

# DTEK Online Store – Maintenance

<https://rob-das-react.azurewebsites.net/>

2023-Nov-01

Rob Das

DTEK Consulting Services Ltd.

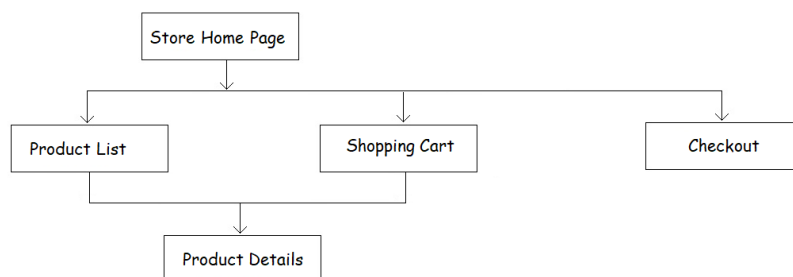
This document provides maintenance and support information for the DTEK Online Store web application. The target audience is: operational support.

The source code for the DTEK Online Store application is your primary resource, which together with this document, has everything you need to provide maintenance and support. For information on how the application was designed, there is a separate document called *DTEK Online Store – Design*. For information on how the application is used, there is a separate document called *DTEK Online Store – User Manual*.

## Component Hierarchy Overview

Following the standard convention for React development, as scaffolded by the *create-react-app* tool, `index.js` links the JavaScript used to build the hierarchy of React components to the `index.html` file. Also, following the standard convention, the highest-level component is *App*, which is bound to the root element in the `index.html` file. *App* contains the *Store* component, together with a user notification utility, `react-toastify`, and a function called *notify* that that uses `react-toastify`, which is passed into the *Store* component.

The *Store* component, serves as the *Store* home page. *Store* serves as the application's landing page; the SPA navigation using `React-Router`; and the containing component for application wide state. Housed within the *Store*'s routing component are four application pages, defined by React components named `ProductList`, `Cart`, `Checkout`, and `ProductDetails`. A navigation bar provides the user with the ability to freely navigate between `ProductList`, `Cart`, and `Checkout`. `ProductDetails` can only be accessed by selecting a specific product, either from `ProductList`, or from `Cart`. This page layout can be summarized with the following site map.



## ProductList

The `ProductList` component, which is shown on the website with the user-friendly name “Stuff You Need”, provides the user with a way to browse all products available for purchase. All products are shown on `ProductList` by means of a collection of `ProductSummary` components. Each `ProductSummary` provides a link to the corresponding `ProductDetails` component.

- Store
  - ProductList
    - ProductSummary → ProductDetails
    - ProductSummary → ProductDetails
    - ...

## Cart

The Cart component, which is shown on the website with the user-friendly name “Shopping Cart”, provides the user with a way to see everything they’ve added to their shopping cart so far. Cart contains a ChosenProductList component, and a Totals component.

The ChosenProductList component contains a collection of ChosenProduct components, which in turn displays information about each product in the cart, also provides a link to the corresponding ProductDetails component.

The Totals component shows a subtotal for all products in the shopping cart, the calculated sales tax, and the grand total.

- Store
  - Cart
    - ChosenProductList
      - ChosenProduct → ProductDetails
      - ChosenProduct → ProductDetails
      - ...
    - Totals

## Checkout

The Checkout component, which is shown on the website simply as “Checkout”, provides the user with a way to finish their shopping by making a final purchase for the entire contents of the shopping cart. The user initiates the final purchase by pressing a purchase button, which is disabled until the user has provided all the required information.

The Checkout page contains a Totals component, and PurchaseInfo component. The PurchaseInfo component provides the user with a way to provide the required information.

The PurchaseInfo component contains Payment and Shipping component, which provide fieldsets for data entry of payment and shipping information, respectively.

Checkout is the containing component for Payment and Shipping state, so that the user can freely navigate back and forth between Checkout, Cart, and ProductList, without having to re-enter Payment and Shipping information.

- Store
  - Checkout
    - Totals
    - PurchaseInfo
      - Payment
      - Shipping

## Third Party Packages

DTEK Online Store makes use of the following third-party components:

- Bootstrap Alerts
- NPM react-toastify
- npm react-confirm-alert

### Bootstrap Alerts

Since JavaScript `window.alert()` is only intended for debugging, a custom build alerting mechanism was needed. Bootstrap alerts were used instead to meet this need. However Bootstrap styling overrode to custom CSS styling that was developed for this website, so the Bootstrap styling that was specific to the Bootstrap alerts was reverse engineered using the browser inspect tool, and then added to the custom stylesheet, and then the Bootstrap CSS was removed.

Only two alerts are needed at the moment, a *missing information* alert and a *payment not accepted* alert. Both are used in the *Checkout* page, so they are declared in the *Checkout* page and wrapped in logic to determine whether they are to be shown. For example, the missing information alert is coded as follows:

```
import Alert from "react-bootstrap/Alert";

// other checkout related code here...

{
  (showMissingInfoAlert && missingInfo())
  ?
  <>
    <div className='center-element oops-alert checkout-width'>
      <Alert variant="primary" onClose={() => setShowMissingInfoAlert(false)} dismissible>
        <Alert.Heading>Missing Information</Alert.Heading>
        <p>
          All payment and shipping information is needed to make your purchase.
        </p>
      </Alert>
    </div>
  </>
  :
  <></>
}
```

If alerts are needed elsewhere later on, the alerts code can be refactored and moved to a common location higher up in the component hierarchy.

## NPM react-toastify

Some actions at runtime aren't clearly visible to the user. For example, if the user clicks the button to add a product to the shopping cart, and the user has scrolled down so that the header isn't visible, then the user gets minimal feedback that the button was clicked (the button's color changes on click).

To address this, a popup messaging mechanism was created using the react-toastify npm package. Toastify popups are by design always visible no matter where the user has scrolled to on the page. The *ToastContainer* component, which provides the display of the popup message, is at the top level in the react component hierarchy (App.cs). A *notify* function is declared that takes a text message as a parameter and displays it. The *notify* function is passed as a prop, into the *Store* react component, and then from *Store*, it is passed into any react component that needs to use it. Here is the code for App.js

```
import Store from "../components/Store";
import { ToastContainer, toast } from "react-toastify";
import { Zoom } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

function App() {
  const notify = (message) =>
    toast.info(
      message,
      {
        position: "top-center",
        autoClose: 1000,
        theme: "dark",
        hideProgressBar: true
      }
    );

  return (
    <div className="App">
      <Store notify={notify} />
      <ToastContainer newestOnTop transition={Zoom} />
    </div>
  );
}

export default App;
```

## NPM react-confirm-alert

Some user actions such as deleting things may cause irreversible outcomes, so the user has to confirm their intent before proceeding. This is commonly done with an “are you sure”. OK, cancel, dialog.

To address this, a confirmation prompting mechanism was created using the react-confirm-alert npm package. To install from the project root using npm CLI, enter the following command:

```
npm install react-confirm-alert
```

The react-confirm-alert produces a modal popup dialog with buttons and corresponding callbacks. For example, the shopping cart offers the user a start again button.

```
import { useNavigate } from "react-router-dom";

// ... more code here ...

import { confirmAlert } from "react-confirm-alert";
import 'react-confirm-alert/src/react-confirm-alert.css';

function Cart({ cartContents, onAdd, onRemove, clearCart, headingClass, onNav, cartTotals }) {
  const checkoutPath = '/checkout';
  const startPath = '/';
  const navigate = useNavigate();
  const gotoCheckout = () => {
    onNav(checkoutPath);
    navigate(checkoutPath);
  }
  const gotoStart = () => {
    confirmAlert({
      title: 'Confirm',
      message: 'Are you sure you want to start again? You will abandon your stuff, your total will be set to zero, and you will get a new empty shopping cart.',
      buttons: [
        {
          label: 'OK',
          onClick: () => {
            clearCart();
            onNav(startPath);
            navigate(startPath);
          }
        },
        {
          label: 'Cancel',
          onClick: () => {
            ;
          }
        }
      ]
    });
  }
  return (
    <>

// ... more code here ...

    <div className="center-element shopping-cart-width text-right">
      <button onClick={() => gotoStart()}>start again</button>
      <button onClick={() => gotoCheckout()}>checkout</button>
    </div>
  </>



```

```
    );  
  }  
  export default Cart;
```

This code produces the following dialog:

The screenshot displays the DTEK Online Store interface. At the top, the store logo and name "DTEK Online Store" are shown, along with the tagline "late-night shopping, made easy" and a shopping cart icon. The current total is \$1806.42. Below this, there are three buttons: "Get Stuff", "Shopping Cart", and "Checkout".

The "Shopping Cart" section is visible, showing a list of items with their respective costs. A confirmation dialog is overlaid on the cart, asking if the user is sure they want to start again, warning that the current cart will be abandoned and replaced with an empty one. The dialog has "OK" and "Cancel" buttons.

your stuff	remove / add	qty	cost
cucumber	 / 	1	\$9.71
new shoes			\$1650.75
cuppa			\$39.99
phone			\$19.95
subtotal			\$1720.40
tax			\$86.02
<b>Total</b>			<b>\$1806.42</b>

start again      checkout

**DTEK Online Store v1.0.24** — a React development proof-of-concept. This eCommerce site is intended to provide a recreational shopping experience. Copyright © Robi Indra Das, 2009-2023. All rights reserved. DTEK Consulting Services Ltd. is a Canadian company.

Stuff are images of real things. They're based on royalty free, public domain pics from [Pexels](#). This version of DTEK Online Store is intended to simulate a recreational shopping experience - it does not process or save any info, payments are not accepted, and no merchandise is shipped.

## Standard Maintenance and Support Activities

This section describes typical activities that come up during coding and deployment.

### Create New React App

React apps are created with npx. To start a new React project:

```
npx create-react-app my-app
```

### Building the React App

The source code is in an online repo in Azure DevOps at

[https://dev.azure.com/dtek-consulting-services/\\_git/online-store](https://dev.azure.com/dtek-consulting-services/_git/online-store)

To get the code:

```
e:  
cd /repo/online-store  
git pull
```

This assumes your development computer is set up with the Git repo rooted at E:\repo\online-store. Adjust accordingly, based on how your computer's directory structure is setup.

To build and run the React app locally during development and local testing:

```
npm run start
```

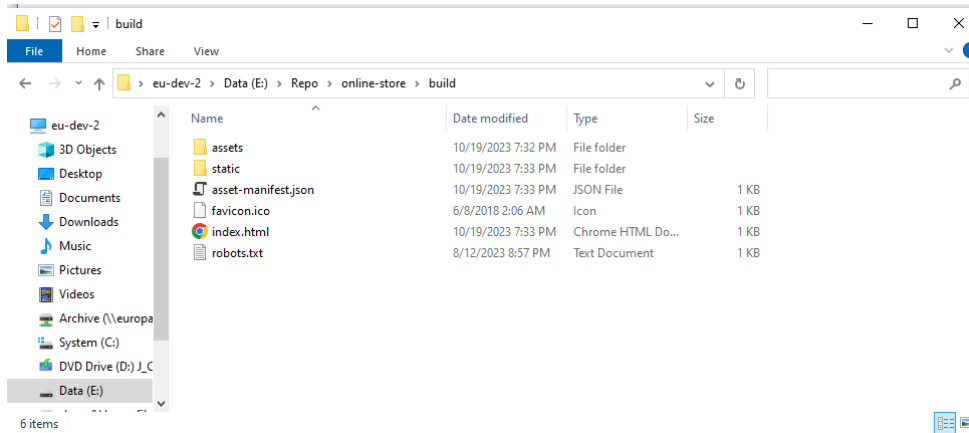
To produce a build folder, which can be sent by ftp to the Azure web app:

```
npm run build
```



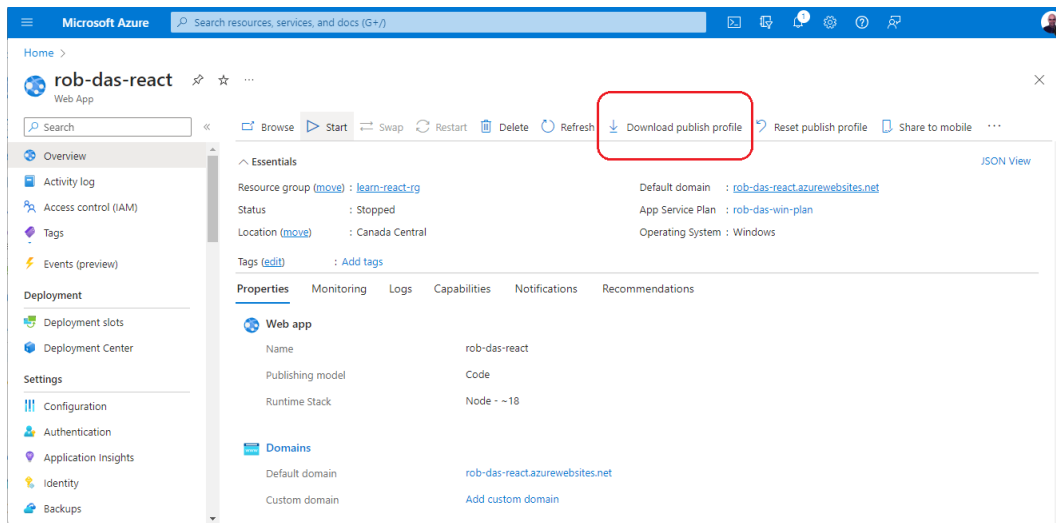
## Hosting the React App on Azure

After the production build is completed successfully, as described in the previous section, a build folder will be created under the project root.



The contents of the build folder can then be FTP'd to the Azure web app.

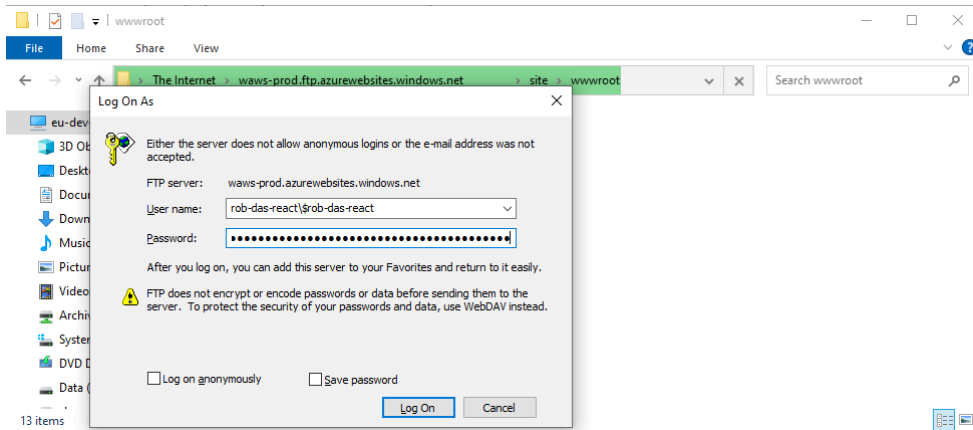
From the Azure Portal, navigate to the App Service, in order to download the publish profile. Verify that the Status is *Stopped*. Click on the “Download publish profile” link.



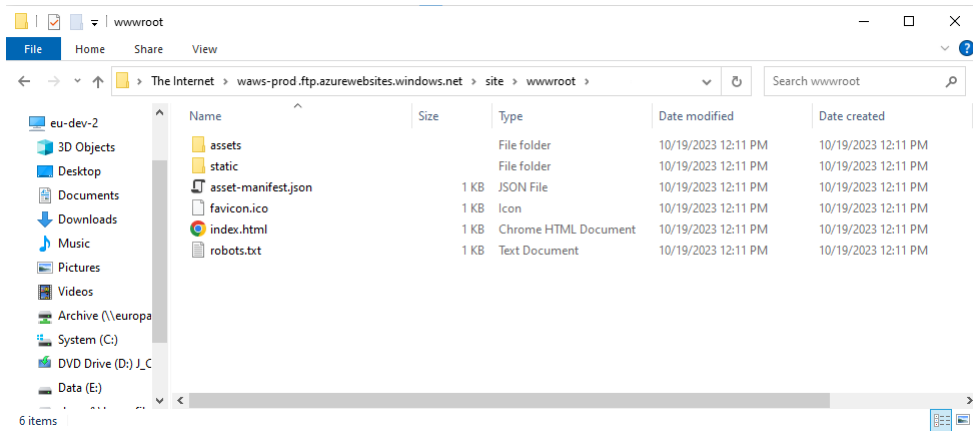
The file that is downloaded will contain a publish profile for FTP. Make note of the *publishURL*, *username*, and *userPWD* values.

Described here is the use of Windows File Explorer as a GUI tool for FTP. You may of course prefer to use another tool, or the FTP command line. But be aware that you will have to recursively copy all subfolders, which isn't handled straightforwardly by the FTP command line.

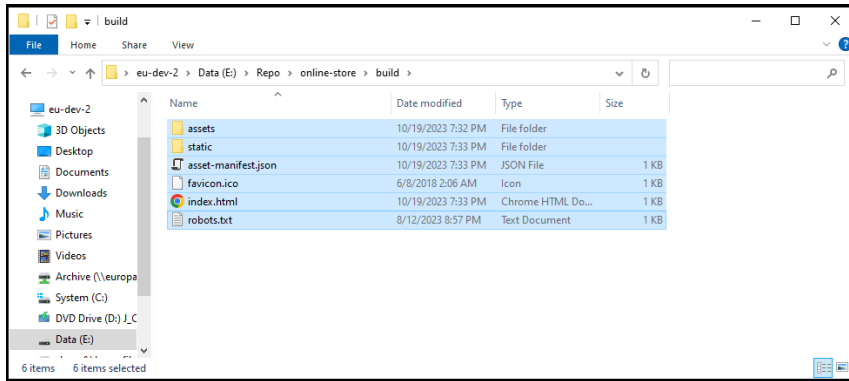
On the local development computer, open another file explorer, and paste the value of the publishURL in the address bar and press enter. You will be prompted for username and password. Enter the values from username and userPWD.



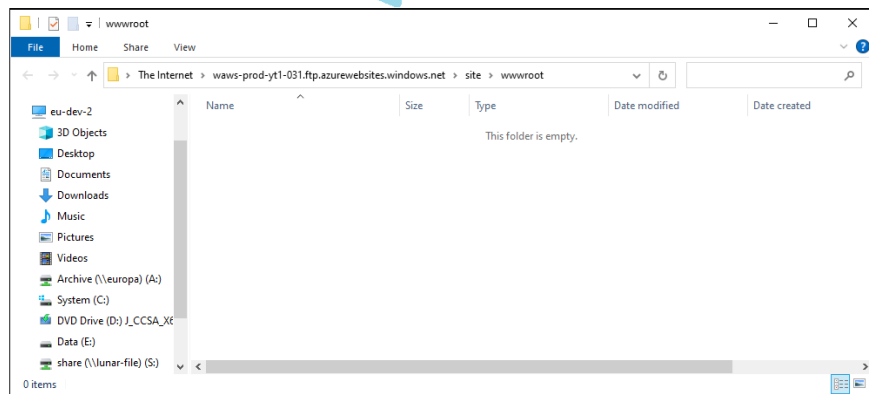
After you click the Log On button, you will see the remote `wwwroot` directory. If it appears empty at first, you can navigate up one level to the `site` directory, and then navigate back down to the `wwwroot` directory and you should now see the current contents.



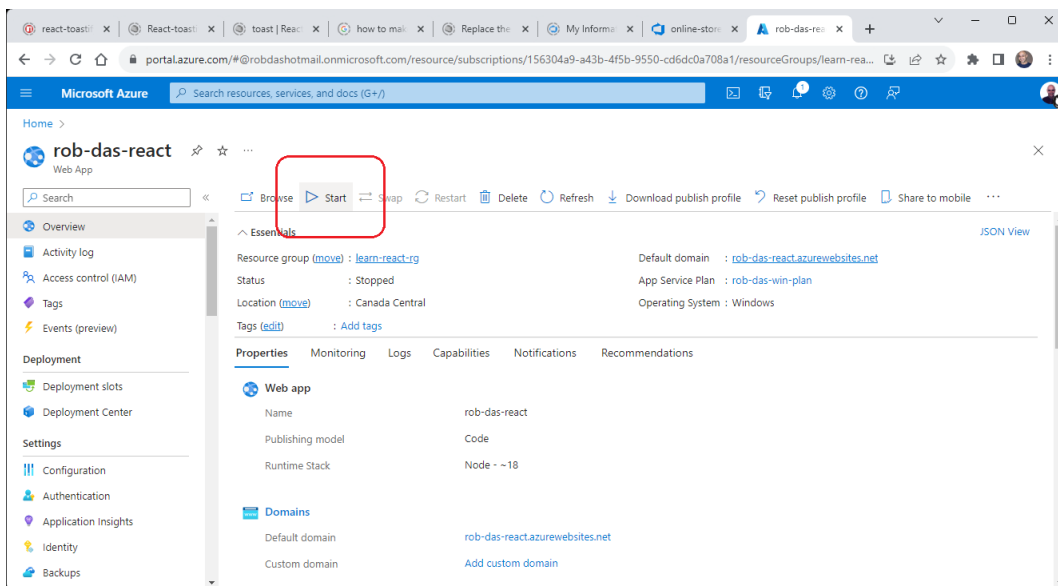
To avoid accidentally leaving any old files, delete everything in the remote `wwwroot` directory before copying over the new stuff. You can now drag-and-drop the new files from the `build` folder in the local computer's file system, into the remote `wwwroot` ftp directory.



DRAG-AND-DROP



When all files and subfolders have been copied, return to the App Service in the Azure Portal. Click the start button to start the web app.



After successfully starting, the Status will change from *Stopped* to *Running* and the web app will now be hosting the latest build that was copied from the local computer.

As a quick smoke-test, in the Azure Portal, click the Browse button and the latest version, now running online, will pop up in a new browser tab.



## Trouble Shooting

### Symptom

When building from the command line using

```
npm run start
```

the following error occurs

```
Failed to compile.
```

```
Module not found: Error: Can't resolve 'react-confirm-alert' in  
'E:\Repo\online-store\src\components'
```

```
ERROR in ./src/components/Cart.js 8:0-51
```

```
Module not found: Error: Can't resolve 'react-confirm-alert' in  
'E:\Repo\online-store\src\components'
```

```
ERROR in ./src/components/Cart.js 9:0-57
```

```
Module not found: Error: Can't resolve 'react-confirm-alert/src/react-confirm-  
alert.css' in 'E:\Repo\online-store\src\components'
```

```
webpack compiled with 2 errors
```

### Cause

The react-confirm-alert package is not installed in the node\_modules folder.

### Resolution

Rerun the install command to install the react-confirm-alert package.

```
npm install react-confirm-alert
```

## REFERENCES

- DTEK Online Store – Documentation  
<https://rob-das-win.azurewebsites.net/docs/OnlineStoreUserManual.pdf>  
<https://rob-das-win.azurewebsites.net/docs/OnlineStoreDesign.pdf>  
<https://rob-das-win.azurewebsites.net/docs/OnlineStoreMaintenance.pdf>
- React Bootstrap Alerts  
<https://react-bootstrap.netlify.app/docs/components/alerts/>
- React-confirm-alert NPM package  
<https://www.npmjs.com/package/react-confirm-alert>  
<https://github.com/GA-MO/react-confirm-alert/blob/master/Document-v2.md>
- React-Toastify NPM package  
<https://www.npmjs.com/package/react-toastify>  
<https://fkhadra.github.io/react-toastify/introduction>  
<https://fkhadra.github.io/react-toastify/api/toast>  
<https://youtu.be/7CcAKre3Ifc?si=l6TuPU1kJU1FUW6x>  
<https://youtu.be/NHlExjLI-d0?si=9IBL7OfYxiFNjWC5>  
<https://stackoverflow.com/questions/53179075/with-useeffect-how-can-i-skip-applying-an-effect-upon-the-initial-render>  
[https://www.w3schools.com/react/react\\_useeffect.asp](https://www.w3schools.com/react/react_useeffect.asp)
- Create New React App  
<https://docs.npmjs.com/cli/v8/commands/npx>  
<https://create-react-app.dev>  
<https://docs.npmjs.com/cli/v8/commands/npx>  
[https://code.visualstudio.com/docs/nodejs/reactjs-tutorial#\\_welcome-to-react](https://code.visualstudio.com/docs/nodejs/reactjs-tutorial#_welcome-to-react)
- Source Code Repo  
[https://dev.azure.com/dtek-consulting-services/\\_git/online-store](https://dev.azure.com/dtek-consulting-services/_git/online-store)
- Hosting the React App on Azure  
<https://enlear.academy/how-to-deploy-react-app-on-azure-dbf36111cd54>