Demo Steps

Installation

1. Install homebrew/chocolatey
    a. On Mac/Linux -- https://brew.sh/
    b. On Windows -- https://chocolatey.org/install
2. Install npm
    a. On Mac/Linux: `brew install node`
    b. On Windows: `choco install node`

3. Install ionic
    a. Run `npm install -g ionic`

Setup

1. Change the contents of the IonLabel in src/App.tsx on Line 57 to from “Tab 1” to “Submit”, and the contents IonLabel on line 61 from “Tab 2” to “News”
2. Delete Lines 63-66, deleting the tab 3 component, and delete lines 47-49 to delete the route to tab 3
3. Replace the Route for tab 1 and tab 2 to

```
<Route path="/tab1" component={Tab1} exact={true} />
<Route path="/tab2" component={Tab2} exact={true} />
```

4. Run `ionic serve`
    a. The application should open automatically, but if not, go to http://localhost:8101/ in your browser
5. Change home redirect to “tab2” instead of “tab1” on line 47
    a. Refresh local host to see the change
6. Examine “Tab1.tsx” and “Tab2.tsx” in src/pages to familiarize yourself with the components
7. Make three folders in the src/pages directory, “NewsArticle”, NewsList”, “SubmitArticle”
8. Create “Details.tsx” in the “NewsArticle” folder. This will show the detailed news article
9. Delete “Tab1.css” and go to “Tab1.tsx” to remove the css import
10. Move “Tab1.tsx” into the “SubmitArticle” Folder. This will be the page that has the form for submitting articles.
11. Delete the “Tab2.css” file. Then delete the css import within “Tab2.tsx”
12. Move “Tab2.tsx” into the NewsList Folder
13. Delete “Tab3.tsx” and “Tab3.css” files, as it is not needed
14. Remove the reference to “tab3” in the “App.tsx” on line 16
15. Create a directory “src/resources” and then create a file inside called “article.ts” within it

Now we’re done with the preliminary setup!

Developing

16. Go to the “Tab2.tsx” and begin developing the code. Delete everything under the “<IonContent>” component.
17. Then change the contents of <IonTitle> to “What's News”
18. Create an article object map in “article.ts” using this code

```
export interface article {
   date: string,
   author: string,
   body: string,
   title: string,
   picture: string,
   id: number,
 }
  export var articles: article[] = [
   {
     date: "01/01/2020",
     author: "Bob",
     body: `Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut
     labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi
     ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate
velit esse cillum
     dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in culpa qui officia
     deserunt mollit anim id est laborum.`,
     title: "Latin Phrase",
     picture: "https://cdn.auth0.com/blog/get-started-ionic/logo.png",
     id: 0,
   }
 ]
```

19. Then jump back to the “Tab2.tsx” file.

We will be creating an article instance that will be clickable.

In order to properly get our article, we need to add the following import:

```
import { articles } from "../../resources/article"
```

And modify the Ion imports so that all of the following are imported:

```
import { IonContent, IonHeader, IonPage, IonTitle, IonToolbar, IonCard, IonCardTitle,
IonCardSubtitle, IonCardContent, IonImg } from '@ionic/react';
```

Then we add the following code block for IonContent so that we can display the content of the articles

```
<IonContent fullscreen>
       {articles.map((article) => (
         <IonCard key={article.id} routerLink={`/articles/${article.id}`}>
             <IonCardTitle>{article.title}</IonCardTitle>
             <IonCardSubtitle>{article.author} {' | '} {article.date}</IonCardSubtitle>
             <IonImg src={article.picture}></IonImg>
             <IonCardContent className="card">{article.body}</IonCardContent>
         </IonCard>
       ))}
</IonContent>
```

20. Then we will create a css file in the NewsList directory called “card.css” and include the following lines

```
.card {
   white-space: nowrap;
   overflow: hidden;
   text-overflow: ellipsis;
}
```

This truncates the text output so that we can show a brief preview. We can click on the article to expand it.

21. Import card.css in Tab2.tsx

```
import "./card.css"
```

22. Head back to the “App.tsx” where you’ll import Details

```
import Details from './pages/NewsArticle/Details';
```

And then in IonRouterOutler change the root path so that it includes a render that would display the added news articles

```
<Route path="/" render={() => <Redirect to="/tab2" />} exact={true} />
```

Add the following Route for the Details component

```
<Route path="/articles/:id" component={Details}/>
```

This will allow us to create a dynamic route based on our NewsList Component

23. Jumping over to the “NewsArticle/Details.tsx” we will specify these imports as well as

```
import React from 'react';
import { IonBackButton, IonButtons, IonHeader, IonPage, IonToolbar, IonTitle, IonContent,
IonCardTitle, IonCardSubtitle, IonCardContent, IonImg } from '@ionic/react';
import { articles } from "../../resources/article"
import { RouteComponentProps } from 'react-router';
type ArticleID = RouteComponentProps<{
 id: string;}>;
```

24. Then we will change the News Article in order to then display the expanded version of our selected news article, this will include reusing some more Ionic Components, and displaying the full news article along with the title, author, date, picture, and full body

```
const Details: React.FC<ArticleID> = ({
 match: {
   params: { id },
 },
}) => {
 return (
   <IonPage>
     <IonHeader>
       <IonToolbar>
         <IonButtons slot="start">
           <IonBackButton defaultHref="/tab2" />
         </IonButtons>
           <IonTitle>What's News</IonTitle>
       </IonToolbar>
```

```
      </IonHeader>
      <IonContent>
        <IonCardTitle>{articles[+id].title}</IonCardTitle>
        <IonCardSubtitle>{articles[+id].author}{' | '}{articles[+id].date}</IonCardSubtitle>
        <IonImg src={articles[+id].picture}></IonImg>
        <IonCardContent>{articles[+id].body}</IonCardContent>
      </IonContent>
    </IonPage>
  );
};

export default Details;
```

This allows us to get the params from the route, being the “id” and use it in our method in order to properly display the expanded form of the article. We then further define it in our component declaration.

Finally, we will implement the submit article functionality which will allow anyone to submit an article to us. For this let’s jump over to the “SubmitArticle/Tab1.tsx” file. The remaining steps will all be within this file.

25. Let’s update our imports so the following are imported:

```
import React, { FormEvent, useState } from 'react';
import { article, articles }  from "../../resources/article"
import {
 IonHeader,
 IonPage,
 IonTitle,
 IonToolbar,
 IonList,
 IonButton,
 IonLabel,
 IonText,
 IonItem,
 IonContent,
 IonInput,
 IonTextarea
} from '@ionic/react';
```

26. Replace the return statement as seen below in order to clear the content.

```
const Tab1: React.FC = () => {
 return (
   <IonPage>
     <IonHeader>
       <IonToolbar>
         <IonTitle>Submit Your Own Article</IonTitle>
       </IonToolbar>
     </IonHeader>
     <IonContent>
     </IonContent>
   </IonPage>
 );
};
```

27. Let's declare a test article in order to show and maintain the state of the article component. This will go before the return statement.

```
var beginArticle: article = {
   date: "01/01/1970",
   author: "Test Author",
   body: "Test Body",
   title: "Test Title",
   picture: "https://miro.medium.com/max/2400/1*y6C4nSvy2Woe0m7bWEn4BA.png",
   id: articles.length,
 };
```

28. Let's add some state mutators to reflect the state changes in the input components.

    This uses 'react-hooks' in order to get, save, and modify the state and then re-render the affected components. The state changes result from us submitting a new article. The following will go after the beginArticle declaration and before the return statement.

```
 const [author, setAuthor] = useState(beginArticle.author);
```

```
const [date, setDate] = useState(beginArticle.date);
const [body, setBody] = useState(beginArticle.body);
const [title, setTitle] = useState(beginArticle.title);
const [picture, setPicture] = useState(beginArticle.picture);
```

29. Next, create the input forms for adding new articles. These will use a blend of ionic components and pure HTML components.

    We use “onIonChange” and cast the “event.target” as a “HTMLFormElement” to pass the inputted value to typescript. This is because “IonInput” is an Ionic Component, and typescript needs the type to be strictly defined.

```
<IonContent fullscreen>
     <form onSubmit={e => addToArticles(e)}>
     <IonList >
       <IonItem>
       <IonLabel position="stacked">Author <IonText color="danger">*</IonText></IonLabel>
       <IonInput value={author} name="author" onIonChange={event => setAuthor((event.target
as HTMLInputElement).value)}></IonInput>
       </IonItem>
       <IonItem>
       <IonLabel position="stacked">Date <IonText color="danger">*</IonText></IonLabel>
       <IonInput value={date} name="author" onIonChange={event => setDate((event.target as
HTMLInputElement).value)}></IonInput>
       </IonItem>
       <IonItem>
       <IonLabel position="stacked">Body <IonText color="danger">*</IonText></IonLabel>
       <IonTextarea value={body} name="author" onIonChange={event => setBody((event.target
as HTMLInputElement).value)}></IonTextarea>
       </IonItem>
       <IonItem>
       <IonLabel position="stacked">Title <IonText color="danger">*</IonText></IonLabel>
       <IonInput value={title} name="author" onIonChange={event => setTitle((event.target as
HTMLInputElement).value)}></IonInput>
       </IonItem>
       <IonItem>
```

```
      <IonLabel position="stacked">Picture <IonText color="danger">*</IonText></IonLabel>
      <IonInput value={picture} name="author" onChange={event => setPicture((event.target
as HTMLInputElement).value)}></IonInput>
      </IonItem>
      <IonButton expand="block" type="submit" class="ion-no-margin">Create
Article</IonButton>
    </IonList>
    </form>
    </IonContent>
```

30. Finally, we will make an event listener that sets the fields as specified by the user. The news window will now include the newly added article. This will all happen when the “Submit Article” button is clicked.

```
const addToArticles = (event: FormEvent<HTMLFormElement>) => {
  event.preventDefault();
  beginArticle.author = author;
  beginArticle.body = body;
  beginArticle.date = date;
  beginArticle.picture = picture;
  beginArticle.title = title;
  articles.push(beginArticle);
 }
```

Note: The articles don’t save after refreshing your browser or opening a new window because we aren’t persisting the list in either local storage or in a backend API.