Sprint : Introduction to React.js

Description Submission History

Step 3/3

Your assignment has been checked by a code reviewer. Some mistakes have been found.

Update the code and resubmit it

GitHub is set up

Assignment has been submitted

Some mistakes have been found

↓Expand to see the whole summary↓

Hello, Sabrina!

Great job on submitting your project 10! This is your first project related to `React`! I hope, it was interesting for you!

Parts I really liked in your project:

- Good job that you `catch` possible errors at the end of server requests
- Good job that you've used `_id` for the `key` rather than the `index`.
- Good job that the submit button is only in `ModalWithForm`, and its text is controlled

There are few things that need to be corrected in your project. They're mostly minor issues that are easy to fix.

Needs correcting:

- All comments in the code marked `Needs correcting` and the comments below:
- Please, reuse the `handleCloseModal` function for any closing, not to duplicate the same code
- `<>` empty tags are not needed if you have a common tag that wraps the whole markup of the component
- The text near the `input` should be in a `label` tag, according to the semantics

- In `Footer` you need to use the `footer` tag, according to the semantics

There are some comments in your project that can help you make it even better. Please check the `Could be improved` comments.

A reminder: It’s necessary to fix all `Needs correcting` comments to get the project accepted. And `Could be improved` comments are optional, they will be up to you if you want to implement them.

Your hard work on this project is truly commendable. While there are still some adjustments needed. If you need any assistance with the feedback or understanding certain concepts, our tutors are here to help. Keep moving forward and maintain your dedication — you are doing an amazing job! ✨

reviewer

Collapse

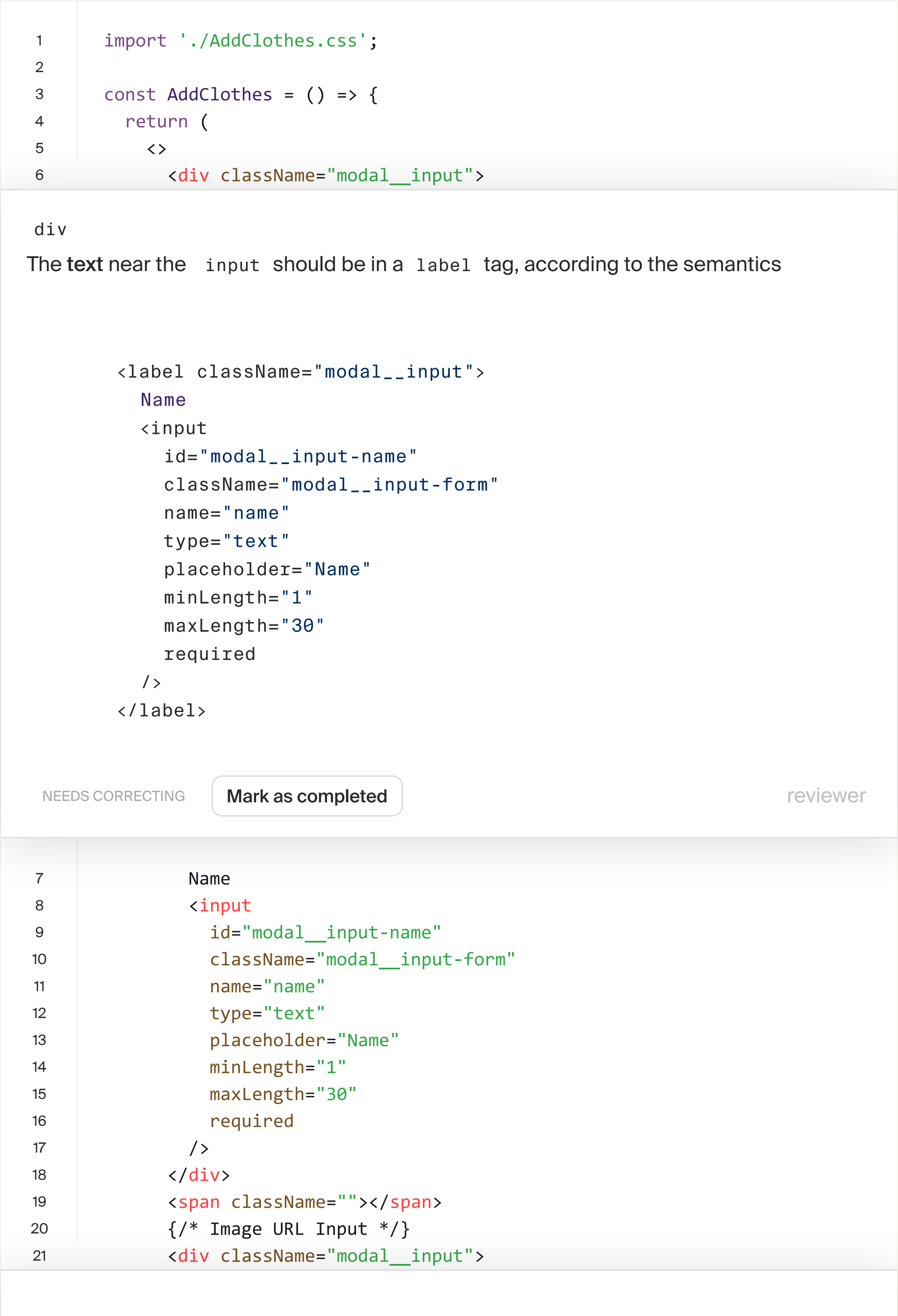
How was the review?

👍 👎

Your project with feedback

Expand all

Seyonce12-se_project_react-b902cce	12	1	5
src	12	1	5
components	12	1	5
AddClothes			2
AddClothes.css			No comments
AddClothes.jsx			2



```
import './AddClothes.css';

const AddClothes = () => {
  return (
    <>
      <div className="modal__input">
```

`div`

The **text** near the `input` should be in a `label` tag, according to the semantics

```
        <label className="modal__input">
          Name
          <input
            id="modal__input-name"
            className="modal__input-form"
            name="name"
            type="text"
            placeholder="Name"
            minLength="1"
            maxLength="30"
            required
          />
        </label>
```

```
          Name
          <input
            id="modal__input-name"
            className="modal__input-form"
            name="name"
            type="text"
            placeholder="Name"
            minLength="1"
            maxLength="30"
            required
          />
        </div>
        <span className=""></span>
        {/* Image URL Input */}
        <div className="modal__input">
```

`div`

The text near the `input` should be in a `label` tag, according to the semantics

```
<label className="modal__input">
```

NEEDS CORRECTING

Mark as completed

reviewer

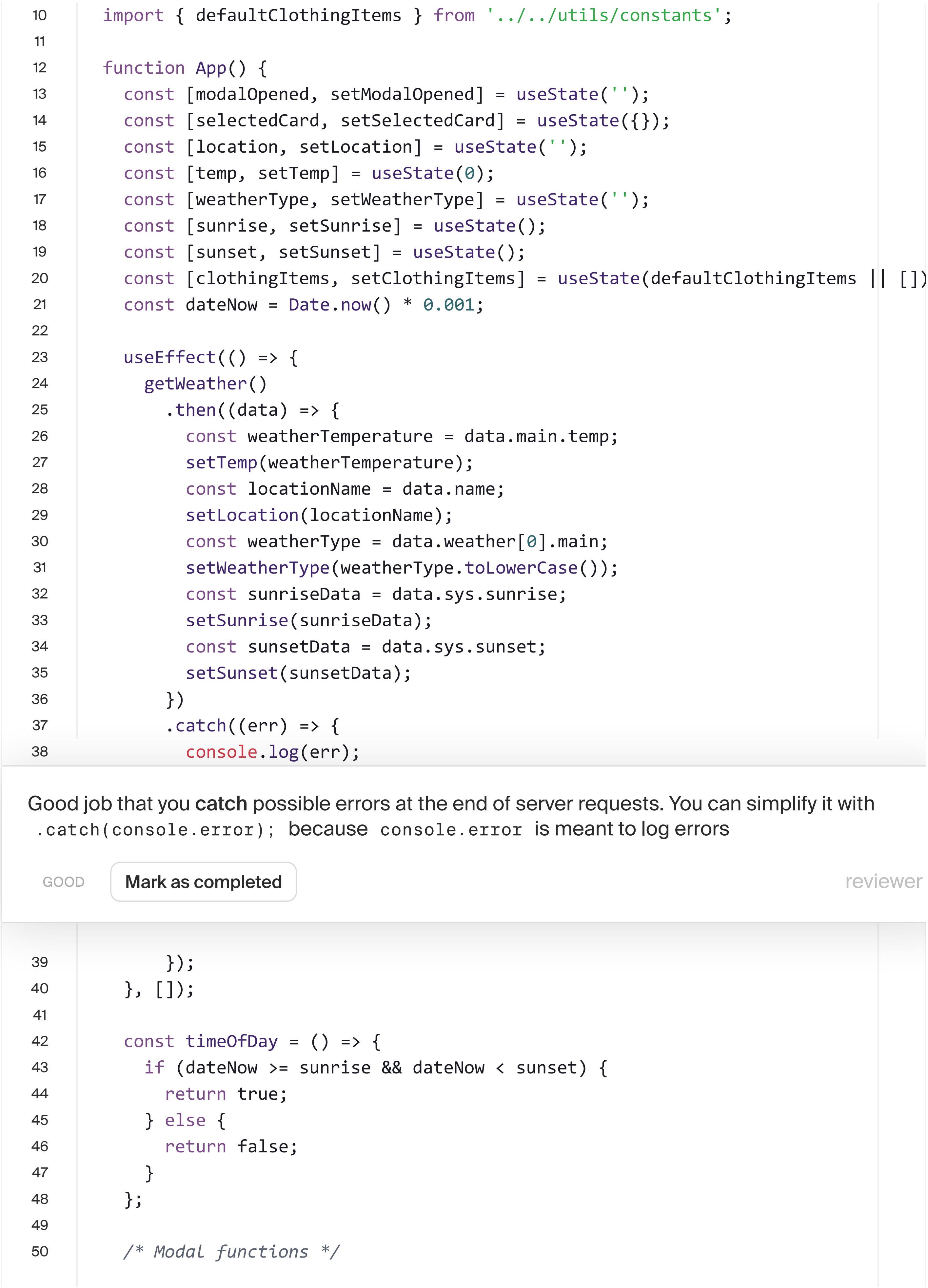
```
            Image
            <input
              id="modal__input-url"
              className="modal__input-form"
              name="link"
              type="url"
              placeholder="Image URL"
              required
            />
          </div>
          <span className=""></span>
          {/* Select weather and radio inputs */}
          <p className="modal__radio-title">Select weather type:</p>
          <div className="modal__radio-container">
            <div className="radio__btn-container">
              <input
                className="radio__btns"
                type="radio"
                id="hot"
                name="weather"
                value="hot"
              />
              <label
                htmlFor="hot"
                className="radio__btn-label"
              >
                Hot
              </label>
            </div>
            <div className="radio__btn-container">
              <input
                className="radio__btns"
                type="radio"
                id="warm"
                name="weather"
                value="warm"
              />
              <label
                htmlFor="warm"
                className="radio__btn-label"
              >
                Warm
              </label>
            </div>
            <div className="radio__btn-container">
              <input
                className="radio__btns"
                type="radio"
                id="cold"
                name="weather"
                value="cold"
              />
              <label
                htmlFor="cold"
                className="radio__btn-label"
              >
                Cold
              </label>
            </div>
          </div>
        </>
      );
    };

    export default AddClothes;
```

App 7 | 1

App.css No comments

App.jsx 7 | 1

```
import { useEffect, useState } from 'react';
import './App.css';
import Header from '../Header/Header';
import Main from '../Main/Main';
import Footer from '../Footer/Footer';
import ModalWithForm from '../ModalWithForm/ModalWithForm';
import ItemModal from '../ItemModal/ItemModal';
import AddClothes from '../AddClothes/AddClothes';
import { getWeather } from '../../utils/weatherApi';
```



```
import { defaultClothingItems } from '../../utils/constants';

function App() {
  const [modalOpened, setModalOpened] = useState('');
  const [selectedCard, setSelectedCard] = useState({});
  const [location, setLocation] = useState('');
  const [temp, setTemp] = useState(0);
  const [weatherType, setWeatherType] = useState('');
  const [sunrise, setSunrise] = useState();
  const [sunset, setSunset] = useState();
  const [clothingItems, setClothingItems] = useState(defaultClothingItems || [])
  const dateNow = Date.now() * 0.001;

  useEffect(() => {
    getWeather()
      .then((data) => {
        const weatherTemperature = data.main.temp;
        setTemp(weatherTemperature);
        const locationName = data.name;
        setLocation(locationName);
        const weatherType = data.weather[0].main;
        setWeatherType(weatherType.toLowerCase());
        const sunriseData = data.sys.sunrise;
        setSunrise(sunriseData);
        const sunsetData = data.sys.sunset;
        setSunset(sunsetData);
      })
      .catch((err) => {
        console.log(err);
```

Good job that you **catch** possible errors at the end of server requests. You can simplify it with `.catch(console.error);` because `console.error` is meant to log errors

```
      });
  }, []);

  const timeOfDay = () => {
    if (dateNow >= sunrise && dateNow < sunset) {
      return true;
    } else {
      return false;
    }
  };

  /* Modal functions */
  useEffect(() => {
```

You should add an `Escape` listener like this to be able to remove it **correctly** when modals are closed:

```
useEffect(() => {

  if (!modalOpened) return; // stop the effect not to add the listener if
```

NEEDS CORRECTING

Mark as completed

reviewer

```
    const handleEscClose = (evt) => {
      if (evt.key === 'Escape') {
        setModalOpened('');
```

`setModalOpened('')`

Please, reuse the `handleCloseModal` function for any closing, not to duplicate the same code

NEEDS CORRECTING

Mark as completed

reviewer

```
      }
    };
    window.addEventListener('keydown', handleEscClose);
    return () => {
      window.removeEventListener('keydown', handleEscClose);
    };
  }, []);
```

You should add an `Escape` listener like this to be able to remove it **correctly** when modals are closed:

```
}, [modalOpened]);  // watch modalOpened here
```

NEEDS CORRECTING

Mark as completed

reviewer

```

  const handleSubmit = (e) => {
    e.preventDefault();

    // Get form values
    const form = e.target;
    const name = form.elements.name.value;
    const link = form.elements.link.value;
    const weather = form.elements.weather.value;

    // Create new clothing item
    const newItem = {
      _id: Date.now().toString(), // Simple unique ID
      name,
      weather,
      link
    };

    // Add item to the state
    setClothingItems([newItem, ...clothingItems]);

    // Close the modal
    setModalOpened('');
```

`setModalOpened('')`

Please, reuse the `handleCloseModal` function for any closing, not to duplicate the same code

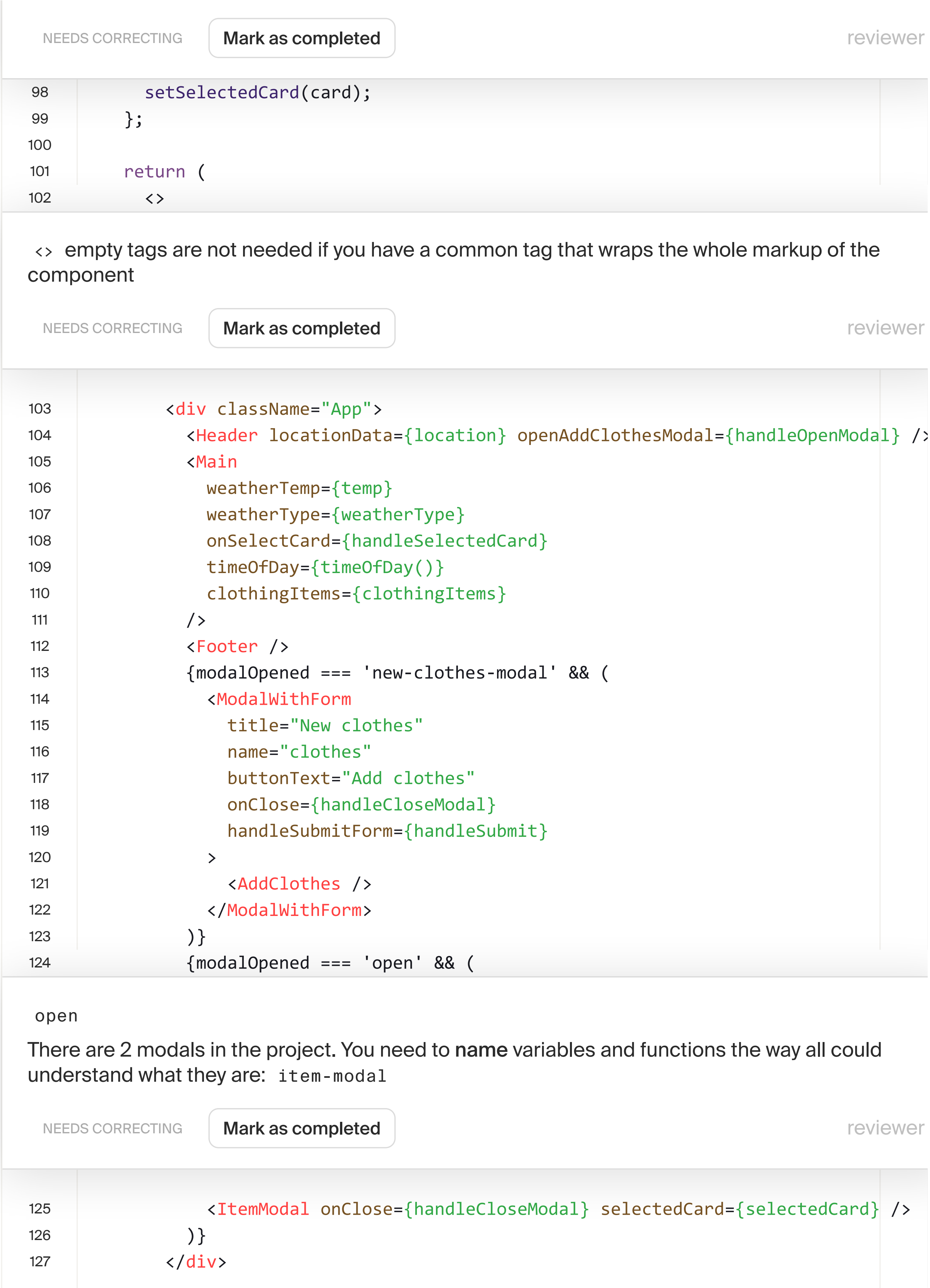
NEEDS CORRECTING Mark as completed reviewer

```
  };

  const handleCloseModal = () => {
    setModalOpened('');
  };

  const handleOpenModal = () => {
    setModalOpened('new-clothes-modal');
  };

  /* Item Card Image Modal functions */
  const handleSelectedCard = (card) => {
    setModalOpened('open');
```

`open`

There are 2 modals in the project. You need to **name** variables and functions the way all could understand what they are: `item-modal`

```
setModalOpened('item-modal');
```



NEEDS CORRECTING

Mark as completed

reviewer

```
      setSelectedCard(card);
    };

    return (
      <>
```

`<>` empty tags are not needed if you have a common tag that wraps the whole markup of the component

NEEDS CORRECTING

Mark as completed

reviewer

```
        <div className="App">
          <Header locationData={location} openAddClothesModal={handleOpenModal} />
          <Main
            weatherTemp={temp}
            weatherType={weatherType}
            onSelectCard={handleSelectedCard}
            timeOfDay={timeOfDay()}
            clothingItems={clothingItems}
          />
          <Footer />
          {modalOpened === 'new-clothes-modal' && (
            <ModalWithForm
              title="New clothes"
              name="clothes"
              buttonText="Add clothes"
              onClose={handleCloseModal}
              handleSubmitForm={handleSubmit}
            >
              <AddClothes />
            </ModalWithForm>
          )}
          {modalOpened === 'open' && (
```

`open`

There are 2 modals in the project. You need to **name** variables and functions the way all could understand what they are: `item-modal`

NEEDS CORRECTING

Mark as completed

reviewer

```
            <ItemModal onClose={handleCloseModal} selectedCard={selectedCard} />
          )}
        </div>
```

```
    </>
  );
}

export default App;
```

Footer 1 1

Footer.css No comments

Footer.jsx 1 1

```
import './Footer.css';

const Footer = () => {
  return (
    <section className="footer">
```

In `Footer` you need to use the `footer` tag, according to the semantics

NEEDS CORRECTING Mark as completed reviewer

```
      <p className="footer__name">Developed by: Sabrina Florence</p>
      <p className="footer__year">{new Date().getFullYear()}</p>
```

Nice year getting

GOOD Mark as completed reviewer

```
    </section>
  );
};

export default Footer;
```

Header No comments

ItemCard 1

css ItemCard.css No comments

jsx ItemCard.jsx 1

```
import './ItemCard.css';

const ItemCard = ({ data, onSelectCard }) => {
  return (
    <ul className="clothing__cards-wrapper">
```

`ul` can't be used for a card because it's not a **list**. It's one `item` of a section, according to the semantics

And `li` will not be needed anymore

You can use `article` here

```
  return (
    <article className="clothing__cards-wrapper">
      <div
        className="clothing__card-items"
        id={`clothing__card-items_${data.name}`}
      >
        <p className="clothing-card-title">{data.name}</p>
        <img
          className="clothing-card-individual"
          id={`clothing-card-individual_${data.name}`}
          alt={data.name}
          src={data.link}
          onClick={() => onSelectCard(data)}
        />
      </div>
    </article>
  );
```

NEEDS CORRECTING Mark as completed reviewer

```
      <li
        className="clothing__card-items"
        id={`clothing__card-items_${data.name}`}
      >
        <p className="clothing-card-title">{data.name}</p>
        <img
          className="clothing-card-individual"
          id={`clothing-card-individual_${data.name}`}
          alt={data.name}
          src={data.link}
          onClick={() => onSelectCard(data)}
        />
      </li>
    </ul>
  );
};

export default ItemCard;
```

✓ ItemModal 1

css ItemModal.css No comments

jsx ItemModal.jsx 1

```
import './ItemModal.css';
import exitButton from '../../images/modalexit.svg';

const ItemModal = ({ onClose, selectedCard }) => {
  const handleModalClick = (e) => {
    if (e.target === e.currentTarget) {
```

Nice overlay handling

GOOD

Mark as completed

reviewer

```
      onClose(e);
    }
  };

  return (
    <section className="modal item__modal" onClick={handleModalClick}>
      <div className="modal__container item__modal-container">
        <button
          className="modal__exit"
          type="button"
          onClick={onClose}
        >
          <img src={exitButton} alt="Close button" />
        </button>
        <div className="item__modal">
          <p className="item__modal-title">{selectedCard.name}</p>
          <img
            className="item__modal-image"
            alt={selectedCard.name}
            src={selectedCard.link}
          />
        </div>
        <p className="item__modal-weather">Weather: {selectedCard.weather}</p>
      </div>
    </section>
  );
};

export default ItemModal;
```

Main 1

Main.css No comments

Main.jsx 1

```
import './Main.css';
import WeatherCard from '../WeatherCard/WeatherCard';
import ItemCard from '../ItemCard/ItemCard';

function Main({ weatherTemp, weatherType, onSelectCard, timeOfDay, clothingItems }) {
  const weatherFilter = () => {
    if (weatherTemp >= 86) {
      return 'hot';
    } else if (weatherTemp >= 66 && weatherTemp <= 85) {
      return 'warm';
    } else if (weatherTemp <= 65) {
      return 'cold';
    }
  };

  const weatherTempFilter = weatherFilter();
  const filterCards = clothingItems.filter((item) => {
    return item.weather === weatherTempFilter;
  });

  return (
    <main className="main">
      <WeatherCard
        day={timeOfDay}
        type={weatherType}
        weatherTemp={weatherTemp}
      />
      <div className="main__title">
        Today is {Math.round(weatherTemp)}°F / You may want to wear:
      </div>
      <section className="clothing">
        {filterCards.map((data) => (
          <ItemCard key={data._id} data={data} onSelectCard={onSelectCard}
/>
```

Good job that you've used `_id` for the `key` rather than the `index`.

Here is why we shouldn't use `index` for that: **click here**

GOOD | Mark as completed | reviewer

```
        ))}
      </section>
    </main>
  );
}

export default Main;
```

ModalWithForm 1 | 1 | 1

ModalWithForm.css No comments

ModalWithForm.jsx 1 | 1 | 1

```
import './ModalWithForm.css';
import exitButton from '../../images/modalexit.svg';

const ModalWithForm = ({
  children,
  title,
  name,
  buttonText,
  onClose,
  handleSubmitForm,
}) => {
  const handleModalClick = (e) => {
    if (e.target === e.currentTarget) {
      onClose(e);
    }
  };

  return (
    <section className={`modal modal_type_${name}`} onClick={handleModalClick}>
      <div className="modal__container">
        <button
          className="modal__exit"
          type="button"
          onClick={onClose}
        >
          <img src={exitButton} alt="Exit button" />
        </button>
        <h2 className="modal__title">{title}</h2>
        <form className="modal__form" id="" onSubmit={handleSubmitForm}>
```

`id=""` is not needed here. It causes jsx validation errors

NEEDS CORRECTING Mark as completed reviewer

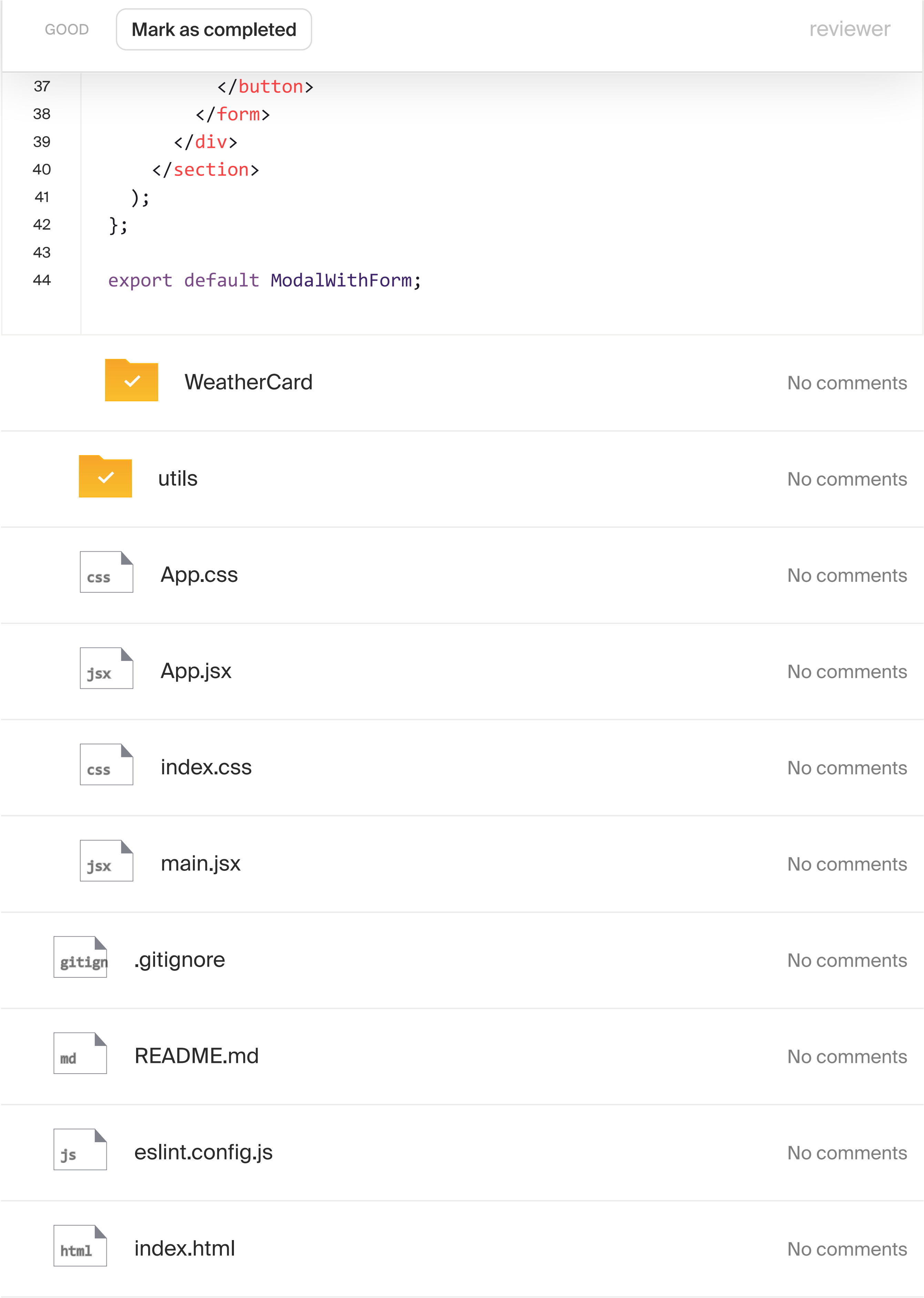
```
          <fieldset className="modal__fieldset">{children}</fieldset>
          <span className="modal__error"></span>
```

You can use **self closing** tags if there are no **children** `<span … />`

COULD BE IMPROVED Mark as completed reviewer

```
          <button
            type="submit"
            className="modal__submit"
          >
            {buttonText}
```

- Good job that the submit button is only in `ModalWithForm`, and its text is controlled



```
          </button>
        </form>
      </div>
    </section>
  );
};

export default ModalWithForm;
```

WeatherCard	No comments
utils	No comments
App.css	No comments
App.jsx	No comments
index.css	No comments
main.jsx	No comments
.gitignore	No comments
README.md	No comments
eslint.config.js	No comments
index.html	No comments

js vite.config.js No comments

roject? 😊 😕

Download code