Build Your Portfolio: 3rd Web Project

Overview

You have come an incredible way over the past 15 weeks on your journey to becoming a full-stack developer. Since the second project, you have been working with Angular, and more recently have been immersed in the world of **React** and **Redux**.

For this second project, you are going to be developing a fully-functional React-based "todo list" web app that utilizes Redux for state management and React Router for navigation--**an extremely common set of tools used in real-world production web apps every day**.

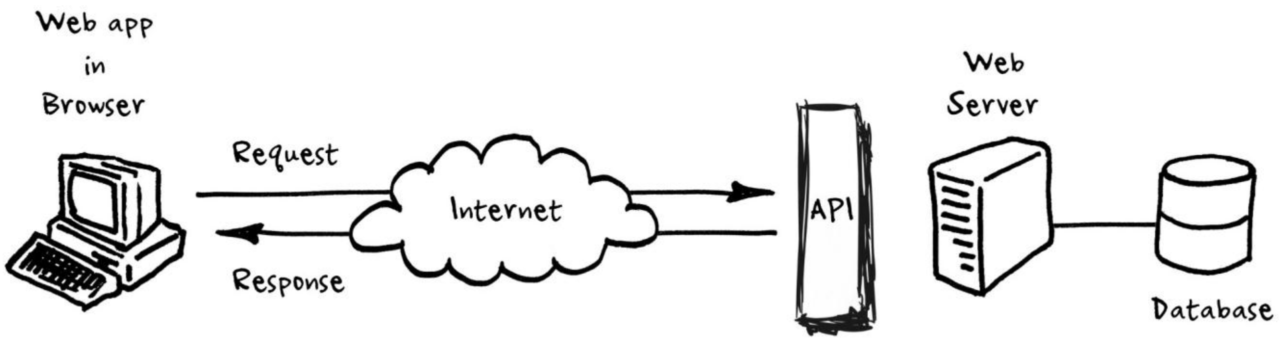
Real-world Context

In practice, React & Redux is a popular combination of tools to use on the frontend of a web app --the part the user sees and interacts with. Most web-apps hinge on the users' ability to create/read/update/delete data, also known as performing "CRUD" operations.

Some examples of CRUD operations offered in popular React web apps are:

- Spotify allows users to create/read/update/delete playlists of songs.
- Facebook allows users to create/read/update/delete posts.
- What'sApp allows users to create/read/update/delete contacts

In production apps, all of that data is typically stored in a database, and the web-app reads and writes data to that database by making calls through an Application Programming Interface (API)-code that lives on the server and is able to communicate directly with the database. A visual representation of this looks something like this:



In the coming weeks, you will be learning how to create a back-end--you will become familiar with working with MySQL databases and writing server-side code. But for now, the focus is going to be on you building out just the front-end of a web app.

Project Context

For the second project, you are going to be building out the frontend portion of your own task management app. Specifically, you will be building out a fully functional todo list feature, as well as a form that uses controlled components.

To provide some context for this project, please take a few minutes to review the two screenshots below beneath "Example--Real World".

The "Example--Real World" screenshots are from the popular task management app "Asana". This full-featured React app allows users the ability to register, login, and save their data--like most modern web-apps do.

While your project will not be as complicated as Asana, it can be extremely helpful to see how similar features to what you will be building are currently being used in real-world production-level apps.

Now, please take a few minutes to compare the screenshots of the Asana web-app to those of the "Example Solution" below. Specifically, focus on the similarities and how the "Example Solution" could potentially be expanded upon to appear more like Asana.

Project Specifics

Similarly to the first project & second projects, you will be evaluated on your ability to meet both the **technical requirements** and the **workflow requirements**. Your project does not have to look identical to this the "Example Solution"...this is to be used for reference and inspiration.

Workflow requirements

The following requirements are related to how you go about building your project

Planning phase - *the following should be completed prior to beginning the development phase and/or touching any code.*

The following can be done on paper or using any number of widely available applications. A free one that may be useful is draw.io. You may find referring back to Module 1 of **Designing a Technical Solution** helpful.

User stories

1) Write out at least three user stories

Wireframes

2) Create wireframes for each view of your app

State and component planning - *you are strongly encouraged to reference Module 2 Tutorial Lab in Redux course for this!*

3) Write out your app's state tree (remember the contact form will manage its own state for the form fields)

4) List out all of the actions (related to redux) that will take place in your app (e.g. ADD_ITEM)

5) Write out a list of the container and presentational components you intend to use in your app

Development phase

1) Create a GitHub repository on Github.com (*before you start coding*)

2) Clone it to your local machine (*before you start coding*)

3) Make frequent commits throughout your development that are descriptive, such as "adds todos reducer" (*throughout development*/coding process)

Technical requirements

The following requirements are related to what your **code** should contain*

1) This should be a React app based off of Create React App

2) Redux should be used to handle the state in your app related directly to the "todos" (e.g. todos list,filter selection etc.)

3) The app should contain at least two views: `/todos` and `/contact`

4) When a user navigates to `/todos` , they should be presented with a view that:

- Display a list of the todo items currently contained in the Redux store
- Displays a form (text input and submit button) that allows users to add a new item to the list
- Offers a way for a task to be marked as "completed" and clearly indicates this status visually (e.g. strike-through effect)
- Offers a way for a task to be removed from the list
- Offers a way to view either
    - all todos
    - completed todos
    - incomplete todos
- When todos are added/updated/marked as complete, these changes should immediately be reflected in your Redux store (visible via Redux DevTools)

5) When a user navigates to `/contact` , they should be presented with a view that:

- Displays a contact form that displays the following fields
    - first name field
    - last name field
    - email field
    - comments field
    - Renders a the form as a controlled component such that after entering text into any of the fields, the form's state has changed (see last screenshot for an example)

6) There should be at least 10 custom CSS rules used throughout the components of your React app

7) You must have a horizontal nav bar at the top of your site

8) You must have at least one example of content side-by-side (e.g. the "new todo" form in the example screenshot)

Deliverables

1) Your user stories

2) A collection of wireframes - one for each view of your app

3) Your state tree and list of actions from the planning phase

4) Your list of container and presentational components from the planning phase

5) Your app source code should be available for viewing in your GitHub repository

6) A `readme.md` file in the root project folder that contains the following information about your project:

- Your name
- Overview/description of the project
- Details on how to use it or what functionality is offered
- Technologies Used (`.html` , `.css`)
- Ideas for future improvement (minimum of 3)

7) Your repository should contain at least 15 commits and should reflect a consistent commit history

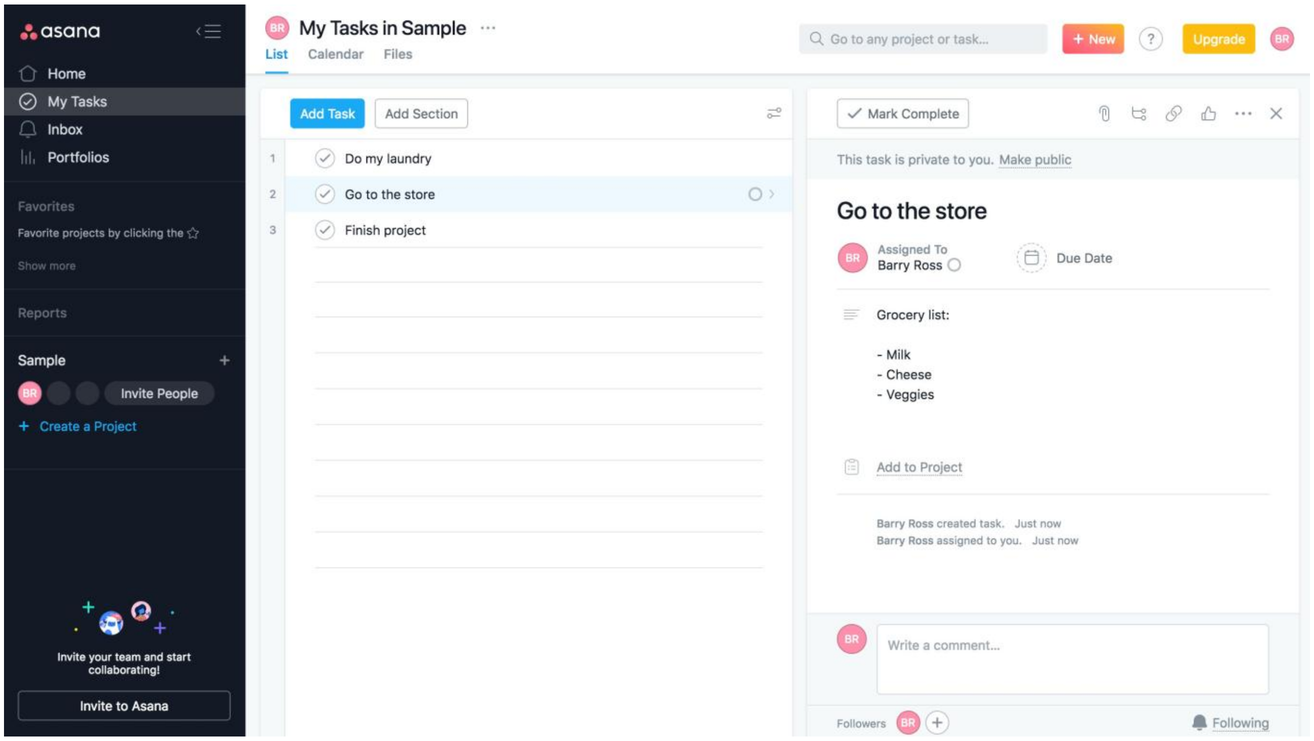
Submission

Submit your GitHub link and hosting link in the communication channel the instructor provided. All deliverables should be included on GitHub and the site should be made publicly available using a hosting service.

Example - real world

Asana - Todos view

Here is an example of how a "todo" list may appear in a real-world React app.



1 Do my laundry
2 Go to the store
3 Finish project

This task is private to you. Make public

Go to the store

Assigned To Barry Ross
Due Date

Grocery list:

- Milk
- Cheese
- Veggies

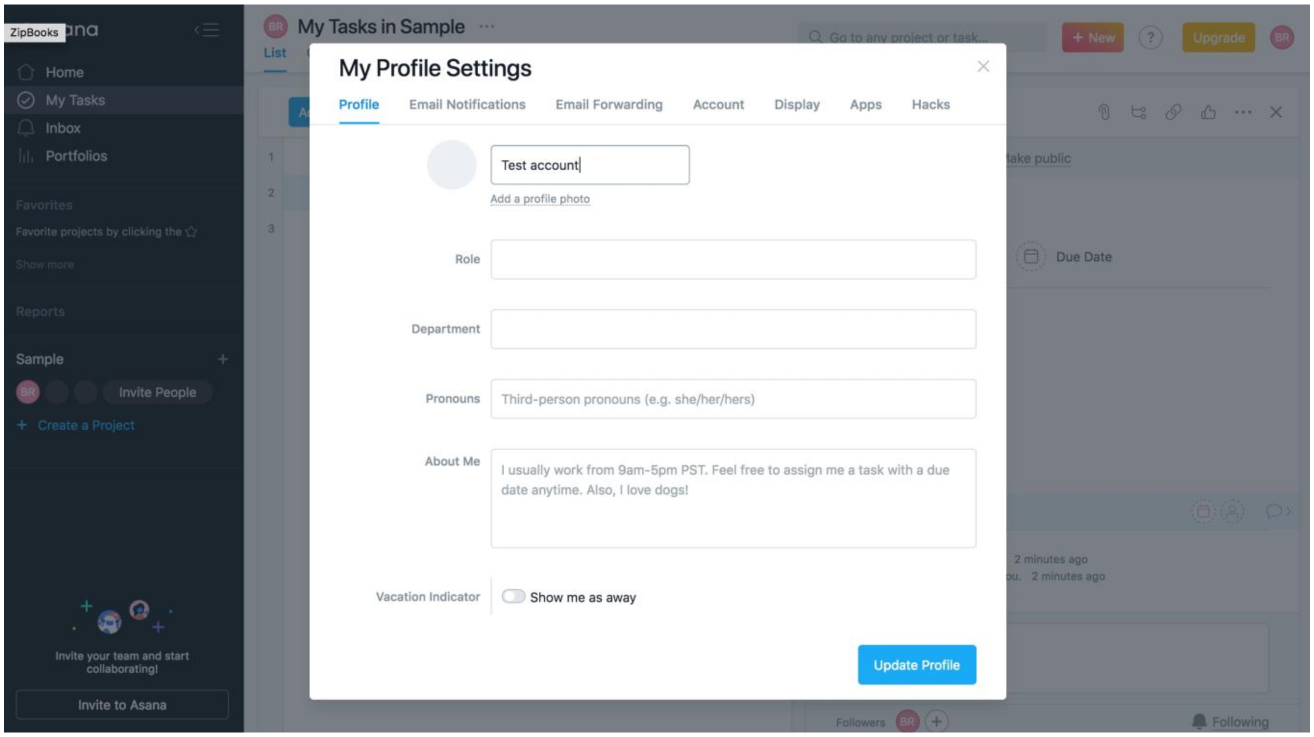
Add to Project

Barry Ross created task. Just now
Barry Ross assigned to you. Just now

Write a comment...

Asana - Form view

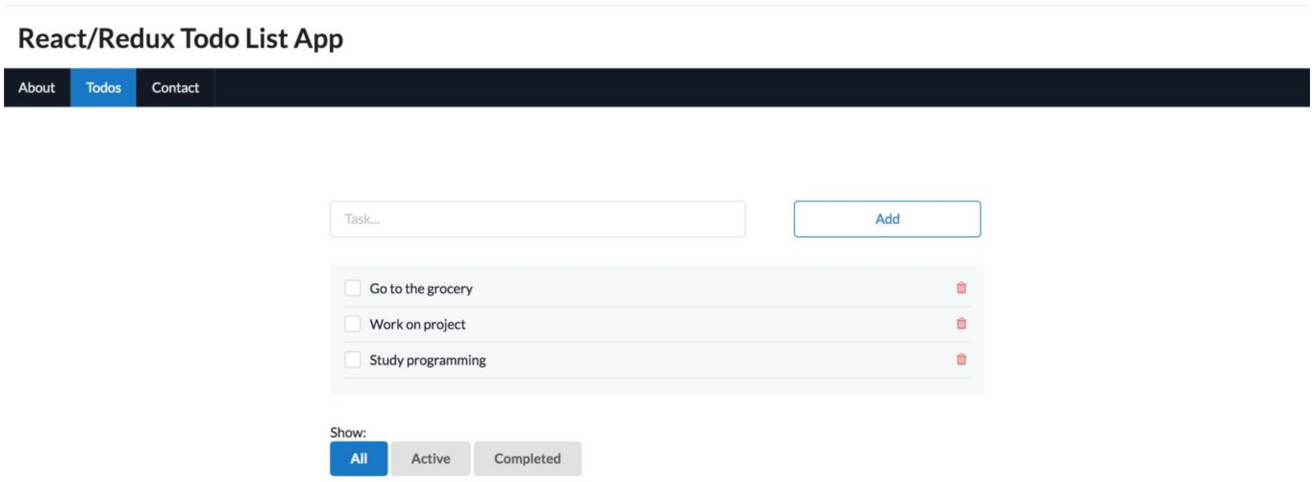
Here is an example of how a form may appear in the context of a real-world React app.



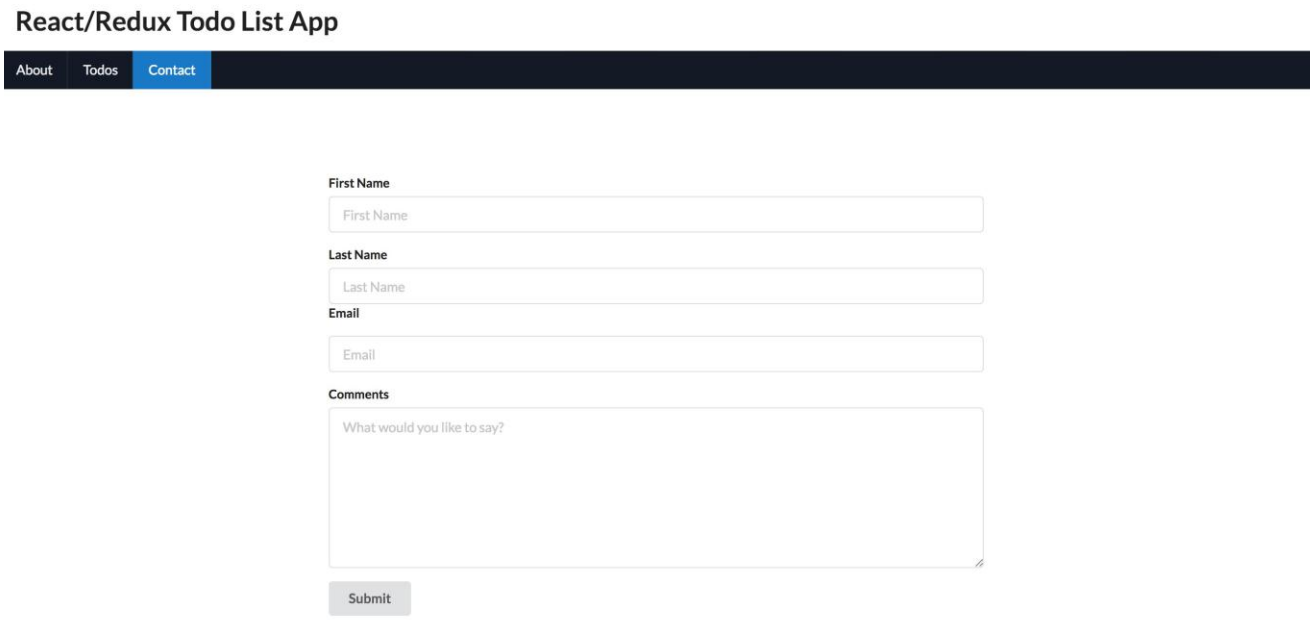
	Test account
Role	
Department	
Pronouns	Third-person pronouns (e.g. she/her/hers)
About Me	I usually work from 9am-5pm PST. Feel free to assign me a task with a due date anytime. Also, I love dogs!
Vacation Indicator	Show me as away

Example - Sample Solution

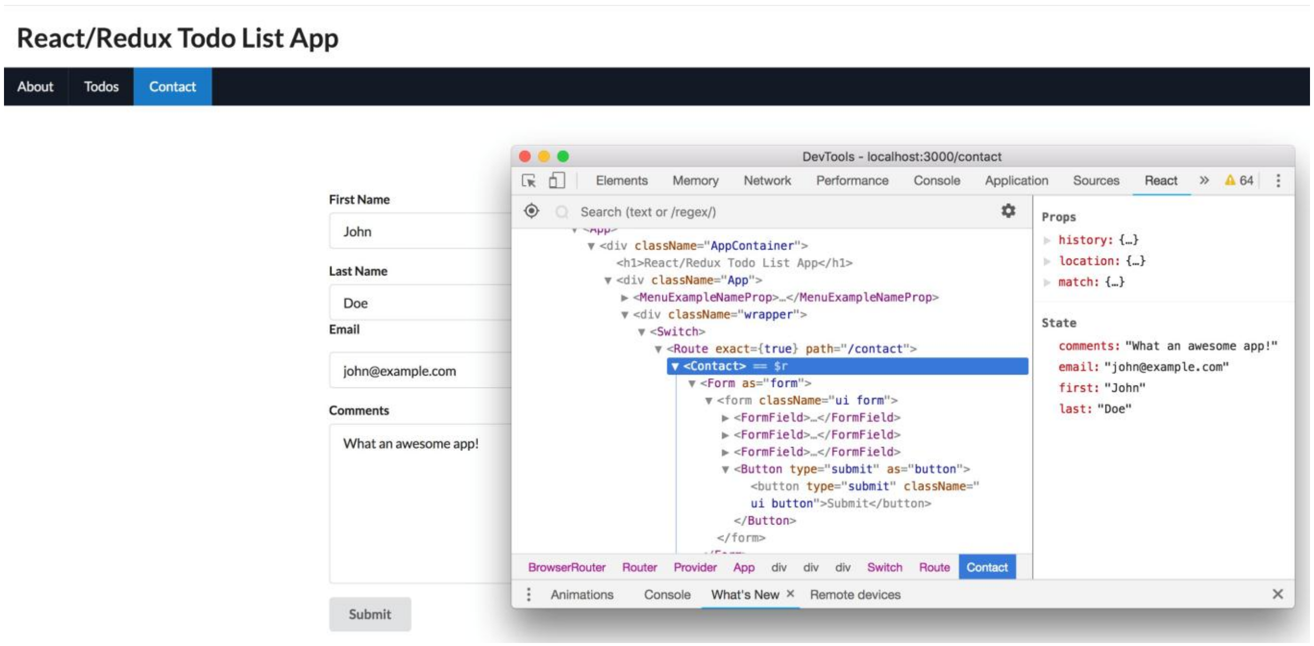
Todos view



Contact form



Contact form's self-managed state



```
▼ <App>
  ▼ <div className="AppContainer">
      <h1>React/Redux Todo List App</h1>
    ▼ <div className="App">
      ▶ <MenuExampleNameProp>…</MenuExampleNameProp>
      ▼ <div className="wrapper">
        ▼ <Switch>
          ▼ <Route exact={true} path="/contact">
            ▼ <Contact> == $r
              ▼ <Form as="form">
                ▼ <form className="ui form">
                  ▶ <FormField>…</FormField>
                  ▶ <FormField>…</FormField>
                  ▶ <FormField>…</FormField>
                  ▼ <Button type="submit" as="button">
                      <button type="submit" className="
                      ui button">Submit</button>
                    </Button>
                  </form>
                </Form>
```

```
Props
▶ history: {…}
▶ location: {…}
▶ match: {…}

State
  comments: "What an awesome app!"
  email: "john@example.com"
  first: "John"
  last: "Doe"
```