

# WEB102 | Intermediate Web Development

Intermediate Web Development Summer 2025 (@ Section 1 | Wednesdays 4PM - 6PM PDT)  
Personal Member ID#: 107789

## Week 1: 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

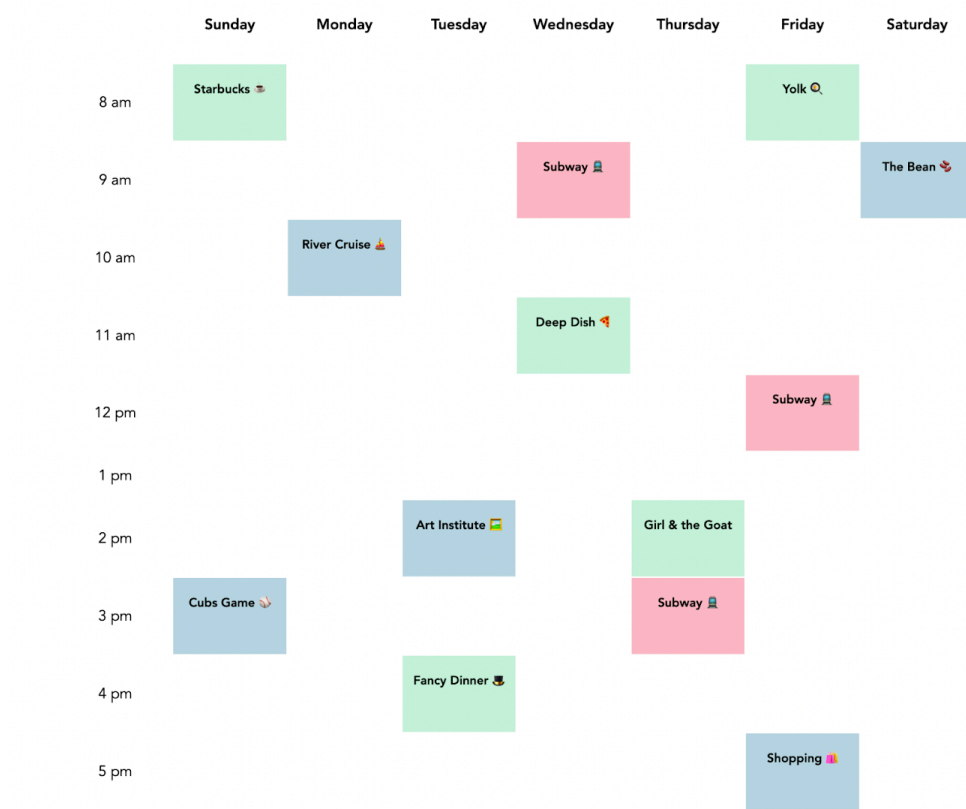
*View an exemplar of what you'll be creating in this lab [here!](#)*

### Required Features

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

## Itinerary for 7 Days in Chicago 🏙️

One week in Chicago is wonderful. If want to get to know the city and see all the sights, then spending 7 days in Chicago is perfect!



## Stretch Features

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

## Itinerary for 7 Days in Chicago 🏙️

One week in Chicago is wonderful. If want to get to know the city and see all the sights, then spending 7 days in Chicago is perfect!

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8 am	Starbucks ☕ 646 Michigan Ave					Yolk 🥚 355 E Ohio St	
9 am				Subway 🚇 Grand Station			The Bean 🍷 Millennium Park
10 am		River Cruise 🚢 Chicago River					
11 am				Deep Dish 🍕 Giodano's			
12 pm						Subway 🚇 Washington Station	
1 pm							
2 pm			Art Institute 🖼️ 111 S Michigan Ave		Girl & the Goat 🐐 809 W Randolph St		
3 pm	Cubs Game 🏈 Wrigley Field				Subway 🚇 Roosevelt Station		
4 pm			Fancy Dinner 🍽️ Maple & Ash				
5 pm						Shopping 🛍️ Magnificent Mile	

## Resources

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

## Lab Instructions

### Getting Started

👉 Before proceeding, make sure you've set up your development environment by following the ✨  
**IDE Setup Guide!**

## Required Features

### Step 0: Create a New React Project Using Vite

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 Opportunity

### ► Use AI to navigate a new IDE → The VS Code Terminal

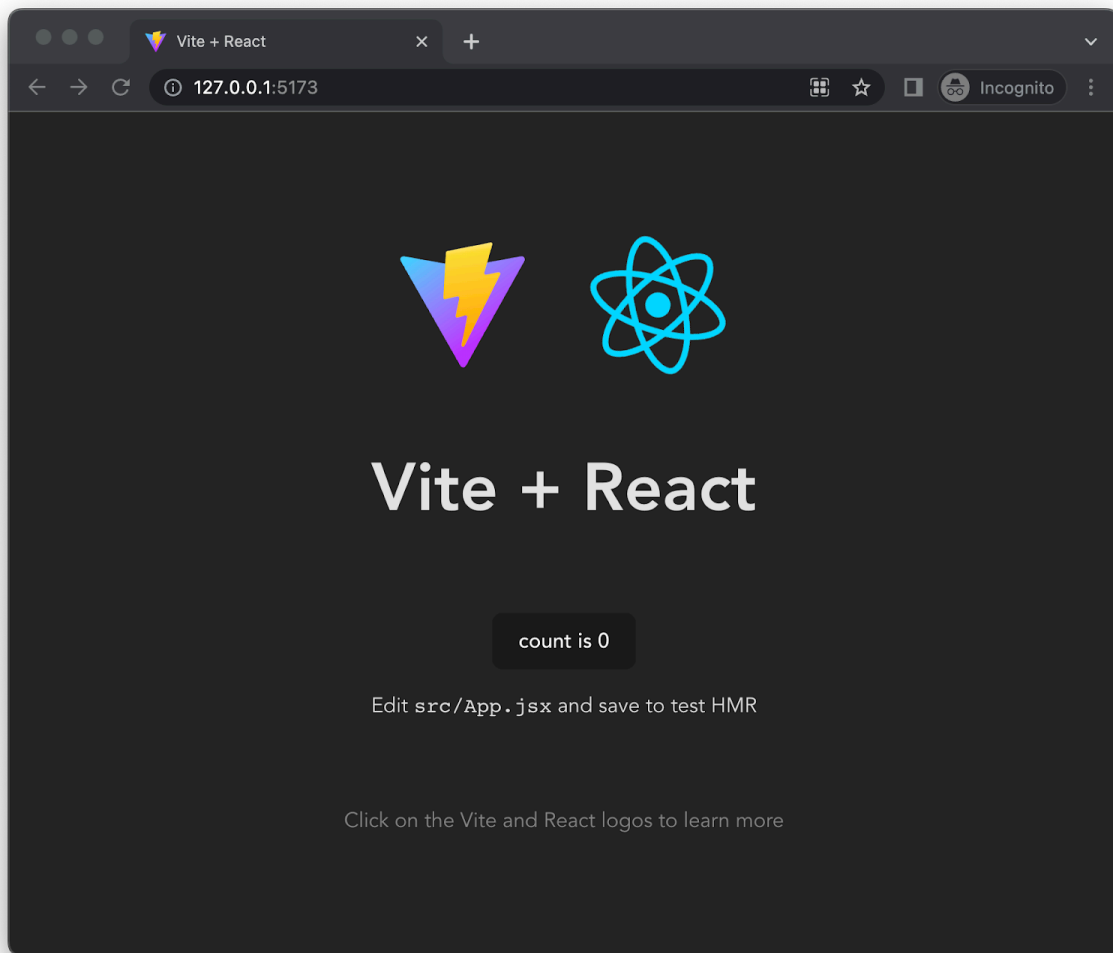
- ☐ Make sure you have added the folder to your VS Code workspace. If you don't see the folder in the Explorer, right-click in the pane, select `Add Folder to Workspace...` , and then add your folder.
- ☐ In your Github repository, initialize a new React project using Vite:
  - ☐ In the Terminal, enter 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>
    )
  }

export 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 Opportunity

► *Use AI to understand provided code* → "Explain this" with Copilot

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 Opportunity

► *Use AI as a pair programming partner* → Adding a title and subtitle

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

### Itinerary for 7 Days in Chicago 🏙️

Welcome to Chicago, Natalia! Check out this calendar to get to know the city and see all the sights during your stay.

## 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.



## Itinerary for 7 Days in Chicago 🏙️

Welcome to Chicago, Natalia! Check out this calendar to get to know the city and see all the sights during your stay.

Testing the calendar

### 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 Opportunity


► *Use AI as a pair programming partner* → Creating an HTML table

► **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 Opportunity

► *Use AI as a pair programming partner* → Adding table headings

► 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:

## Itinerary for 7 Days in Chicago 🏙️

Welcome to Chicago, Natalia! Check out this calendar to get to know the city and see all the sights during your stay.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8 am							
9 am							
10 am							
11 am							
12 pm							
1 pm							
2 pm							
3 pm							
4 pm							
5 pm							

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 (  
  
)
```

Between 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 Opportunity

► *Use AI as a pair programming partner* → Creating a table cell with an event name

- ☐ 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>
  <td></td>
</tr>
```

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

Itinerary for 7 Days in Chicago 🏙️							
Welcome to Chicago, Natalia! Check out this calendar to get to know the city and see all the sights during your stay.							
	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8 am	Test Event						
9 am							
10 am							
11 am							
12 pm							
1 pm							
2 pm							
3 pm							
4 pm							
5 pm							

## 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 Opportunity

► *Use AI to understand provided code* → "Explain this" with Copilot

- ☐ 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 Opportunity

► *Use AI to brainstorm ideas for your code* → Event styles

► **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:

## Itinerary for 7 Days in Chicago 🏙️

Welcome to Chicago, Natalia! Check out this calendar to get to know the city and see all the sights during your stay.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8 am	Fancy Dinner 🍽️						
9 am							
10 am							
11 am							
12 pm							
1 pm							
2 pm							
3 pm							
4 pm							
5 pm							

## 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 Opportunity

► *Use AI as a pair programming partner* → Speeding up repetitive tasks

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

### Itinerary for 7 Days in Chicago 🏙️

Welcome to Chicago, Natalia! Check out this calendar to get to know the city and see all the sights during your stay.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8 am	Starbucks ☕					Yolk 🐔	
9 am				Subway 🚇			The Bean 🌱
10 am		River Cruise 🚢					
11 am				Deep Dish 🍕			
12 pm						Subway 🚇	
1 pm							
2 pm			Art Institute 🎨		Girl & the Goat		
3 pm	Cubs Game 🏟️				Subway 🚇		
4 pm			Fancy Dinner 🍽️				
5 pm						Shopping 🛍️	



🎉 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 Opportunity

► *Use AI as a pair programming partner* → Adding the location

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

# Itinerary for 7 Days in Chicago 🏙️

One week in Chicago is wonderful. If want to get to know the city and see all the sights, then spending 7 days in Chicago is perfect!

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8 am	Starbucks ☕ 646 Michigan Ave					Yolk 🥚 355 E Ohio St	
9 am				Subway 🚇 Grand Station			The Bean 🍷 Millennium Park
10 am		River Cruise 🚢 Chicago River					
11 am				Deep Dish 🍕 Giodano's			
12 pm						Subway 🚇 Washington Station	
1 pm							
2 pm			Art Institute 🖼️ 111 S Michigan Ave		Girl & the Goat 🐐 809 W Randolph St		
3 pm	Cubs Game 🏈 Wrigley Field				Subway 🚇 Roosevelt Station		
4 pm			Fancy Dinner 🍽️ Maple & Ash				
5 pm						Shopping 🛍️ Magnificent Mile	

🎉 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!