WEB102 | Intermediate Web Development

For-Credit-Intermediate-Web-Development-Spring-2024 (@ Florida Atlantic University)
Personal Member ID#: 96406

Lab 1: Timetabled

Overview

In this project, you will build a grid-style calendar with one-hour events that plan out a single week. You must create the calendar for someone else. This can be a friend, family member, historical figure, role model, or imaginary person. Timetables can be informative, humorous, or exploratory. For example:

- General: Plan a vacation for a friend, family member, or pet
- History: A week in the life of a historical figure, timeline of a coup
- Psychology: How to adapt to a polyphasic sleep cycle
- True crime: The week before a famous crime

View an exemplar of what you'll be creating in this lab here!

Required Features

- A one-week calendar that includes one-hour time blocks
- Events have different titles
- Events have different colors based on their type

Stretch Features

- Event blocks have additional information, such as a description and location

Resources

- Getting Started with Vite
- ReactJS: Introducing JSX
- ReactJS: Rendering Elements
- ReactJS: Components and Props

Lab Instructions

Getting Started

GitHub Copilot is an AI-powered code completion tool developed by GitHub in collaboration with OpenAI. Leveraging the GPT-3 model, Copilot suggests whole lines or blocks of code as developers type, helping them write code faster and with fewer errors. It's trained on a vast amount of public code repositories, enabling it to provide suggestions in various languages and frameworks.

GitHub Copilot Chat is a chat interface that allows developers to ask and receive answers to coding-related questions within a supported IDE. The chat interface is contextually aware of the code a developer has typed and any error messages.

Note: Alternatively, you can use ChatGPT as an AI assistant instead of Copilot. This will require you to copy and paste code to and from your IDE and ChatGPT instead of using the Copilot extension in VS Code. A ChatGPT account is free of charge and gives you access to ChatGPT 3.5.

Getting GitHub Verified Student Access

Verified student access is required to obtain **free access** to GitHub Copilot, otherwise it is only available as a paid service ($10 per month after 30 day free trial). You will need to complete this step before requesting access for GitHub Copilot Chat.

- [ ] Go to GitHub's Student Developer Pack page at **education.github.com/pack**.
- [ ] Click on the green **Sign up for Student Developer pack** button.
- [ ] Under **Individuals**, click on **Get student benefits**.
- [ ] Under **Select the academic status**, select **Student**.
- [ ] Select or add the email address you use for school, enter your school's name, and describe how you plan to use GitHub.
- [ ] Click **Continue**, and you will then be prompted to upload proof of your academic status.
- [ ] Click **Take a picture** to use your computer's camera to upload proof.
    - [ ] Place your school ID or other proof of current academic status in the frame, then click **Take photo**.
    - [ ] Under **Proof Type**, select the type of proof you are providing.
- [ ] Verify your application details, then click **Process my application**.

Note: If you see a banner asking you to fix something in your application after clicking the **Process my application** button, fix it and then click **Reprocess my application**.

You'll receive a confirmation email letting you know if your application is approved. Sometimes this happens within a few minutes, but it can sometimes take a few days.

Verified student access also gives you access to the **GitHub Student Developer Pack**, which is a toolkit offered by GitHub to support your learning and development in the coding world. It grants you **free access** to the best developer tools in one place so you can learn by doing and be ready for the professional world.

Requesting Access for GitHub Copilot Chat

Do this step *after* you have received confirmation that you have verified student access on GitHub.

- [ ] Navigate to **Github Copilot Chat Signup**
- [ ] If you are not already logged into ChatGPT, click the green **Sign in to join the waitlist** button.

> **Note:** If you see a message about needing to have a GitHub Copilot license, you do not yet have verified student access on GitHub. Try again after you have received confirmation that you have verified student access.

- [ ] Select Visual Studio Code, then click the green **Join the waitlist** button.

You should receive access to GitHub Copilot Chat within a few days, although it can take up to a week. It is important to complete this step as soon as possible to ensure you have access before the first class.

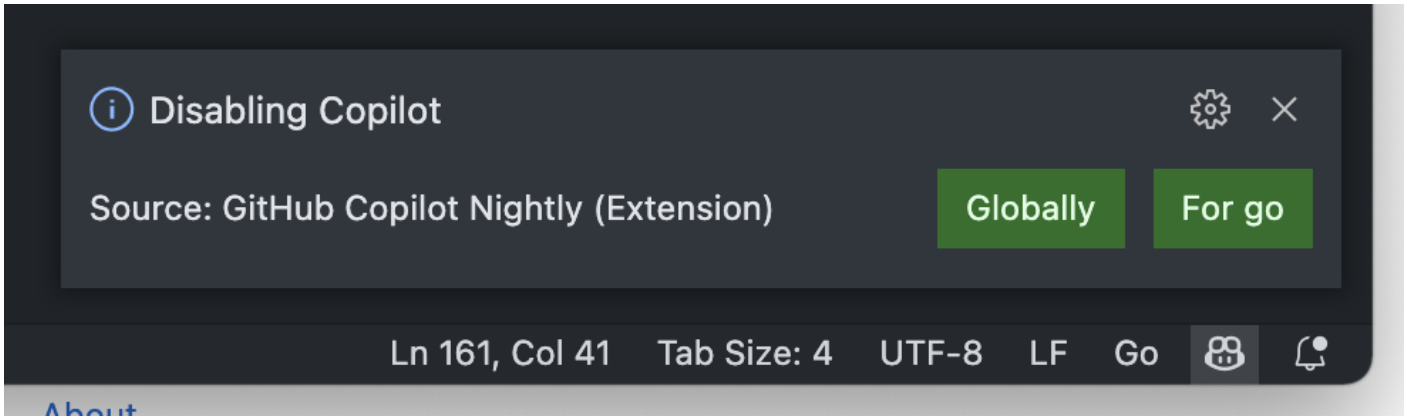
Adding GitHub Copilot to VS Code

You will need to use the Insiders edition of VS Code to access GitHub Copilot Chat.

- [ ] Download and install **VS Code Insiders**
- [ ] In VS Code Insiders, click the **Extensions** icon.
- [ ] In the search bar, type **copilot**.
- [ ] Install the **GitHub Copilot** and **GitHub Copilot Chat** extensions.

We'll be using the Copilot Chat to leverage it as a programming assistant instead of using the inline code-generating feature. We recommend disabling the inline Copilot as it can be distracting and sometimes not directly related to the task we're trying to do.

To disable the inline Copilot, click the Copilot icon in the bottom right corner of the status bar. If prompted, choose **Globally**.



Required Features

Step 0: Create a New React Project Using Vite

In this step, we will create a new React project using Vite.

- [ ] Download and install Node.js
- [ ] Open the Terminal in VS Code using the menu (`View -> Terminal`) or the shortcut (`ctrl + ` `)
    - [ ] Navigate to the folder on your computer where you keep GitHub files (it may be named `github` , or you may wish to create a new folder using the `cd` (change directory) command.

If you want more info about how moving within the Terminal works, check out this tutorial.

▶ 🤖 **AI Pro Tip: Using AI to Explore New Concepts**

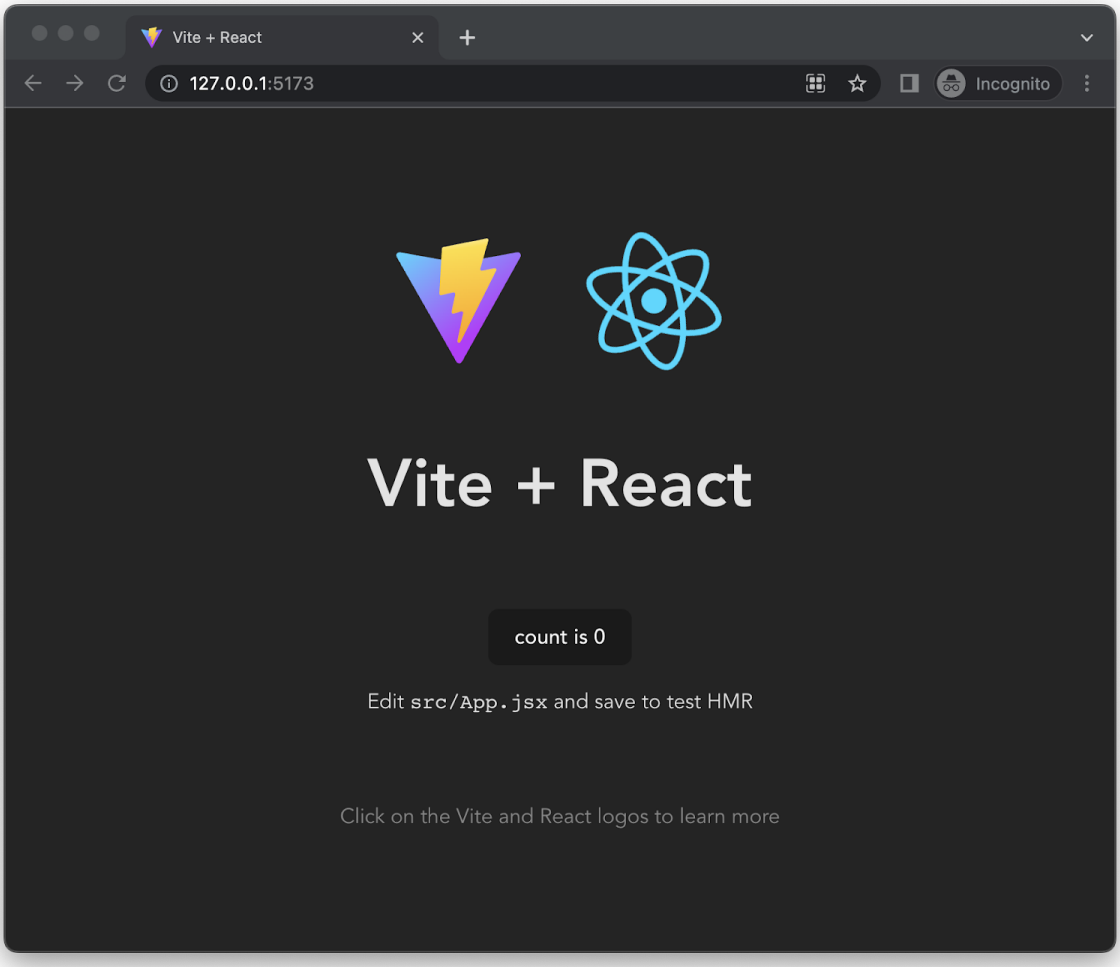
- [ ] Make sure you have added the folder to your VS Code workspace. If you don't see the folder in the Explorer, right-click in the pane, select `Add Folder to Workspace...`, and then add your folder.
- [ ] In your Github repository, initialize a new React project using Vite:
  - [ ] In the Terminal, enter the command `npm create vite@latest`
  - [ ] Name the project `timetabled`
  - [ ] Select **React** as the framework (use the arrow keys and `enter` to navigate the menu)
  - [ ] Select **JavaScript** for the language variant

Now, let's install the required dependencies and run the app!

- [ ] Move into the `timetabled` directory: `cd timetabled`
- [ ] Install the dependencies by running the command `npm install`
- [ ] Run the application in developer mode by running the command `npm run dev`
- [ ] Open project in the browser. Vite will display a link, such as `http://127.0.0.1:5173` to click on or copy/paste that will take you to the localhost port where the project is running.

💡 **Tip:** If you'd like to stop the server, you can use `ctrl + c` or `cmd + c` within the Terminal, or use the trash can icon in the top right of the Terminal within VS Code. To run the server again, simply use `npm run dev` again.

📍 **Checkpoint 0**: At this point in the lab, your app should look like this:



Step 1: Update the Starter Code for Root File `App.jsx`

In this step, we will modify the root file of our React application. In `main.jsx`, you will see a method named `createRoot()`. This method lets you create a root section to display React components inside a browser DOM node. Based on the provided starter code, the root of our React applications is the `App` component.

- [ ] Go to the root file `App.jsx` in the `src` directory.
- [ ] Replace the provided starter code in `App.jsx` with the following code:

```
import './App.css';

const App = () => {

  return (
    <div className="App">


    </div>
  )
}

xport default App
```

At this point, saving your code will update the project and you should see the page in your browser refresh and display a blank screen.

The above code snippet uses JavaScript ES6. JavaScript ES6 brings new syntax and features to make your code more modern and more readable.

▶ 🤖 **AI Pro Tip: Using AI to Explore New Concepts**

The first line of code `import './App.css'` imports the CSS file with code that styles of `App` or the root of the application. Style rules included in `App.css` are applied to all elements and components rendered in the React application.

- [ ] Open the `App.css` file in the `src` directory.
- [ ] Replace the provided starter code in `App.css` in the with the following code:

```
#root {
  max-width: 1280px;
  margin: 0 auto;
  padding: 2rem;
  text-align: center;
  font-family: Inter, Avenir, Helvetica, Arial, sans-serif;
  color: black;
  background-color: white;
}


 body {
  margin: 0;
  display: flex;
  place-items: center;
  min-width: 320px;
  min-height: 100vh;
}
```

- [ ] Remove all of the CSS style rules in `index.css`. We will add new CSS rules to that file later in this tutorial.

Now you have removed all of the elements and style rules from the starter template, your React application should be a blank page. (If you don't see a blank page, make sure you've saved your changes.) Let's add your first elements to your React application:

In the `div` in `App.jsx`

- [ ] Create an `h1` element with the title of your project.
- [ ] Create an `h2` element with a subtitle containing a message for your schedule recipient.

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

📍 **Checkpoint 1**: At this point in the lab, your app should look like this:

Step 2: Create the `Calendar` Component

In this step, you will create a `Calendar` component with tester code. You'll then import and add it to your web application.

- [ ] In the `src` directory, create a subdirectory called `components`.
- [ ] In the `components` directory, create a file called `Calendar.jsx`.

In `Calendar.jsx`:

- [ ] Import `React` from the `react` library.

```
import React from "react";
```

- [ ] To create a React functional component, define a function called `Calendar` using arrow function notation.

```
const Calendar = () => {

}
```

- [ ] Between the curly braces, write a return statement followed by a pair of parentheses.

```
return (

)
```

Between the parentheses, we'll write JSX statements to construct the component.

- [ ] Create a `div` where `className="Calendar"`
- [ ] Inside the `div`, write the text `"Testing the calendar component"`.
- [ ] At the end of the file, export the component by writing:

```
export default Calendar;
```

This exports the component so we can import it into our `App`.

In `App.jsx`:

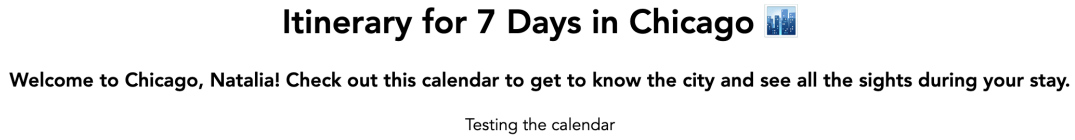
- [ ] Import the `Calendar` component.

```
import Calendar from './components/Calendar'
```

- [ ] After the `h2` element, add a `Calendar` component.

```
<Calendar />
```

📍 **Checkpoint 2**: In the browser, you should now see the test message `Testing the calendar` on your React application.



Step 3: Create a Grid in the `Calendar` Component

The `Calendar` component is a grid that will be used to construct your seven day timetable. The grid will store 1 hour blocks from Sunday through Saturday, 8am to 5pm. In this step, you will update a `Calendar` component with a grid.

In `Calendar.jsx`:

- [ ] Remove the test message `Testing the calendar`, but leave the `div`.
- [ ] Between the parentheses `()`, create a table with a table header.

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

▶ **Want to double-check your code? Compare what you have to this** 🔽

- [ ] In the table header, create a table row containing the following 8 table headings: `""`, `"Sunday"`, `"Monday"`, `"Tuesday"`, `"Wednesday"`, `"Thursday"`, `"Friday"`, `"Saturday"`.

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

▶ **Want to double-check your code? Compare what you have to this** 🔽

- [ ] After the table header, create a table body.

```
<table>
  ...
  <tbody>
  </tbody>
</table>
```

- [ ] Inside the table body, create 10 table rows, each with 8 table data cells. To do so, copy and paste the following code snippet 10 times.

```
<tr>
  <td className="time">Insert Time</td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
</tr>
```

This code represents a row in the timetable. Each row in the timetable will represent a 1 hour block for the week. The first cell in each row, `<td className="time">`, will store the hour (e.g., 8 am). The remaining 7 cells will store any events for each day of the week.

- [ ] For each copy, replace the `Insert Time` with the following hour blocks: `8 am`, `9 am`, `10 am`, `11 am`, `12 pm`, `1 pm`, `2 pm`, `3 pm`, `4 pm`,and `5 pm`. **Note:** You may choose to use a different time spread or include additional hours (if you're planning a vacation, for example).

In `index.css`:

- [ ] Add the following code snippet to create more white space in the table:

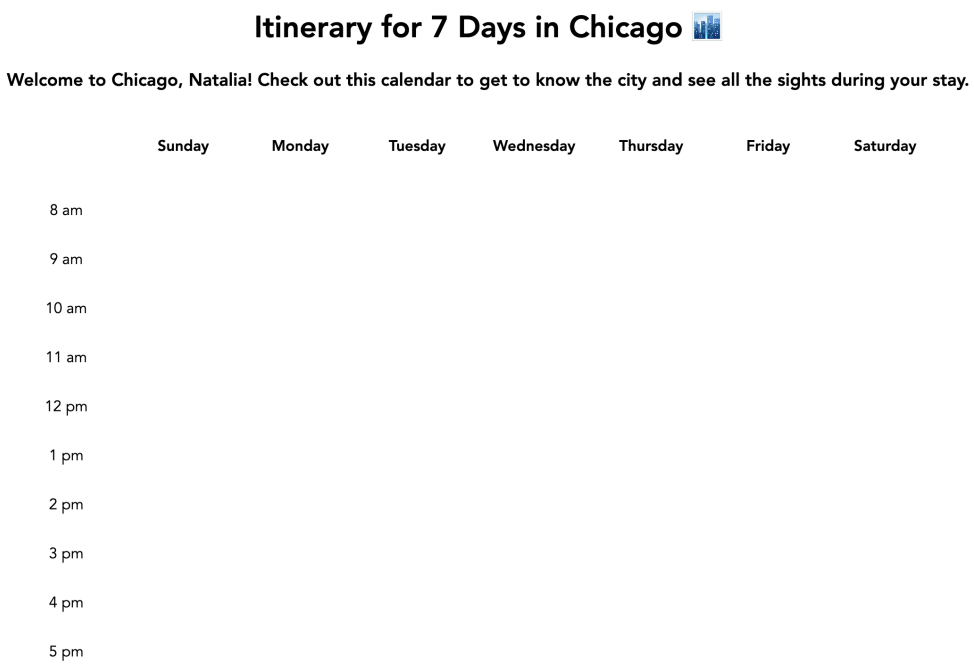
```
body {
  margin: 0;
  display: flex;
  place-items: center;
  min-width: 320px;
  min-height: 100vh;
}

th {
  width: 120px;
  padding-top:25px;
  padding-bottom:30px;
}

td {
  padding: 5px;
}

.time {
  height: 40px;
}
```

📍 **Checkpoint 3**: At this point in the lab, your app should look like this:



Step 4: Create the `Event` Component

In this step, you will create an `Event` component. The `Event` component will be a table cell but customized with information related to an event, such as the location.

- [ ] In the `components` directory, create a file called `Event.jsx`.

In `Event.jsx`:

- [ ] Import `React` from the `react` library

```
import React from "react";
```

- [ ] Create a React functional component using arrow function notation called `Event`. It should have a single parameter called `props`.

```
const Event = (props) => {

}
```

- [ ] Between the curly braces, write a return statement followed by a pair of parentheses.

```
return (

)
```

tween the parentheses, we'll write JSX statements to construct the component.

- [ ] Create a `td` element where `className="Event"`.
- [ ] Inside the `td` element, create an `h5` element with the text `Test Event Name`.

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

- [ ] At the end of the file, export the component by writing:

```
export default Event;
```

This exports the component so we can import it into our `Calendar`.

In `Calendar.jsx`:

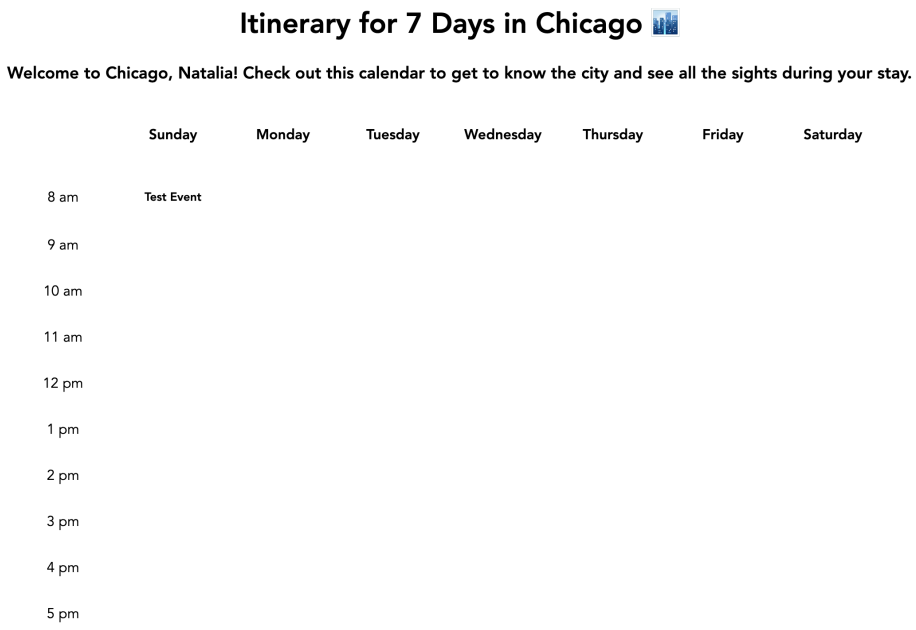
- [ ] Import the `Event` component.

```
import Event from './Event'
```

- [ ] Replace one of the `td` elements in the table with an `Event` component. For example:

```
<tr>
  <td className="time">8 am</td>
  <Event />
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
</tr>
```

📍 **Checkpoint 4**: In the browser, you should now see the test event on the Calendar React application:



Step 5: Pass props to the `Event` Component

In this step, you will pass props to the `Event` component with the location and the color.

In the `Event` component you added to the `Calendar`:

- [ ] Create an `event` prop and set it to `'Fancy Dinner 🎩'`

```
<Event event='Fancy Dinner 🎩' />
```

- [ ] Create `color` prop and set it to `'green'`

```
<Event event='Fancy Dinner 🎩' color ='green'/>
```

In `Event.jsx`:

- [ ] Replace the text within the `h5` element with `{props.event}`. The curly braces allow for JavaScript code to be evaluated within the HTML. Therefore, the event string that was passed to the `Event` component will replace the test text.

```
<h5>{props.event}</h5>
```

▶ 🤖 **AI Pro Tip: Using AI to Explore New Concepts**

- [ ] We can also use the props to set CSS attributes. Replace the `className` to `{'Event ' + props.color}`. This adds an additional `className` to the element that we can dynamically set the color of elements based on the color that you pass as a prop.

```
<td className={'Event ' + props.color}>
```

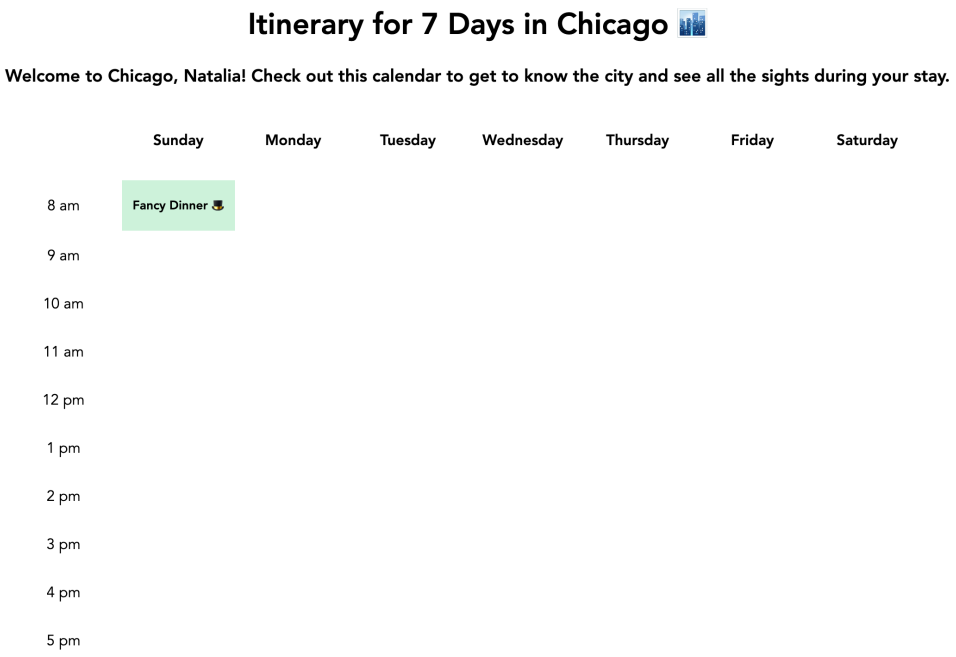
In `index.css`:

- [ ] Create rules to style the `Event` components and set the background for the different colors that will be passed as props.

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

▶ **Curious what the exemplar uses? Check out our styles here** 🔽

📍 **Checkpoint 5**: In the browser, you should now see the event with the event title and and green background on the Calendar React application:



Step 6: Add More Events

In this step, you will add more events to the calendar.

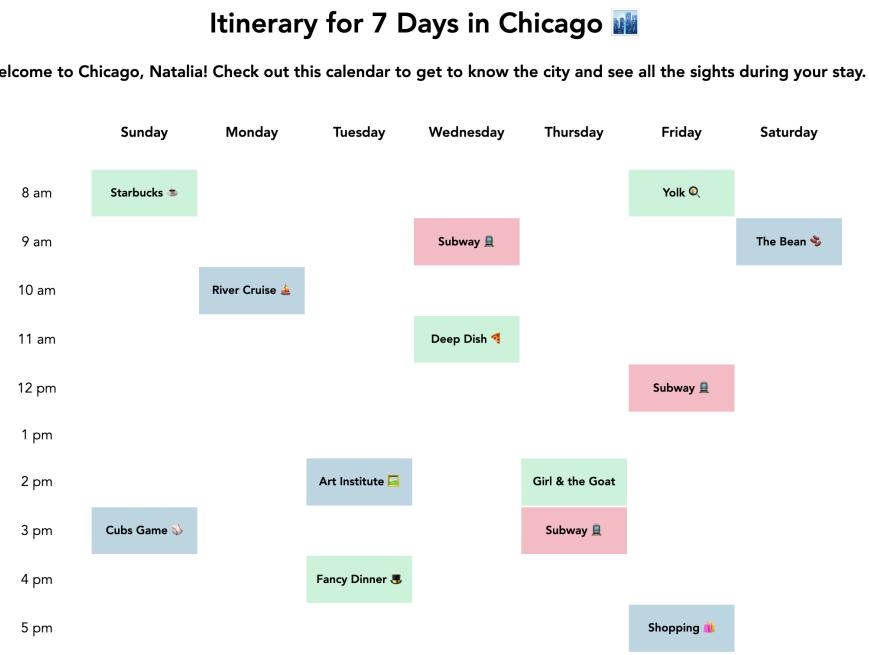
- [ ] To add an event to the calendar, replace one of the `td` elements in the table body with an `Event` component. Be sure to pass a value for `event` and a `color`. Currently, we only have CSS rules for `green`, `blue`, and `pink`. For example:

```
<tr>
  <td className="time">8 am</td>
  <Event event='Starbucks ☕' color ='green'/>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <Event event='Yolk 🍳' color ='green'/>
  <td></td>
</tr>

<tr>
  <td className="time">9 am</td>
  <td></td>
  <td></td>
  <td></td>
  <Event event='Subway 🚇' color ='pink'/>
  <td></td>
  <td></td>
  <Event event='The Bean 🫘' color ='blue'/>
</tr>
...
```

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

🎈 **Checkpoint 6**: In the browser, you should now the events you added to the calendar with the event title and and background:



🎉 Congratulations, you've completed your first lab! 🎉

If you have time left over, continue on to the stretch features to customize and improve your app!

Stretch Features

Step 7: Add the Location to Event

In this step, you will add pass another prop to the Event component with the location of the event.

In the `Event` component you added to the Calendar:

- [ ] Create a `location` prop and set it to string with the location name or address. For example:

```
<Event event='Fancy Dinner 🎩' location='Maple & Ash'/>
```

In `Event.jsx`:

- [ ] Add the `{props.location}` in an `h6` element.

▶ 🤖 **AI Pro Tip: Using AI as a Programming Assistant**

📍 **Checkpoint 6**: In the browser, you should now see the events you added to the calendar with the event title and and background:

🎉 Congratulations 🎉

You've completed your first lab AND stretch goals! 🚀

💡 **Tip:** Remember to come back and reference this lab when you need to do similar things in your project!