749",
    "type": 0,
    "amount": 1985.4,
    "balance": "$12,234.45"
}
```

Question - 5
React: Date API

In the given single page React project, a button click prompts an HTTP GET request to an API endpoint that returns a *date* as its API response. Complete the project so it displays the current day, month, and year after clicking the button. Model the implementation after the instructions below.
Certain core React functionalities have already been implemented. Complete the React application in order to pass all the unit tests.

Demo

- Link to an animated GIF
- Scroll sideways below to explore the screenshots. Clicking each screenshot opens it in a new window.


Date API

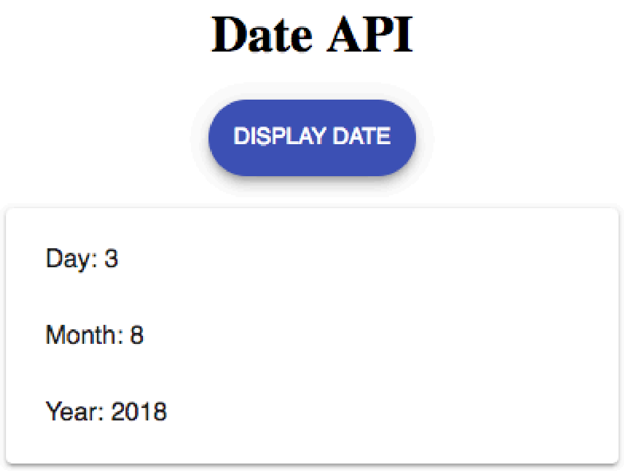
DISPLAY DATE

Day:

Month:

Year:

Application's initial state before clicking the button.


Date API

DISPLAY DATE

Day: 3

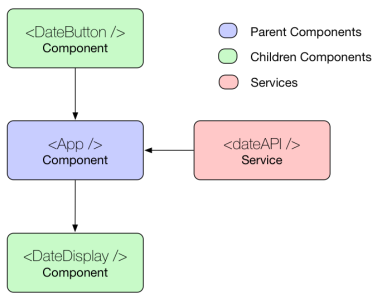
Month: 8

Year: 2018

Display date after clicking the button.

Specifications

- Component flow diagram of the application is as follows:



▼ Framework Specific Instructions

- The project uses React 16 by default. Changing the React version may interfere with tests and is strictly discouraged. You may refer to the React 16 docs here.
- The project uses create-react-app and react-scripts to automate serving and testing of the application.
- The project uses enzyme as a testing framework. Please refrain from changing the test framework or the tests themselves.
- The project uses React Material UI as a design framework. API docs are available here and can be used as a reference.