IMY 220
Assignment 4

NodeJs & React Intermediate

General Instructions

- This assignment must be completed and submitted by the due date which is available on ClickUP.
- This assignment is a take-home assignment and may take one or two days to complete.
- This assignment should be completed on your own, but you may come to the tutorial session or email the assistant lecturer if you need further assistance.
- No late / email submissions will be accepted.

This assignment focuses on **NodeJs and React, and React Intermediate (React Router, Component State)**. You will be required to create a simple React application using webpack and babel that displays a list of posts that can be edited.

Note**: For this assignment you are required to set up a React project using webpack and babel as is taught in the lecture videos. You must make use of the correct file structure, and each component should be in a **separate file** in the correct folder as taught in the lecture videos. **You may not use a builder such as Create-react-app or Vite. Doing so will result in -10% from your assignment mark.

Part 1 - React App With Webpack and Babel

Create a **simple React App using webpack and babel** as taught in the lecture 19 videos (NodeJs & React).

You are more than welcome to copy and paste in a different project that you have already set up from another practical or your semester project, just make sure that you have a **working project**, and that you **only** have what is required from you for this assignment (i.e., please remove any extra files from your project or your practical).

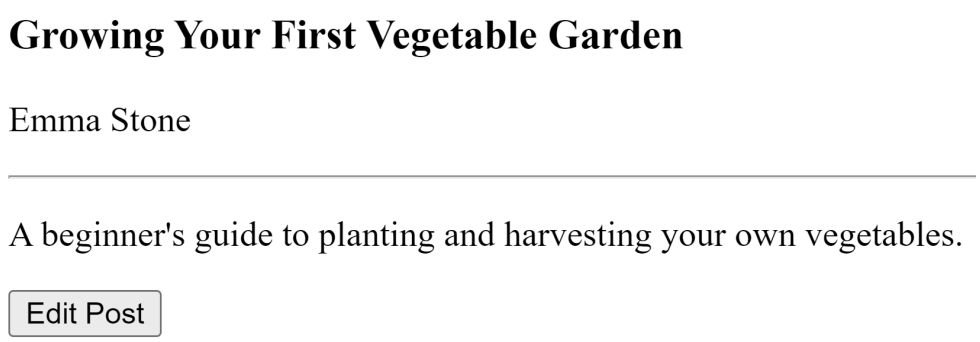
You do not have to set up the folder structure like you need to for your project (i.e., with `frontend` and `backend` folders, etc.) but you may do so if you like.

The rest of your assignment depends on this step working, so make sure you have set everything up correctly before moving on to the next step.

Part 2 – Post Component

In a file called ***Post.js*** in the components folder…

Create a class component called **Post** that displays some details for a post as follows:



This component should get its title, author and description from props that are passed in. In other words, the values **shouldn't be hardcoded**.

- This component should have an **edit post** button that, when clicked, shows a previously hidden form that allows you to edit that post (this form is defined further in the next step).
- If the **edit post** button is clicked **again**, the form should again be hidden. The form should be hidden by default when the user loads the page.
- After the form is submitted, the post's *data should update* (this is also further explained in the next step).

Therefore, the **Post** component should also have **state** in order to keep track of the **visibility** of the edit form, and the post's **data**.

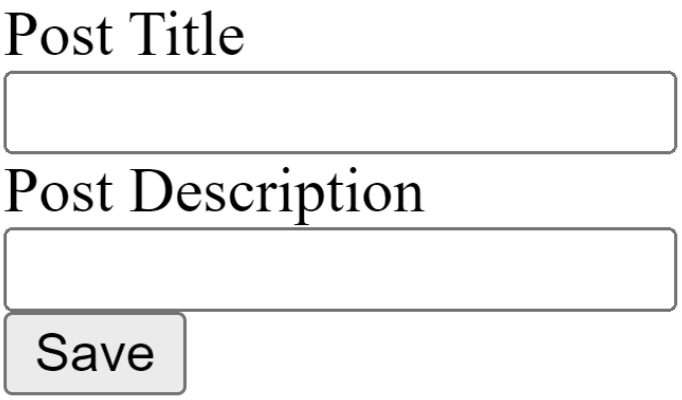
For now, to test show/hide functionality, you can render a simple "Hello World" string that you show/hide by clicking the edit post button.

Import this component into your index.js and render this component in the "#root" div in the browser with some test data to test your code is working before moving on to the next step.

Part 3 – EditPost Component

In a file called ***EditPost.js*** in the components folder…

Create a class component called **EditPost** that allows you to edit the **title** and **description** of the corresponding *Post* component. The form should look as follows:



You do **not** need to do any sort of form validation here (i.e., the user can enter in anything for the title and description).

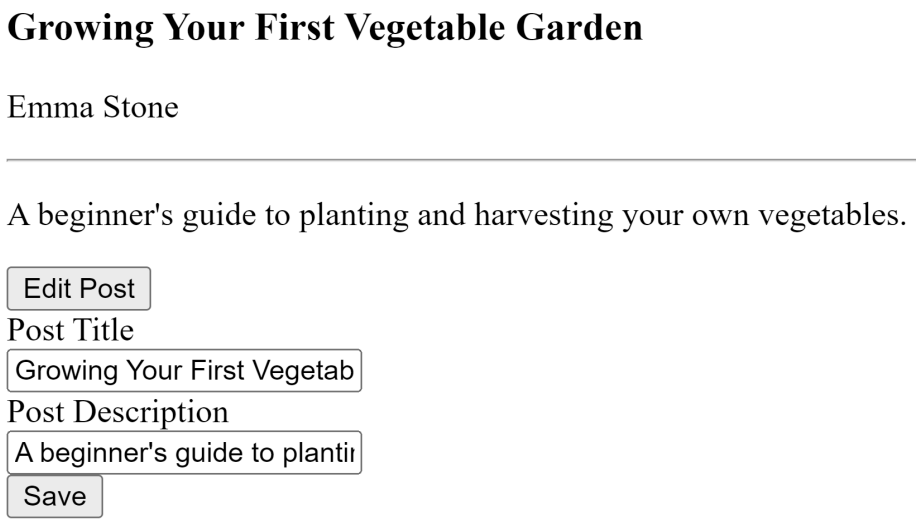
This form should be rendered inside the *Post* component, and should be shown/hidden when the **edit form** button is clicked.

When the *EditForm* is shown, its **input fields should already be filled** with the post's title and description (***Hint****: you can make use of the input field's default value for this*).

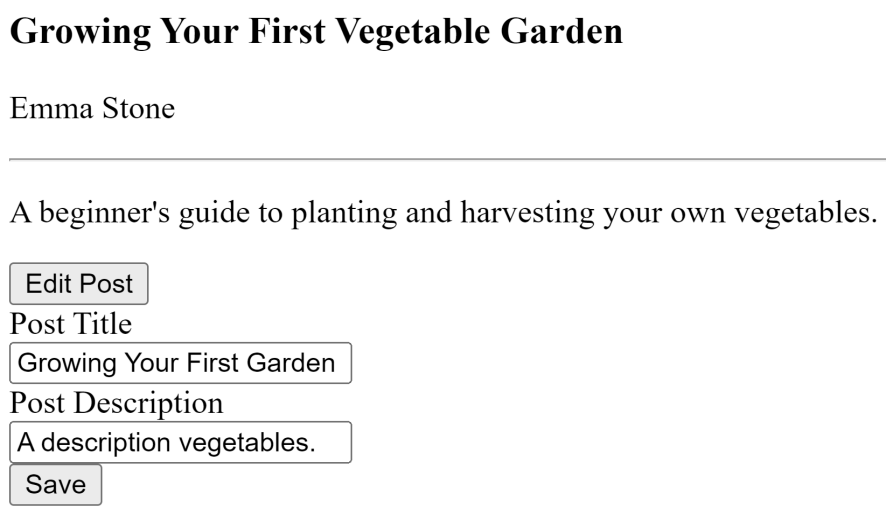
When the form's **submit** button is clicked, the values (title, description) in the *Post* component should be **updated** with the form's input values.

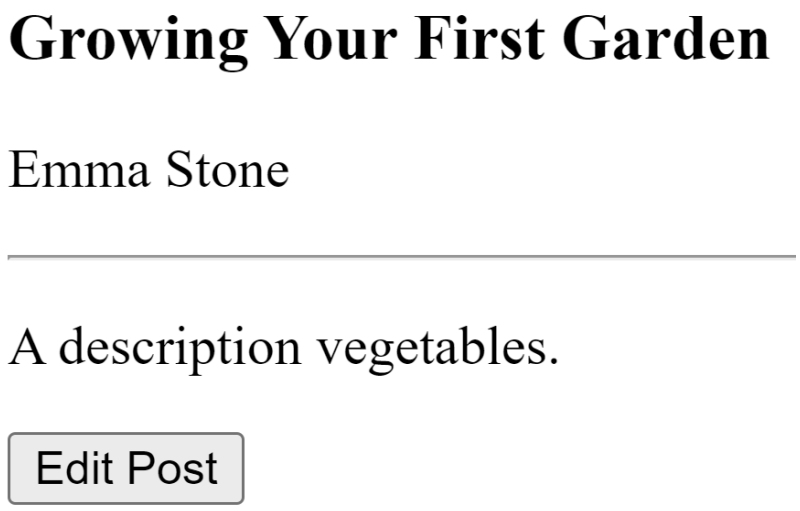
- To do this, you will need to define a function in **Post** that updates the state, that you then pass to **EditPost** and call from within a function in EditPost.
- The form should also be automatically **hidden**. You can do this from within the **Post** component.

If you have done this correctly, the interaction should look as follows.
Clicking the edit post button brings up the populated form like this:



The user can then edit the form values and click “save” and the component will be updated, and the form is hidden again.





Import the Post component into your index.js and render this component in the “#root” div in the browser with some test data to test your code is working before moving on to the next step.

Part 4 – Basic Routing with React Router

The last part of this assignment requires you to set up some very basic routing using **React Router**.

Start by installing the required React Router package: **react-router-dom**

Create a component called **App** in a file called **App.js**.

- *App.js* should be in the **src** folder on the same level as *index.js*.

- This component should be used to set up your different routes using the **BrowserRouter** as taught in the lecture videos.
    - When the user navigates to “http://localhost:3000/”, the **Home** page should be shown (defined below)
    - When the user navigates to “http://localhost:3000**/posts**”, the **Posts** page should be shown (defined below)
- You may make use of either method (old or new), as long as the routing works.

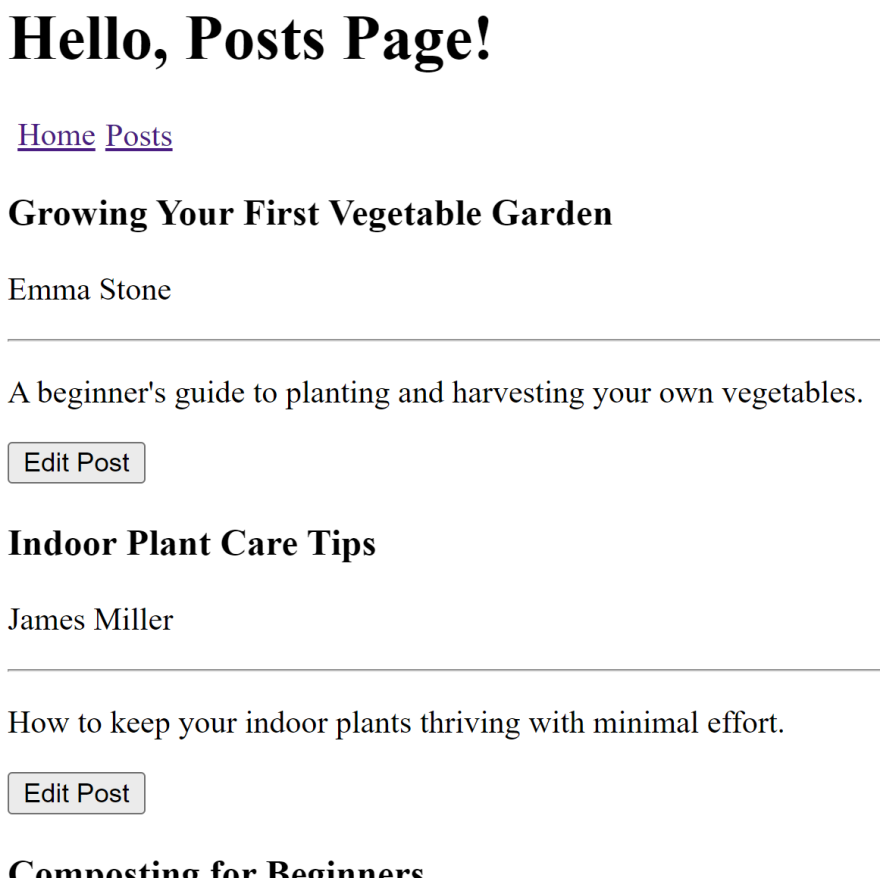
Create a component called **Home** in a file called **Home.js**

- *Home.js* should be in a folder called **pages** in the **src** folder. The pages **folder** should be on the same level as the components **folder** (i.e., pages and components should be siblings in the hierarchy).
- This component can render a simple page as follows:

- ***Note**: A bit of padding has been added to the links to make them more readable.*
- You should use the **React Router equivalent of rendering links** for the two links.

Create a component called **Posts** in a file called **Posts.js**

- *Posts.js* should be in a folder called **pages** in the **src** folder (with *Home.js*).
- This component should render a list of **Post** components on the page as follows:



- ***Note**: A bit of padding has been added to the links to make them more readable.*
- You should use the **React Router equivalent of rendering links** for the two links.

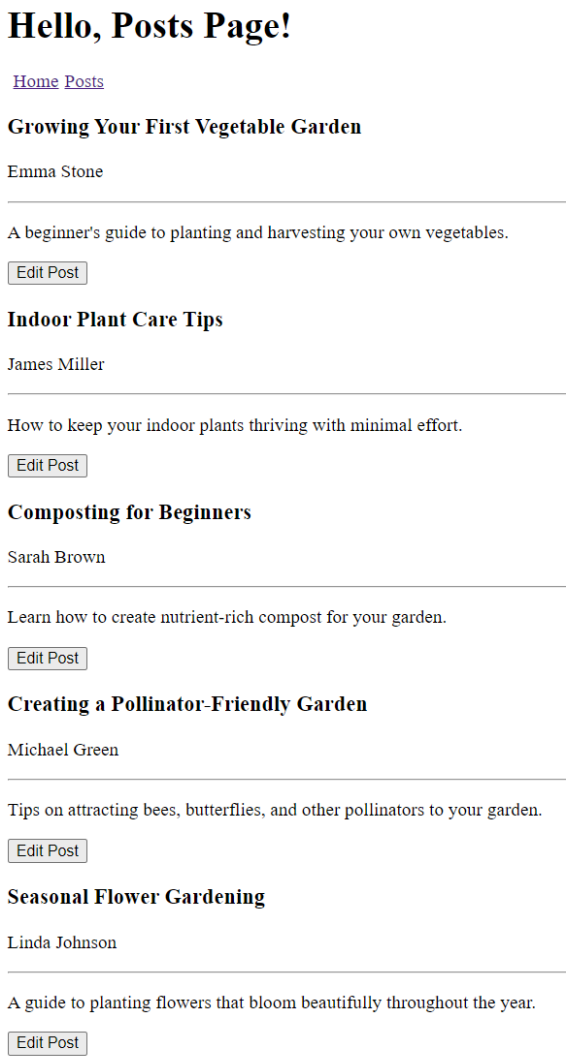
You can use this array to pass data to your posts:

```
const postsArr = [
  {
    title: "Growing Your First Vegetable Garden",
    author: "Emma Stone",
    description:
      "A beginner's guide to planting and harvesting your own vegetables.",
  },
  {
    title: "Indoor Plant Care Tips",
    author: "James Miller",
    description: "How to keep your indoor plants thriving with minimal
effort.",
  },
  {
    title: "Composting for Beginners",
    author: "Sarah Brown",
    description: "Learn how to create nutrient-rich compost for your
garden.",
  },
  {
    title: "Creating a Pollinator-Friendly Garden",
    author: "Michael Green",
    description:
      "Tips on attracting bees, butterflies, and other pollinators to your
garden.",
  },
  {
    title: "Seasonal Flower Gardening",
    author: "Linda Johnson",
    description:
      "A guide to planting flowers that bloom beautifully throughout the
year.",
  },
];
```

Import the **App** component into your *index.js* and render this component in the "#root" div in the browser. This should be the root of your application.

At the end, your final application should look like this:
When the user starts the server (and navigates to http://localhost:3000/"):

When the user clicks on the “Posts” link (or navigates to http://localhost:3000/posts”):



Each post should be able to be edited and updated as described in the steps before. This state does **not have to persist**, i.e., if the user reloads the page or clicks on one of the links again, the state can **reset**.

Submission Instructions

Place these in a folder named `A4_u12345678` where `12345678` is your student number.

- Submit *all* the files required for this assignment to work (including the config files for babel and webpack) **except** for your node_modules folder and its contents.
- **Do not submit your node_modules folder. This will result in -10% from your assignment mark.**

Zip the **folder** and upload this to ClickUP in the relevant submission slot before the deadline.