WordPress Site #4 (Headless)

Complete the tasks using the provided React files and your live Mindset site.

Use the Mindset Headless Demo Site as a reference for the tasks.

After completing all other tasks, reflect on your experience by answering some questions when submitting your assignment on the Learning Hub.

The assignment is worth a total of 20 points and 10% of your grade for the course.

Getting Started

- Navigate to the **mindset-headless** folder in your terminal and run the following command:
  - `npm install`
- After the **node_modules** folder and **package-lock.json** file are created, run the following command:
  - `npm run dev`
- This should give a loading icon and a console error. Move on to the next tasks to get it displaying properly.

Utilities.jsx

- In **Utilities.jsx**, replace the URL in **restBase** with your Mindset live site's URL.
  - Note: This should end with **/wp-json/wp/v2/**
- You will concatenate onto this URL in the React components mentioned below.

Page Tasks

Home – (1 point)

- In **Home.jsx**, complete the **endpoint** in the restPath constant so that it goes to your Home page.
- This should output the title and content of your Front Page. The CSS and JS from the block editor does not automatically come over so it will look broken but you will fix some of the styling later.

Contact – (2 points)

- Create a **Contact** component by duplicating your Home component.
- In **App.jsx**, import the **Contact** component, add a link to a Contact page in the header navigation, and add a Route in the main element.
- In the **Contact** component, add the **endpoint** to your Contact page. Modify the return code to output the page title, and the custom (meta) fields for the Company Address and Company Email. If your address field includes HTML tags from the WordPress database, check the Home component for an example of how to deal with that in React.

Page Template – (2 points)

- Create a **Page.jsx** component by duplicating your Home component.
- Add a parameter of **pageID** to the exported constant and use that parameter to build a **dynamic endpoint** to the Page whose ID is passed to the function.
- In **App.jsx**, import the **Page** component, add a link to a **Services** page in the header navigation, and add a Route with a **pageID** parameter in the main element. Pass in the ID of the Services Page from your live Mindset site.
- Use this **Page.jsx** component to also create Routes to your About and Privacy Policy Pages. Add links to About and Privacy Policy in the footer navigation.

Blog Tasks

Blog Index: Author's Name – (1 point)

- In **Posts.jsx**, add the **endpoint** to your Posts.
- Output the name of the **Author** of each Blog post. This is available in the JSON data when you query the REST API for the optional **embeddable** content.

Single Blog Posts: Featured Images – (2 points)

- In **Post.jsx**, add a **dynamic endpoint** to each single Blog post by using the **slug** parameter stored as a constant. Get the embeddable content too.
    - *Note: The REST endpoint URL will use a slug different than an ID.*
- Add the necessary code to use the **FeaturedImage** component before outputting the post's title. The featured images already appear on the Blog index, so check **Posts.jsx** to see code that is *similar* to what you need.

Single Blog Posts: Next Post Link – (3 points)

- In the provided **fwd-rest-api-plugin**, register a second REST field to store a link to the next Post.
- Update the version number of the WordPress plugin.
- Upload the plugin to your live Mindset site.
    - *Note: You may need to clear the server cache after uploading.*
- After verifying that the new REST field exists and has data, output the link to the next Post in the nav element that contains the link to the previous Post.
    - *Reminder: The newest Post will **not** have a "next" Post and the oldest Post will **not** have a "previous" Post.*

CPT & Taxonomy Tasks – (3 points)

You will practice retrieving and outputting posts directly from a Custom Post Type with these tasks.

- Create a **Works** component. You may want to duplicate the **Posts.jsx** component to use as a starting point.
- In **App.jsx**, import the component, add a link to a Works page in the header navigation, and add a Route in the main element.
- In the **Works** component, write the **endpoint** so that it does the following:
    - goes to the Work CPT,
    - orders the Work posts by title,
    - orders the Work posts from A to Z.
    - gets the embeddable content,
- In the **Works** component, display the Title, Featured Image, Content, and Work Category for each Work post.

ACF Tasks – (4 points)

You will create a new Page on your Mindset site that only uses ACFs for these tasks.

- In the backend of your live Mindset site, create a Page titled **Profile**.
- Install ACF Pro and use the ACF Documentation for the remaining steps.
- Create a Field Group assigned to this Page with these four fields:
    - ❖ Field Type: Text
    - ❖ Field Label: Profile Name
    - ○ Field Type: Text Area
    - ○ Field Label: Profile Bio
    - ○ Note: Set New Lines to automatically add paragraphs.
    - ➢ Field Type: Image
    - ➢ Field Label: Profile Photo
    - ➢ Note: Return format should be Image Array.
    - ▪ Field Type: Relationship
    - ▪ Field Label: Related Projects
    - ▪ Note: Filter by Post Type of Work. Return Format as Post ID.
- *Don't forget to make the Field Group **Show in REST API**.*
- Edit the Profile Page to fill in the Text and Text Area fields with any text and use any image for the Image field. For the Relationship field, choose 4 Work posts.
- Create a **Profile.jsx** component by duplicating your Home component.
- In **App.jsx**, import the **Profile** component, add a link to a Profile page in the header navigation, and add a Route in the main element.
- In **Profile.jsx**, add the **endpoint** to your **Profile** Page with the necessary URL parameter to use the ACF return values you set.
- In **Profile.jsx**, instead of outputting the content, output the four ACF fields:
    - ○ Output Profile Name inside an h2 and Profile Bio inside of a div.
    - ○ Profile Photo should use the provided **ACFImage** component.
        - ▪ Read the comments in the component to see what data to pass.
    - ○ Related Projects should use the provided **RelatedProjects** component.
        - ▪ Read the comments in that file to because it is incomplete.

Styling Tasks – (2 points)

You are welcome to style the app further than this but completing these styling tasks at a minimum:

- **Home**
  - Fix the layouts for the Featured Works and Latest Blog Posts.
- **About**
  - Remove the bullet points and spacing for the testimonials.
- **Services**
  - Style the in-page navigation.
- **Profile**
  - Style this page according to the Demo site or your own design.

Optional Tasks

Job Postings CPT

If you added the optional Job Postings custom post type to your Mindset site, you can add those to your React app.

- Create **two** new components very similar to **Posts.jsx** and **Post.jsx**.
- In **App.jsx**, import the both components, add links and routes similar to the Blog.

More ACFs

If you are interested in seeing one of the most useful ACFs, add a Repeater Field to the Profile page. Create a couple of sub-fields and see if you can access and output that data in React. The Repeater and Flexible Content fields in ACF are hugely beneficial in classic themes and headless sites that don't use the Block Editor. You may consider using them for your Portfolio Projects.

Build the App

To build then upload the React app to your server:

- Decide the name of the folder that will contain your app on the live server.
    - If the URL will be example.com/mindset-headless, then the folder will be “mindset-headless”.
- Edit **package.json**. Set the value of homepage to the folder name. For example:
    - `"homepage": "/mindset-headless"`
- Edit **main.jsx**. Change the value of basename in <BrowserRouter> to the folder name. For example:
    - `<BrowserRouter basename="/mindset-headless">`
- Edit **vite-config.js**. In the defineConfig object, set the value of base to the folder name. For example:
    - `base: '/mindset-headless/'`
- Create a **.htaccess** file in the public folder. Copy the code here and read the comments to make sure your app can refresh on all URLs after it is built:
    - https://gist.github.com/jtleathers/8e534489a512f67aa6ee1d5969e0c764
- Run the following command:
    - `npm run build`
- Upload the “dist” folder to the live server and then rename it to match what you specified as the folder name above. Check that the app works on the server.

Reflection & Submission

Upload the development files to the Learning Hub:

- Zip **only** the “src” folder and upload it to the Learning Hub. You do not need to include any other React files/folders or the WordPress plugin.

Answer the following questions when submitting on the Learning Hub:

1. What is the URL to your completed Headless WordPress React app?
2. What do **you personally** think are one or two advantages of using a headless CMS instead of a traditional CMS?
3. What do **you personally** think are one or two disadvantages of using a headless CMS instead of a traditional CMS?
4. Do you feel confident you could use this React app as a reference to build another React app that pulls data from a different WordPress website?