Refer to the Canvas Assignment for screenshots, video, and updates.
This Nov 20, 2023 document will not be remade when updates are made.

Assignment 8 ~ Portfolio Frontend

Introduction

In this Portfolio assignment, you'll modify your existing static website, so it runs in the React framework. This will involve reconfiguring the folder and file structure, as well as moving content into new page and component files. In addition, you will develop the frontend user interface for interacting with your backend MongoDB collection to perform CRUD operations.

You may opt to update the Gallery, Order, and Contact pages, for extra credit.

Note: The material you need to know to complete this assignment is covered in these Explorations:

- Module 1, 2 (Semantic HTML)
- Module 3 (Images and Icons)
- Module 4 (CSS)
- Module 5, 6, 7 (JavaScript, Express)
- Module 8 (MongoDB, Mongoose, REST)
- Module 9 (React)

Be sure to periodically review the Assignment 8 Tips thread in Ed discussion (pinned at the top).

Learning Outcomes

- Convert the existing site/app to the React framework.
- Move existing styles into the React file structure.
- Define a component to render global navigation.
- Define a global page layout structure.
- Develop pages and components that perform CRUD on the existing backend MongoDB app.
- Develop a new home page to explain the website's technologies.
- Convert previous home page to a new page of Topics.
- (Optional) Convert the Gallery to use JSON objects with the React Image Gallery package.
- (Optional) Convert the Products page to use row components with icons and events that map() to a table.

- (Optional) Convert the Contact page to send an email message using a new package.

Tools

The React framework starter code (Movies example), React icons, and React Router DOM dependencies will be installed via `npm`. Continue to use Node.js v18. Incorporate the backend app into the final Portfolio folder. Use LanguageTool browser plugin to check grammar, spelling, and punctuation on all pages and responses.

Instructions

This Assignment will require use of your existing website files: index.html, main.css, background image, and favicons. In addition, the backend MongoDB app will be combined to perform CRUD. A screenshot of your Collection user interface page will show the results of your design work.

Optionally, the gallery.html (and the folder of optimized image files), order.html (and related products.js file), and contact.html pages can be incorporated for extra credit.

Use these tabs to view file-specific instructions.

Folder and File Structure

1. **In your existing M9 folder, make a new folder called *portfolio.***
   This folder must hold the *backend* folder made previously as well as the new *frontend* folder, explained below:
2. **The portfolio > backend folder** will have its own package.json, node_modules, and .env files.
   1. In VS Code, right-click on the backend folder and choose **Open in Integrated Terminal**. This will allow you to operate the backend separately from the frontend. Npm install and npm start.
   2. The backend of your site will now run in the browser at this address (but nothing will display there): `http://localhost:3000/`.
3. **To make the frontend:**
   1. Download a new copy of the movies-frontend-starter-1123-1.zip into the **portfolio folder.**
      1. Shorten the folder name to *frontend.*
   2. The frontend folder has its own package.json and .env files. In addition, you will install node_modules in this folder.
      1. In VS Code, right-click on the new frontend folder and choose **Open in Integrated Terminal**. This will allow you to operate the frontend separately from the backend. In the console/terminal, type npm install and npm start.
   3. The frontend of your site will now display in the browser on `http://localhost:8000/`.
      The backend will be working on `http://localhost:3000/`.

Structure of folders and files

Set up the new Assignment for success:

1. In your existing Module 8 folder, make a copy of the **Movies Backend Starter Code 7-23**
2. Download Movies Backend Starter Code 7-23
3. folder and rename it ***a7-username-backend***.
    1. Replace -*username*- with *your* ONID username.
    2. This folder holds the model.mjs, controller.mjs, package.json, and .env files needed to run the app.

    3. Change the model.mjs and controller.mjs filenames to include your topic. For example: exercises-model.mjs.

    4. The package.json file includes the necessary dependencies and start script (`dotenv`, `express`, `mongoose`, `nodemon`, and `rest`).
        1. Update the file with your topic name, description, and your name as the author.
        2. Update the "main" and "start" scripts to reflect your new controller.mjs filename.

    5. To the .env file, add your MongoDB connection string and port number:

```
MONGODB_CONNECT_STRING='mongodb+srv://YourAccountName:YourAccountPassword@cluster0.ybrfb.mongodb.net/test'
```

        1. `PORT=3000`
           Replace the account and username in the string to match your Atlas Cluster. Use the connection string provided by your account; it may not match this string exactly.

           OR, if you have MongoDB working in your local hard drive, you can use this string:
           `mongodb://localhost:27017/`
4. In VS Code, **File** > **Open in Integrated Terminal**. Type `npm install` and `npm start`.
    1. **Note:** when we add this backend app to the next Module's frontend, you'll open each of the folders in their own integrated terminal.

Update the React index.html and Favicons

Locate the frontend's public > index.html file, and:

1. Update the `<title>` tag with your **name**. You may also add a descriptive phrase, such as *Web Dev Portfolio*.
2. Update the `<meta>` **description** to describe your app's purpose and technologies used.
3. Replace the existing **favicon** `<link>` tags, so they reference *your* set of favicons.

While you are here, notice the `<div id="root"></div>` tag. This is where the content of each of your pages will be routed to, so they can render in the index of the app. Do not alter this division; leave it as is.

Important: Do not copy any other existing content or tags into this file! This is a React index page, not a static HTML home page.

Add your existing **favicon** image files into the root of the public folder. They should replace the existing React favicon files.

Update the App.css file.

Locate the App.css file in the root and:

1. **Delete** all existing styles in the App.css file.
2. **Copy** styles from your existing main.css and **paste** them into App.css.
3. To ensure your background image renders, make an images folder in the src folder. Copy just your optimized background image into the new folder.

Create Global Navigation.

Links to each page *from each page* is easily accomplished in React by using the built-in `<Link>` component. It replaces the `<a>` tag in our code but renders in the browser as `<a>` tags.

When a block of `<Link>`s are imported into the App.js file, they become the *global* navigation for the site/app. This single place to add navigation to new pages saves time and reduces mistakes.

Create a component file for a block of page links:

1. In the src > components folder, a Navigation.js already exists. Open it.
2. In Navigation.js, you'll see that React and Link features have already been imported:
   `import { Link } from 'react-router-dom';`
3. Change the `Menu()` function name to match the file name. You may change the file name as well.
4. Update the list of `<Link/>`s to reflect each page that you will have in your website (excluding Create, Update, and Delete; they do not need to be in the global navigation). The `to=""` path for each `<Link/>` will refer back a directory with ../ and reference the `<Route>` path as defined in the App.js file (which is created in the next tab.) The only exception is the Home page Route; it must not refer to a path; just itself, like this: `<Link to="/">Home</Link>`.
5. Export the function name you edited in step 3.
6. **Do not include links to any pages that are not working.** If you complete extra credit pages, then add only those pages to the navigation. **We don't have time to test all pages that are not working.**

Update the App.js file.

Locate the App.js file in the src folder and:

Notice that React, BrowserRouter, Routes, Route, and App.css have already been imported:
`import React from 'react';`

`import { BrowserRouter, Routes, Route } from 'react-router-dom';`

1.
2. Notice that the Navigation.js component has been imported. Change the function and filename to match the name you gave them earlier (from instructions in the previous tab).

Each time you complete a new page for this app, you'll import it in this file (just under the previous imports). For the time being, you can **comment them out** (to keep React from crashing) and **uncomment** them one at a time, as you finish creating each.

Import them using this method: `import functionName from './folder/file.js'` for each:

1. ./data/products.js
2. ./pages/HomePage.js
3. ./pages/TopicPage.js
4. ./pages/GalleryPage.js (Extra Credit)
5. ./pages/OrderPage.js (Extra Credit)
6. ./pages/ContactPage.js (Extra Credit)

Modify the existing `App()` function to include:

1. If any **page layout tags** are missing, then add them in the appropriate places.
2. If the **favicon** in your <header> is not rendering, then update the path to reference it from the public root. Feel free to use a React icon or other .png or .svg file.
3. Add or update the `<Route>` tags for each page:
   `<Route path="/page" element={<PageName params />} />`
   Match the path with the page name (but only use a slash "/" for the HomePage path).

   These pages will need params added to their component names:
    1. LogPage
    2. EditPage
    3. OrderPage (Extra Credit)

4. Update the copyright statement to include your name.

Convert the Home page and Create the Topics page.

The old home page will become the new Topics page and you'll write a new Home page.

1. In the src > pages folder, paste your existing A6 index.html file and rename it TopicsPage.js (or something similar).
    1. Remove all the tags and content from *above* the `<h2>` page name.

Replace that old code with the React import and function for `TopicsPage()` and begin the `return`.

```
function TopicsPage() {

    return (

        <>
```

    2.
    3. Remove all the tags and content from *below* the ending `</article>` tag.

Replace that old code with the end of the return and function, then export the default `TopicsPage` function.

```
</>

    );

}

export default HomePage;
```

2.
3. Now that this page exists, you can import it into the App.js file.
4. In the src > pages folder, add a new file called **HomePage.js** (or something similar). Add the necessary imports and define a function for `HomePage()` that includes the following in the `return()`:
1. `<h2>` with page name.
2. `<article>` with a `<p>` that explains your career goals.
3. `<p>` and a list (either `<dl>, <ul>,` or `<ol>`) that **explains** the technologies used in the website. Incorporate the terminology you learned in the course.
4. You may also embellish the page with icons and other optimized images to improve user experience.

Collection User Interface

The functionality of the Collection page is to retrieve your MongoDB collection's documents, as well as provide icons for adding, editing, and deleting documents. Because you started this project using the Movies frontend starter code, the frontend files are present and ready for you to modify, based on your schema, models, and controllers from the backend. You have the choice of using a form or table for the Add and Edit features.

1. In the src > pages folder, **rename** the existing **MoviesPage.js** to fit the topic of your collection. From here on out, these instructions will refer to it as the Log page, since everyone will have a different topic, function, and file name.

2. In the LogPage.js file, notice the imports for `React`, `useState`, `useEffect`, and `useNavigate`. Also notice the import for the `MovieList` component.

3. **Change the filenames, function() names, and import values** (and all other references to movies) for the following files, so they correlate with your topic:
    1. MovieList.js
    2. Movies.js
    3. AddMoviePageForm.js*
    4. AddMoviePageTable.js
    5. EditMoviePageForm.js*
    6. EditMoviePageTable.js

4. **Important**
    1. * If data entry for Add/Create and Edit/Update pages will require *sentences* or *long phrases*, then use the Form versions of the pages. Otherwise, use the Table versions, so the layout stays the same from page to page.
    2. **Change one name at a time** and notice where it is used and what its relationship is to another item. Use the Find/Replace feature in VS Code, but do not change all instances of 'movie' or 'movies' without checking every single instance.
5. **Change icon choices** for Add, Edit, and Delete, so they are not like the demo. Use the **React Icons**
6. **Links to an external site.**
7. website to search for and find the names for your choices. Note that not all icons will render properly on all systems, so if one doesn't work, try another one. Also note that the abbreviation isn't provided; you might have to guess.

    import { *Add*, *Edit* } from 'react-icons/*abbr*'; // **Replace** *abbr* with the correct abbreviation.

    Feel free to use React icons in other places in your website, such as on navigation and headings.

8. **Update all status messages** so they are complete sentences that explain what happened. Do not delete the
`${response.status}. ${errMessage.Error} variables.`

9. Change the path of `useNavigate` so it leads to your Log page, *rather than* the new Home page.

10. When calling the **date**, `.slice()` the default date format, so it displays just the first 10 characters (this excludes the timestamp), like this:
    `useState(logName.dateName.slice(0,10));`

Testing

If your connection to the localhost is not active, right-click on the new frontend folder and choose **Open in Integrated Terminal**.

Test the page in the browser at `http://localhost:8000/`. At the very least, the page heading and table should render. Once all the components and pages are working smoothly with the backend, the data should render on this Log page.

1. Does the Log page **Retrieve** all the documents from the collection?
    1. If not, check to ensure all references to *movies* have been revised.
    2. Check syntax. All variable names must use camelCase. End all statements with a semicolon. Match up starting and ending {}, (), and [].

2. Click the **Create** icon to add a new document to the collection. If the add function was not successful, then:
    1. Check to ensure all references to *movies* have been revised.
    2. Check syntax. All variable names must use camelCase. End all statements with a semicolon. Match up starting and ending {}, (), and [].

3. Click the **Update** icon to edit a specific document in the new Edit table or form. Does all the existing data display? If so, edit data in each form control to ensure it works.
    1. The **date** should be present and sliced to 10 characters.
    2. All data should be **visible** in each form control. If it isn't then adjust the width of `input {}` or `textarea {}` in App.css.

4. **Delete** a row. The response will be an instantaneous deletion without any alert (unless it doesn't work, then an error alert should popup).

Screenshot the Collection/Log page

So that you learn to screenshot a whole web page, and so that our staff has a visual record of your fantastic work, use the following tool to take a single-page screenshot (saved as .png) and submit it with your .zip archive.

1. Navigate to the Log page of your website.

2. Zoom out 1 or 2 times using `cntl-` or `⌘-`.

3. **Squeeze** the browser window, so that it is as **skinny** as possible without the content spilling out of the main area. Do not provide an image of the entire browser and your desktop.

4. Take a screenshot of the skinny vertical window using a tool like this:

   GoFullPage
5. Links to an external site.
6. **:**

   Chrome extension to take a screenshot of an entire webpage.

   FireShot
7. Links to an external site.
8. **:**
   FireFox add-on to take a screenshot of an entire webpage.
9. Optimize/compress the screenshot so it is no larger than **999k**. Drag the screenshot file to the I❤IMG Compress Image
10. Links to an external site.
11. **tool** to reduce the file size.
    1. Optional: If you plan to post your screenshot in Ed Discussion (Student Portfolio Gallery post), then reduce the **height** to **400px tall** and **compress** again.
12. Rename the screenshot with your ONID userID.

13. Add the userName-screenshot.png file to the frontend folder's root (so we can find it).

Convert and update the Gallery page. (0ptional Extra Credit)

Instead of a full-page responsive gallery of your projects, you'll convert that content to a slideshow, using the React Image Gallery package

Links to an external site.

. You previously installed its package, which will allow us to import its component while passing in arrays of the existing images. You'll also need to convert the `figure` tags to an object called `images` that list those arrays.

1. Copy your existing A4 images folder to the new React app's public folder.
2. In the new App.css file, locate the section about importing component stylesheets (just under where you imported fonts) and paste: `@import "~react-image-gallery/styles/css/image-gallery.css";`. The path in this import assumes you have installed the package, and it resides in the node_modules folder.

3. Create a `GalleryPage()` function to replace the A6 gallery.html file:
    1. In the src > pages folder, paste your existing A6 gallery.html file and rename it GalleryPage.js.
    2. Remove all the tags and content from *above* the `<h2>` page name.
    3. Replace that old code with imports for React and the Image Gallery package: `import ImageGallery from 'react-image-gallery';`

Define an **object** with parameters for each of the gallery images:

```
const images = [

    {

      original: 'images/name-of-my-project.png',

      thumbnail: 'images/name-of-my-project.png',

      description: `figcaption text goes here`,

      originalHeight: '450px',

    },
```

4. ... and so on. This object will be used instead of your existing `<figure>` tags. And, the `.gallery {}` rules will not be used. The `originalHeight` value should not be any taller than `450px`, so that the entire image fits above the scroll line. Adjust the value as needed.
    1. If you previously added **pet** photos, **remove** them from the object arrays.
    2. If you previously made hard-to-read **code** screenshots, **remake** them so they are easy to read. Now that you've made a website, you can screenshot some of that code. :-)
5. Begin the function for `GalleryPage()` and provide the `return`. You'll just need the return to include the `<h2>`, `<p>`, and `<article>` tags.

Between the `<article>` tags, add the gallery component and pass in the object name:
`<ImageGallery items={images} />`

6.
7. Remove all the tags and content from *below* the ending `</article>` tag.

Replace that old code with the end of the return and function, then export the default `GalleryPage` function.

```
        </>

    );

}

export default GalleryPage;
```

8.

4. **Important:** **update the App.css file** by *removing* or *commenting* out the `.gallery{}` rule; we can't use it with the new slideshow styles. Or, remove the `class="gallery"` from the HTML.

Now that this page exists, you can import it into the App.js file.

Convert and update the Order page.

(Optional Extra Credit)

The revised Order page will incorporate two components; one for changing the quantity of an item based on clicking React Icons, and one for mapping one row of the product data file.

In addition, the Order page will take advantage of `useState` and increment/decrement the quantity based on the users number of clicks.

Important: This new version of this revised Order page **does not**:

- take the user's name, email, address, or delivery instructions. Do not leave those fields in your file if they are not going to work.
- take a typed quantity, or total the quantity. It will ***only*** allow the users to increase or decrease the quantity dynamically using icon clicks. The main purpose of this page is to get this functionality working.

To begin:

1. **Copy the existing products.js file** into the src > data folder. **Comment out** the `module.exports.products = products;` line and **replace it with** `export default products;`. Also, you may comment out the `console.table(products);` for rendering the object in the terminal, if you are tired of looking at it.

2. Create a component file for a function that uses a React icons. The function will increment and decrement a `setQuantity` between 0 and 10:
    1. In the src > components folder, create a file called ProductQuantity.js.

    2. In ProductQuantity.js import React and useState :
3. `import React, { useState } from 'react';`
    1. Also, import two icons from the React Icon Library
    2. Links to an external site.
    3. . One icon must represent "increase" and one must represent "decrease", (but they cannot be the same ones used in the video demo). Your import for each icon must include the icon component names as well as the library's *abbreviation*, like this:
       `import { Up, Down } from 'react-icons/abbr'; // Replace abbr with the correct abbreviation.`
    4. Create a `ProductQuantity()` function. The function must return the `quantity` based on the user's mouse click for an increase or decrease. **Delete** the *quantity* form control; we won't need it here. Refer to the State and React Hooks Exploration's Bookmark and Counter component examples.

        1. Define a variable for `quantity` and `setQuantity` to take advantage of `useState(0)`.
        2. Define a variable to **increase**.
        3. Define a variable to **decrease**
        4. Return a `<div>` that includes these (in any order you prefer):

            - Icon to increase (`<NameOfIcon onClick={increaser} />`)
            - Icon to decrease (`<NameOfIcon onClick={decreaser} />`)
            - The updated `{quantity}`.
    5. End the return and function, then export the default `ProductQuantity` function.
4. This component will import into the row component (below).
   Create a component file for a single row of the Order table:
    1. In the src > components folder, create a file called ProductRow.js.

In ProductRow.js, import React and the `ProductQuantity` component.

```
import React, { useState } from 'react';

    import ProductQuantity from './ProductQuantity';
```

    2. Use of **./** is needed because the component files are in the same directory.
    3. Create a `ProductRow()` function that passes in a single product. The function must return one row `<tr>` of the table. Each `<td>` will reference the params for: `.company`, `.product` (not .item), and `.price`. The `.price` param will need the built-in `toLocaleString()` function to format in US Dollars. One of the `<td>`s will call the `<ProductQuantity />` component.
    4. End the return and function, then export the default `ProductRow` function.
5. This component will import into the Order page (below).
   Create an `OrderPage()` function to replace the A6 order.html file.
    1. In the src > pages folder, paste your existing A6 order.html file and rename it OrderPage.js.
    2. Remove all the tags and content from *above* the `<h2>` page name.

Replace that old code with imports for `React` and the `<ProductRow />` component:

```
import React, { useState } from 'react';

    import ProductRow from '../components/ProductRow.js';
```

    3. Use of **../** is needed because the component file is outside the pages directory and inside the components directory.
    4. Begin the function for `OrderPage()` and provide the `return`. You'll just need the return to include the `<h2>`, `<p>`, and `<article>` tags.
    5. Between the existing `<table>`'s `<tbody>` tags, call the map feature using arrow
    6. Links to an external site.
    7. syntax so it references the `<ProductRow namePassedToRow={someLocalName} key={index} />`
    8. Remove all the tags and content from *below* the ending `</article>` tag.

Replace that old information with the end of the return and function, then export the default `OrderPage` function:

```
        </>

    );
```

```
}

export default OrderPage;
```

9.

6. (This page *will not submit data* to a route; it will just allow you to dynamically increase and decrease quantities. You are welcome to enhance the capabilities of this page, but it is not necessary and does not earn extra credit.)
   Now that this page exists, you can import it into the App.js file.

Convert the Contact Page. (Optional Extra Credit)

The existing Contact page works well as a static page, but within React, it will not send data without additional package(s) and coding. You are welcome to tackle this using any form package you like. No support will be provided to implement it due to the short timeline of this course.

Be sure to remove any `<Link>`s to it in your navigation if the page does not work.

What to Turn In

Make sure the file structure is like what you see to the right:

- **Remove** your MongoDB account **username** and **password** from the backend folder's .env file. Remember that the .env file will be included in your .zip archive automatically when .zipping/compressing from the hard drive's filing system.
- **Provide 1 zip archive.**
    - Zip the backend and frontend folders together into ***a8-username-portfolio. Leave out both node_modules folders!***
- Change *username* to your ONID username.
- Upload the .zip archive to this Assignment.
- **If you make updates** to your work after you submit the .zip, a new .zip archive must be uploaded.
- **If you submit early, your work may be scored early.** We don't typically have time to score your work twice, so be sure it is ready for scoring when you submit.
- The grader will unzip the archive, and test your site using *their own* MongoDB account.

Pat yourself on the back! Congrats for finishing the course.

We look forward to interacting with your portfolio website!

Rubric and scoring criteria

Review the Rubric below to see how we'll score your assignment:

u23 A8 Portfolio MERN

Criteria	Ratings		Pts
This criterion is linked to a Learning Outcome Framework	**5 to >0.0 pts** **Met** __App frontend and backend launch on the correct ports using the required frameworks. All required files and dependencies are provided. .env file does not include private data.	**0 pts** **Not met**	5 pts
This criterion is linked to a Learning Outcome Home Page	**5 to >0.0 pts** **Met** __Home page launches in the root of the localhost. __The title, headings, and paragraphs have been updated. __An article explains the student's career goals and the technologies used to build the website. __Writing does not have any grammar or spelling errors.	**0 pts** **Not met**	5 pts
This criterion is linked to a Learning Outcome User Experience	**5 to >0.0 pts** **Met** __All images, headings, text, and controls are easy to read. __Navigation from page to page is provided near the top. __Page layout is consistent from page-to-page. __The content is responsive to different browser widths.	**0 pts** **Not met**	5 pts

This criterion is linked to a Learning Outcome Topics Page	**2 to >0.0 pts** **Met** __The Topics page includes articles with IDs that are reachable via a local set of navigation under the page heading. __All text is legible. __There are no grammar or spelling errors.	**0 pts** **Not met**	2 pts
This criterion is linked to a Learning Outcome Collection UI	**10 to >0.0 pts** **Met** __The Collection user interface retrieves documents from the backend and allows users to create, update, and delete data, using appropriate form controls, such as input, select, and textarea. __Input and output works without errors.	**0 pts** **Not met**	10 pts
This criterion is linked to a Learning Outcome Status Messages	**2 pts** **Met** __Status messages for success and failure are helpful and written in complete sentences.	**0 pts** **Not met**	2 pts
This criterion is linked to a Learning Outcome Prepopulated Edit Data	**2 pts** **Met** __Edit command prepopulates all the form controls with existing data. __Data is easy to read in the form controls.	**0 pts** **Not met**	2 pts

This criterion is linked to a Learning Outcome Date is sliced	**2 pts** **Met** __Date is retrieved with just 10 characters. __The timestamp has been sliced out.	**0 pts** **Not met**	2 pts
This criterion is linked to a Learning Outcome React Icons	**2 pts** **Met** __React icons are provided for the create, update, and delete functions. __Icons are easy to perceive because they are large and well-placed. __Icons are unique from the Movie starter code's icons.	**0 pts** **Not met**	2 pts
This criterion is linked to a Learning Outcome Screenshot	**5 pts** **Met** __A screenshot of the Collection/Log page displays only the viewport with data. __The file is optimized to under 999k.	**0 pts** **Not met**	5 pts
This criterion is linked to a Learning Outcome Gallery Page (Optional)	**5 to >0.0 pts** **Met** __Gallery page uses the React Image Gallery package to provide a responsive slideshow of 10 or more optimized images that reflect the student's projects, hobbies, work, and travels. __Color variables have been applied.	**0 pts** **Not provided**	5 pts

This criterion is linked to a Learning Outcome Order Page (Optional)	**10 to >0.0 pts** **Met** __Order page uses React icons to increase and decrease the number of items chosen in each row of data built using the map() feature. __Other form controls have been removed unless they respond with a complete order message.	**0 pts** **Not provided**	10 pts
This criterion is linked to a Learning Outcome Contact Page (Optional)	**10 to >0.0 pts** **Met** __Conctact page incorporates a React package to process and send an email message that includes all the data in the form.	**0 pts** **Not provided**	10 pts
Total Points: 65			