实践项目：会议活动策划



预计所需时间: 90分钟

理解任务

Alejandre 管理一个商务会议的场地。她的母公司“BudgetEase”希望雇佣你开发一个网站，以便 BudgetEase 的客户能够轻松定价他们的会议活动。

该应用程序的需求包括允许用户选择和定价会议中心的房间、附加选项，如麦克风和投影仪，以及为特定数量的客人提供餐饮服务。

BudgetEase 会议费用规划器的功能将包括：

- 一个动态用户界面，实时根据用户选择进行更新
- 场地选择、附加选项和餐饮选择的组件
- 使用 Redux Toolkit 进行 Redux 集成，以管理状态变化
- Redux 切片以管理不同部分的状态
- 在弹出窗口中显示所选项目及其成本的表格
- 根据用户选择计算并显示小计和总成本

学习目标

完成此实验后，您将能够：

- **创建 React 组件**：使用组件组合和嵌套创建功能性 React 组件。
- **使用钩子管理状态**：实现 React 钩子，特别是 useState 和 useEffect 钩子。您将使用钩子来管理组件级状态和控制元素的可见性。
- **集成 Redux**：使用 Redux 概念（如动作、减速器和存储）在应用程序中集成 Redux。
- **渲染动态数据**：将从对象数组中获取的数据动态渲染到用户界面。您将遍历数组以生成组件列表。
- **使用条件渲染处理事件**：处理用户事件，例如按钮选择，并触发相应的动作。

项目任务

1. 设置项目环境
2. 审查 `ConferenceEvent.jsx` 组件的结构
3. 审查场地模块的代码
4. 将 Redux 与组件结合，以管理更新和状态变化
5. 添加逻辑以计算小计和总成本
6. 制作一个动态表格以显示所选产品，显示项目名称、单价、数量和该项目的总体成本
7. 创建一个舒适用户体验的网页设计
8. 将您的网站部署到公共托管服务

解决方案

您将在本实验结束时找到解决方案代码。如果您在完成任何任务时需要帮助，可以在那里找到建议的工作代码版本。同时，请确保保存您的解决方案或本实验结束时的代码。这将帮助您开发最终项目的代码。

先决条件

- 基础 HTML 和 CSS
- 中级 JavaScript
- 熟悉 React 函数组件、钩子和用于状态管理的 Redux Toolkit
- 使用 GitHub 进行代码管理

如果您需要关于如何在 GitHub 上工作的指导，请查看 这些说明。

关于此实验环境的重要通知

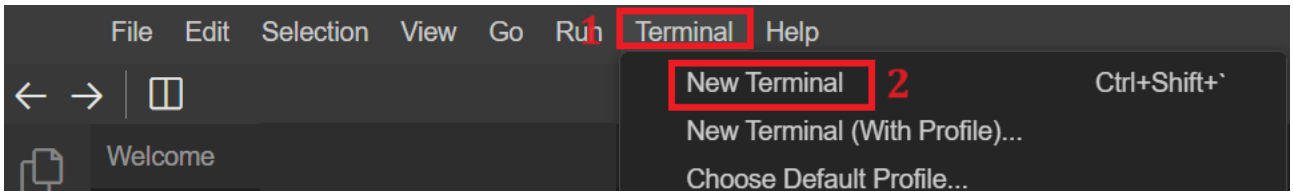
Skills Network Cloud IDE（基于 Theia 和 Docker）是一个开源的集成开发环境（IDE），为课程和项目相关的实验提供了一个动手实验的环境。

请注意，此实验环境的会话不是持久的。每次您连接到此实验时，都会为您创建一个新的环境。如果您在未保存到 GitHub 或其他外部源的情况下退出环境，您将丢失数据。请计划在单个会话中完成这些实验，以避免数据丢失。

任务 1：设置环境

1. 创建一个公共的 GitHub 仓库。

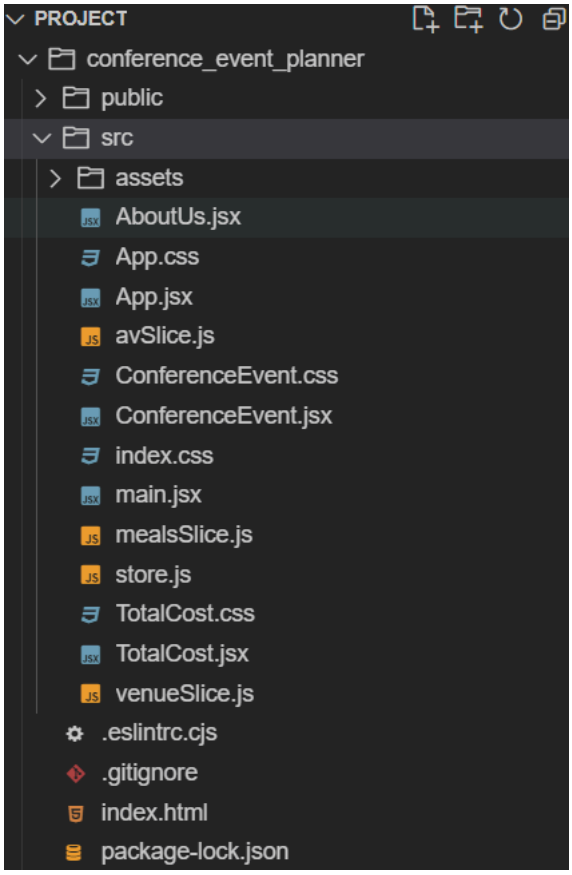
2. 在 IDE 顶部的 **终端** 菜单中，选择 **新建终端**。



3. 在终端中，输入以下命令以克隆此 React 应用程序的模板。按 **Enter**。

```
git clone https://github.com/ibm-developer-skills-network/conference_event_planner.git
```

上述命令将在 "Project" 文件夹下创建一个名为 "conference_event_planner" 的文件夹。截图显示了目录结构。



4. 您可以为项目寻找自己的图片，或者使用来自 Pixabay 的这些推荐图片，Pixabay 是一个提供免版税图片的网站。

会议室
https://pixabay.com/photos/chairs-empty-office-room-table-2181916/

礼堂：
https://pixabay.com/photos/event-venue-auditorium-meeting-1597531/

演示室：
https://pixabay.com/photos/convention-center-chair-seminar-3908238/

会议室：
https://pixabay.com/photos/chairs-empty-office-room-table-2181916/

小会议室：
https://pixabay.com/photos/laptops-meeting-businessmen-593296/

投影仪：
https://pixabay.com/photos/business-computer-conference-20031/

扬声器：
https://pixabay.com/photos/speakers-bluetooth-tech-speaker-4109274/

麦克风：
https://pixabay.com/photos/public-speaking-mic-microphone-3926344/

白板：
https://pixabay.com/photos/whiteboard-dry-erase-marker-blank-2903269/

标志：
https://pixabay.com/photos/signpost-waypoint-wood-grain-board-235079/

5. 使用以下命令将您的终端路径更改为 **conference_event_planner** 文件夹。

```
cd conference_event_planner
```

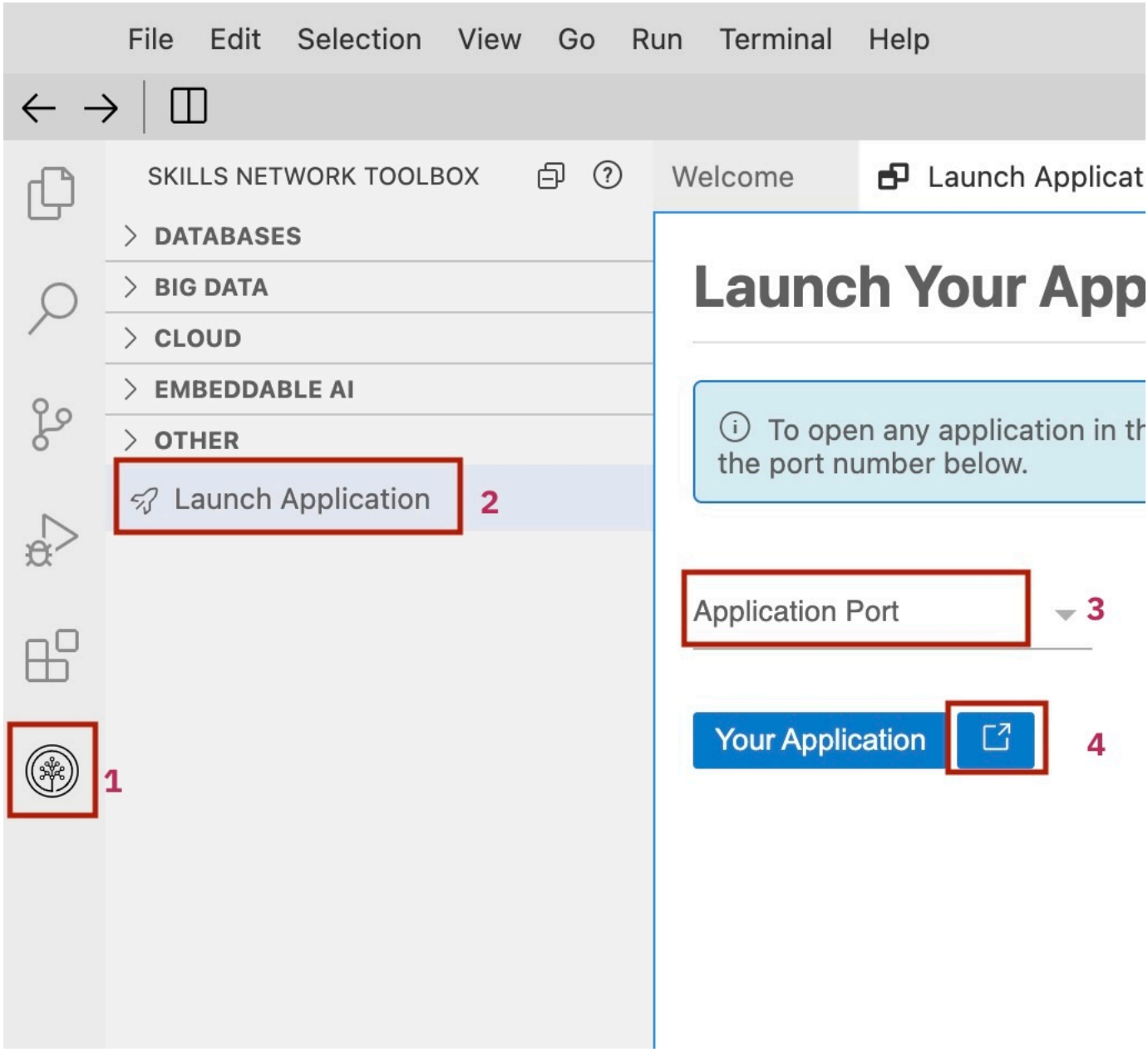
6. 确保你克隆的代码能够正常工作。
使用 npm 安装执行应用程序所需的包。

```
npm install
```

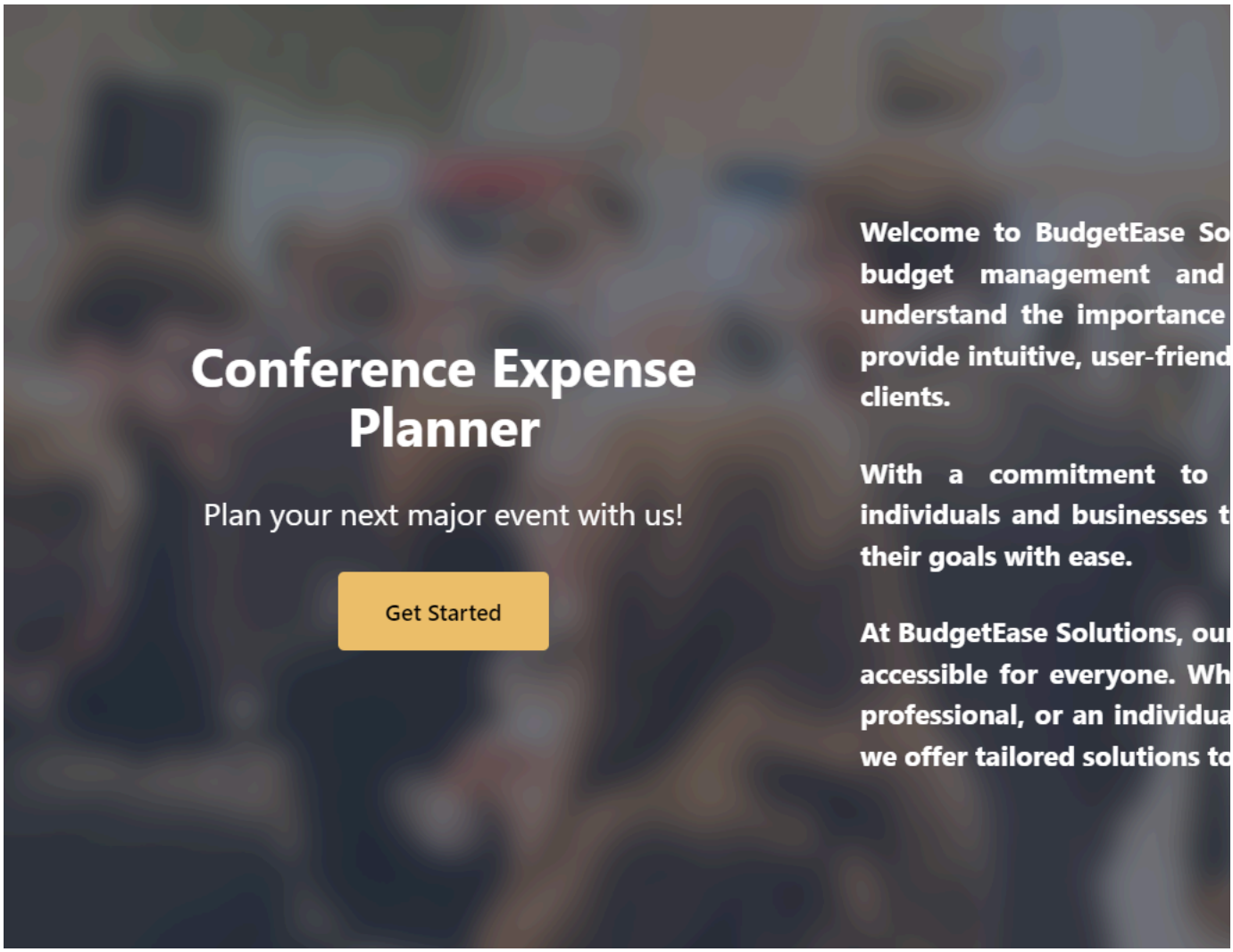
执行此命令以运行应用程序，这将启动在端口号 4173 上运行的应用程序服务器。

```
npm run preview
```

7. 现在，您可以查看应用程序。
点击左侧面板上的 **Skills Network** 图标（参见数字 1）。此操作将打开 **Skills Network Toolbox**。接下来，点击 **Launch Application**（参见数字 2）。在 **Application Port** 中输入端口号 **4173**（参见数字 3），然后点击 。

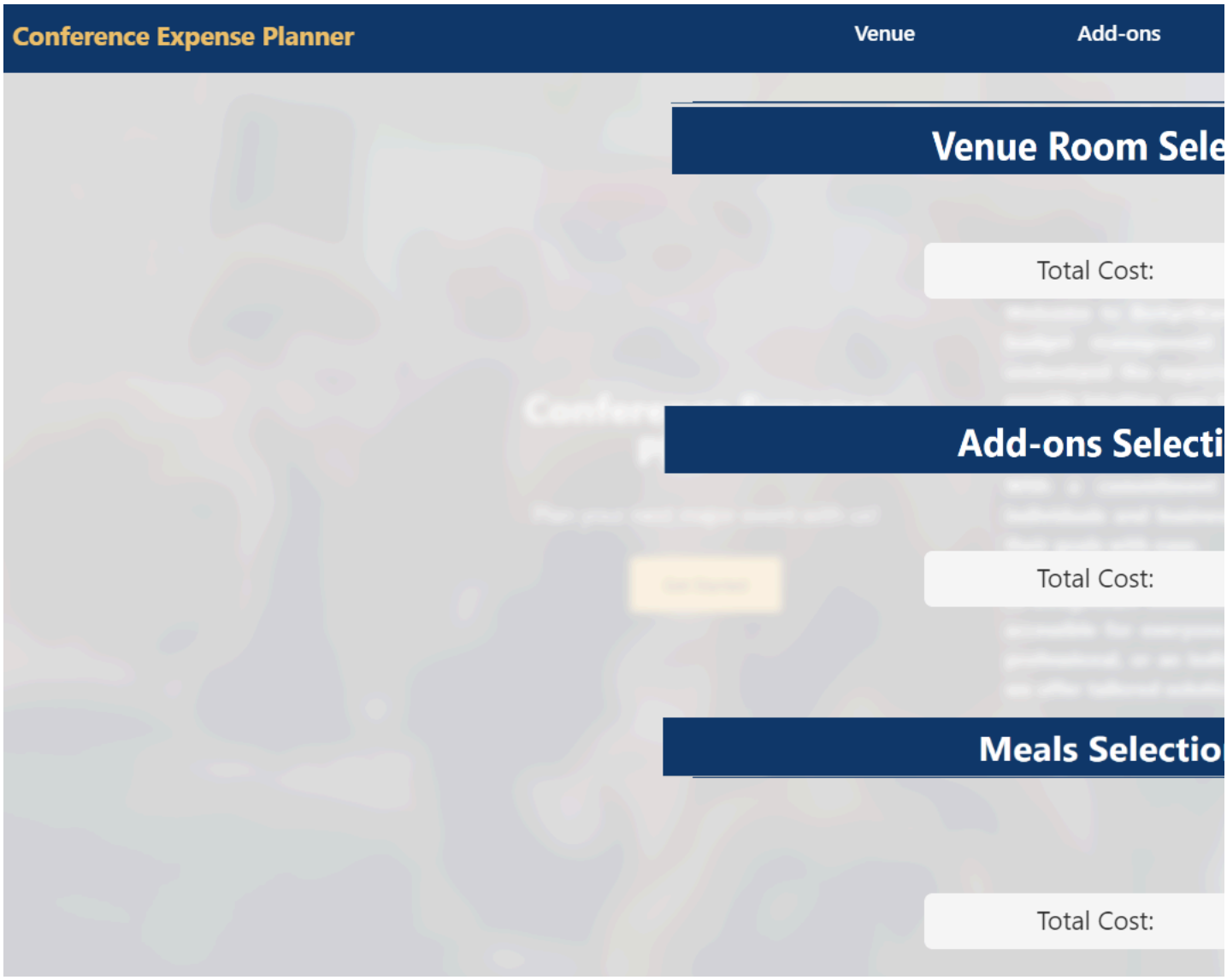


8. 输出应类似于以下屏幕截图。点击 **Get Started** 按钮应将用户带到产品选择页面。标题和第一部分为“房间选择”。



*注意：您已被提供 `backgroundColor` 而不是 `backgroundImage`。如果您更喜欢图像而不是颜色，可以使用您自己的图像。

使用给定代码的应用程序产品页面应类似于以下内容：



10. 现在保存所有代码。如果您进行了任何更改，请执行 `git add` 和 `git commit`。然后，执行 `git push` 命令将已提交的更改上传到您的远程 GitHub 仓库，以确保您保存了代码的最新版本。

注意：在将代码推送到您的 GitHub 仓库时，可能会要求您提供用户名和密码。如果您需要更多关于如何在 GitHub 中工作的指导，请查看 这些说明。

任务 2：审查 ConferenceEvent.jsx 结构

您有两个工作模块：

- 包含开始按钮和公司描述的登录页面。
- “场地部分”包括场地中的房间选择以及增减按钮。

打开 `src` 目录中的 `ConferenceEvent.jsx` 文件。该组件包括产品选择页面的函数和布局。

布局

- 一个导航栏元素
- 在 `<div>` 标签中的 `main_container` 类
- 一个 `showItems` 变量，用于使用条件运算符 `?` `:` 切换功能
- 在 `<div>` 标签中的 `items-information` 类
- 在 `<div>` 标签中的 `venue`、`addons` 和 `meals` 布局。每个布局都有两个公共类名，`venue_container` 和 `container_main`
- 在 `<div>` 标签中的 `total_amount_detail` 类，用于显示所选项目的详细信息

任务 3：审查场地模块代码

接下来，您将逐步了解提供给您的场地房间选择功能的代码。重点关注状态流以及 `ConferenceEvent` 组件如何与 `venueSlice` 协同工作。场地房间选择功能的代码分布在三个文件中：`venueSlice.js`、`ConferenceEvent.jsx` 和 `store.js`。让我们讨论这些代码及它们之间的交互。

场地功能使用 Redux 工具包中的 **slices**，通过导入 `createSlice()` 函数来实现。一个 slice 将您的应用程序状态分解为更小的特性，有助于组织您的代码，使其更易于阅读和维护。

venueSlice.js：

让我们逐步了解 `venueSlice.js` 文件中的代码。它包含使用 @reduxjs/toolkit 中的 `createSlice` 切分与场地选择相关的 Redux 状态的代码。

1. 初始状态由一个场地对象数组组成，每个对象代表一个可租用的房间。场地对象具有缩略图、名称、费用和数量等属性。
2. `venueSlice.js` 文件包含 reducer 函数 `incrementQuantity` 和 `decrementQuantity`，用于管理状态中场地项目的数量。
    - 注意：在本实验的最后一页，我们为您提供了来自 Pixabay 的图像链接，并附上适当的引用，或者您可以找到自己的图像。
3. incrementQuantity():
    - 此函数处理状态中场地项目数量的增加。它接收一个包含要 **增加** 的项目索引的 action。
    - 它首先检查在提供的索引处状态中是否存在该项目。如果项目存在且是一个 Auditorium Hall 且数量大于或等于 3，则提前返回而不修改状态。
    - 否则，它将该项目的数量增加一个。
4. decrementQuantity():
    - 此函数处理状态中场地项目数量的减少。它接收一个包含要 **减少** 的项目索引的 action。
    - 它首先检查在提供的索引处状态中是否存在该项目以及其数量是否大于 0。
    - 如果两个条件 **都满足**，该项目的数量将减少一个。

```
reducers: {
    incrementQuantity: (state, action) => {
      const { payload: index } = action;
      if (state[index]) {
        if (state[index].name === "Auditorium Hall (Capacity:200)" && state[index].quantity >= 3) {
          return;        }
        state[index].quantity++;
      }
    },
    decrementQuantity: (state, action) => {
      const { payload: index } = action;
      if (state[index] && state[index].quantity > 0) {
        state[index].quantity--;
      }
    },
  },
```

Redux Store 设置

现在我们将查看 `store.js` 文件。

1. 使用 `@reduxjs/toolkit` 中的 `configureStore()` 函数创建 Redux store。
2. `store.js` 文件包含一个名为 `venue()` 的 reducer，从 `venueSlice.js` 导入。

```
import { configureStore } from '@reduxjs/toolkit';
import venueReducer from './venueSlice';
export default configureStore({
    reducer: {
        venue: venueReducer,
    },
});
```

3. 这段代码使用 `@reduxjs/toolkit` 的 `configureStore()` 函数创建一个全局 Redux 存储，以便应用中的所有组件都可以访问由 `venueReducer()` 管理的状态。

ConferenceEvent 组件

现在让我们逐步查看 `ConferenceEvent.jsx` 文件中的相关代码。

1. 首先，导入所需的依赖项。

```
import { useSelector, useDispatch } from "react-redux";
import { incrementQuantity, decrementQuantity } from "./venueSlice";
```

2. `useSelector()` 函数从 Redux 存储状态中检索场地项目。

```
const venueItems = useSelector((state) => state.venue);
```

3. 然后它定义了事件处理程序，如 `handleAddToCart()` 和 `handleRemoveFromCart()`，用于管理用户交互中的数量增加和减少。

```
const handleAddToCart = (index) => {
    if (venueItems[index].name === "Auditorium Hall (Capacity:200)" && venueItems[index].quantity >= 3) {
        return; // Prevent further additions
    }
    dispatch(incrementQuantity(index));
};
const handleRemoveFromCart = (index) => {
    if (venueItems[index].quantity > 0) {
        dispatch(decrementQuantity(index));
    }
};
```

4. 接下来，它将可用的礼堂数量计算为三，因此用户不能请求超过三个。

```
const remainingAuditoriumQuantity = 3 - venueItems.find(item => item.name === "Auditorium Hall (Capacity:200)").quantity;
```

5. 当用户选择增加或减少房间数量的按钮时，系统应计算所有选定房间的费用。

6. 为此，我们定义一个 `calculateTotalCost()` 函数，并声明一个 `venueTotalCost`。

```
const calculateTotalCost = (section) => {
    let totalCost = 0;
    if (section === "venue") {
        venueItems.forEach((item) => {
            totalCost += item.cost * item.quantity;
        });
    }
    return totalCost;
};
const venueTotalCost = calculateTotalCost("venue");
```

7. 函数声明：

- 该函数使用箭头函数语法定义，并赋值给常量 `calculateTotalCost`。
- 它接受一个字符串参数 `section`，表示计算的部分 **被计算**。

8. `totalCost` 的初始化：

- `totalCost` 变量初始化为 0。该值将保存指定部分的累计总成本。

9. 条件检查：

- 函数检查作为参数传递的部分是否等于字符串“venue”。
- 如果为真，**将计算场地项目的总成本**。

10. 遍历 `venueItems`：

- `venueItems` 数组表示具有 `cost` 和 `quantity` 属性的项目。
- `forEach` 函数遍历 `venueItems` 数组中的每个项目。对于每个项目，它将 `item.cost` 乘以 `item.quantity` 并将结果加到 `totalCost` 中。

11. 返回语句：

- 循环完成后，函数返回计算出的 `totalCost`。

12. 函数调用：

- 调用函数 `calculateTotalCost`，参数为“venue”，计算“venue”部分项目的总成本。
- 此计算的结果存储在常量 `venueTotalCost` 中。

13. 最后，组件显示所有选定场地项目的总成本。

```
<div className="total_cost">Total Cost: ${venueTotalCost}</div>
```

任务 4：为附加组件编写代码

现在轮到你编写一些代码了。在本节中，你将创建附加组件部分。首先，你需要在 **src** 文件夹下的 `avSlice.js` 中创建逻辑。

1. 用对象初始化 `initialState` 数组变量，以提供数据结构。请注意，你需要在下面的数据中添加自己的图片及其适当的路径。示例图片的链接在 **任务 1：设置环境** 中提供。将以下代码包含在 `avSlice.js` 的 `initialState` 中。

```
    {
        img: "https://pixabay.com/images/download/business-20031_640.jpg",
        name: "Projectors",
        cost: 200,
        quantity: 0,
    },
    {
        img: "https://pixabay.com/images/download/speakers-4109274_640.jpg",
        name: "Speaker",
        cost: 35,
        quantity: 0,
    },
    {
        img: "https://pixabay.com/images/download/public-speaking-3926344_640.jpg",
        name: "Microphones",
        cost: 45,
        quantity: 0,
    },
    {
        img: "https://pixabay.com/images/download/whiteboard-2903269_640.png",
        name: "Whiteboards",
        cost: 80,
        quantity: 0,
    },
    {
        img: "https://pixabay.com/images/download/signpost-235079_640.jpg",
        name: "Signage",
        cost: 80,
        quantity: 0,
    },
```

增加和减少

现在，您需要为 `incrementAvQuantity()` 和 `decrementAvQuantity()` 函数创建逻辑。

1. `incrementAvQuantity()` reducer 函数增加状态中特定项目的数量。
2. 它接受两个参数：`state` 和 `action`。
3. `action.payload` 对象包含要增加的项目的标识符。
4. reducer 使用 `state[action.payload]` 从状态中检索该项目。
5. 如果该项目存在，它将其数量属性增加 1。

```
incrementAvQuantity: (state, action) => {
    const item = state[action.payload];
    if (item) {
        item.quantity++;
    }
},
```

6. `decrementAvQuantity()` reducer 函数减少状态中特定项目的数量。
7. 与 `incrementAvQuantity()` 类似，它接受两个参数：`state` 和 `action`。
8. `action.payload` 对象包含要减少的项目标识符。– 它的 reducer 使用 `state[action.payload]` 从状态中检索该项目。
9. 如果项目存在且数量大于 0，它将其数量属性减少 1，确保数量不低于 0，并表示没有更多可用项目。

```
decrementAvQuantity: (state, action) => {
    const item = state[action.payload];
    if (item && item.quantity > 0) {
        item.quantity--;
    }
},
```

10. 导出 `avslice.js` 中的所有 reducer 函数和动作。

11. 将 `avSlice` 提供给 `store.js` 文件。导航到 `store.js` 文件并导入 `avSlice.js`。文件结构应如下所示。

```
import { configureStore } from '@reduxjs/toolkit';
import venueReducer from './venueSlice';
import avReducer from './avSlice';
export default configureStore({
  reducer: {
    venue: venueReducer,
    av: avReducer,
  },
});
```

12. 现在您需要显示在 `avSlice.js` 中初始化的附加组件的详细信息。

13. 打开 `ConferenceEvent.jsx` 组件。从 `store.js` 中导入附加组件项到 `ConferenceEvent.jsx` 组件，`avSlice.js` 通过 reducers 发送详细信息。

```
  const avItems = useSelector((state) => state.av);
```

注意：在 `ConferenceEvent.jsx` 组件中，将上述代码包含在 `const venueItems = useSelector((state) => state.venue);` 之后。

14. 使用 `map()` 方法显示 `avItems` 变量中的项目。同时包括增加和减少按钮，以便每次增加或减少数量1。还要将代码包含在一个 class 名称为 `addons_selection` 的 `<div>` 标签中，如以下代码所示。您可以在代码片段之后阅读代码说明。

```
{avItems.map((item, index) => (
    <div className="av_data venue_main" key={index}>
        <div className="img">
            <img src={item.img} alt={item.name} />
        </div>
    <div className="text"> {item.name} </div>
    <div> ${item.cost} </div>
        <div className="addons_btn">
            <button className="btn-warning" onClick={() => handleDecrementAvQuantity(index)}> &ndash; </button>
            <span className="quantity-value">{item.quantity}</span>
            <button className=" btn-success" onClick={() => handleIncrementAvQuantity(index)}> &#43; </button>
        </div>
    </div>
))}
```

15. 这段代码使用 `map()` 函数遍历一个名为 `avItems` 的数组，该数组包含有关视听项目的信息。让我们回顾一下代码的工作原理。

16. 数组中的每个项目都会创建一个带有类 `av_data` 和 `venue_main` 的 `<div>` 元素。

17. 在这个 `<div>` 内，包含：

    - 一个 `<img>` 标签显示项目的图片。图片源 (src) **来自** `item.img`，替代文本设置为 `item.name`。
    - 一个 `<div>` 显示项目的名称 (item.name)。
    - 另一个 `<div>` 显示项目的成本 `item.cost`，以美元计。
    - 一组按钮被包装在一个类为 `addons_btn` 的 `<div>` 中。这些按钮允许用户调整项目的数量。共有两个按钮：
        - 第一个按钮，类为 `btn-warning`，标签为 n-dash (–)。选择时会减少数量。其点击事件绑定到函数 `handleDecrementAvQuantity()`，并将项目的索引作为参数。
        - 第二个按钮，类为 `btn-success`，标签为加号 (+)。选择时会增加数量。其点击事件绑定到函数 `handleIncrementAvQuantity()`，并将项目的索引作为参数。

18. 项目当前的数量显示在减少和增加按钮之间。这个值 **来自** `item.quantity` 对象。

19. 接下来，您需要分发操作。对于函数 `handleDecrementAvQuantity()` 和 `handleIncrementAvQuantity()`，在返回组件之前在各自的函数中包含以下代码。

```
const handleIncrementAvQuantity = (index) => {
    dispatch(incrementAvQuantity(index));
};
const handleDecrementAvQuantity = (index) => {
```

```
    dispatch(decrementAvQuantity(index));
};
```

20. 要实现上述代码，请确保从 `avSlice` 导入 `incrementAvQuantity` 和 `decrementAvQuantity`。将上述代码包含在 `ConferenceEvent.jsx` 组件的顶部。

```
import { incrementAvQuantity, decrementAvQuantity } from "./avSlice";
```

21. 现在，在 `calculateTotalCost` 函数中创建逻辑，以计算所选 AV 项目的总成本，就像为 `venueSlice` 计算成本一样。

```
 const calculateTotalCost = (section) => {
    let totalCost = 0;
    if (section === "venue") {
      venueItems.forEach((item) => {
        totalCost += item.cost * item.quantity;
      });
    } else if (section === "av") {
      avItems.forEach((item) => {
        totalCost += item.cost * item.quantity;
      });
    }
    return totalCost;
  };
```

22. 另外，创建一个名为 `avTotalCost` 的变量，并通过传递 **av** 作为参数来调用此函数。

```
const avTotalCost = calculateTotalCost("av");
```

23. 在附加组件部分包含 `avTotalCost` 以显示所选 av 项目的总成本。

```
<div className="total_cost">Total Cost: {avTotalCost}</div>
```

任务 5：编写餐点选择的代码

在本节中，您将编写餐点选择功能的代码。为了实现此功能，您将包括复选框以选择餐点，允许客户单击复选框以选择项目或取消选中以取消选择。

餐点状态

1. 导航到 `mealsSlice.js` 文件，在 `src` 文件夹中。它将包含一个数组中的四个餐点项目。在 `initialState` 变量中包含下面的代码。

```
{ name: 'Breakfast', cost: 50, selected: false },
{ name: 'High Tea', cost: 25, selected: false },
{ name: 'Lunch', cost: 65, selected: false },
{ name: 'Dinner', cost: 70, selected: false },
```

2. 接下来，创建逻辑以选择或取消选择餐点项目，以管理 reducer 中的状态。

```
toggleMealSelection: (state, action) => {
      state[action.payload].selected = !state[action.payload].selected;
},
```

3. `toggleMealSelection` 函数切换状态中特定项目的选中属性。它接受当前状态和一个动作对象，使用 action.payload 来识别要更新的项目。然后，它将该项目的选中状态从 true 切换为 false 或反之。

▼ 查看 mealsSlice.js 的代码

```
import { createSlice } from '@reduxjs/toolkit';
export const mealsSlice = createSlice({
    name: 'meals',
    initialState: [
        { name: 'Breakfast', cost: 50, selected: false },
        { name: 'High Tea', cost: 25, selected: false },
        { name: 'Lunch', cost: 65, selected: false },
        { name: 'Dinner', cost: 70, selected: false },
    ],
    reducers: {
        toggleMealSelection: (state, action) => {
            state[action.payload].selected = !state[action.payload].selected;
        },
    },
});
export const { toggleMealSelection } = mealsSlice.actions;
export default mealsSlice.reducer;
```

将 reducers 提供给商店

4. 接下来，编辑 `store.js` 文件以提供导入 `mealsSlice.js` 的 reducer 细节。

```
import { configureStore } from '@reduxjs/toolkit';
import venueReducer from './venueSlice';
import avReducer from './avSlice';
import mealsReducer from './mealsSlice';
export default configureStore({
  reducer: {
    venue: venueReducer,
    av: avReducer,
    meals: mealsReducer,
  },
});
```

显示餐点项目

5. 接下来，您需要显示在 `mealsSlice.js` 中初始化的餐点项目的详细信息。为此，请打开 `ConferenceEvent.jsx` 文件。

6. 从 `store.js` 中导入餐点项目到 `ConferenceEvent.jsx` 组件，`mealsSlice.js` 使用 reducers 发送详细信息。

```
const mealsItems = useSelector((state) => state.meals);
```

7. 接下来，在类名为 **input-container** 的 `<div>` 标签内添加一个输入框，以获取总人数。

```
<div className="input-container venue_selection">
    <label htmlFor="numberOfPeople"><h3>Number of People:</h3></label>
    <input type="number" className="input_box5" id="numberOfPeople" value={numberOfPeople}
        onChange={(e) => setNumberOfPeople(parseInt(e.target.value))}
        min="1"
    />
</div>
```

8. 上面的代码创建了一个带标签的输入字段，用于指定人数。它使用了一个类型为数字的 `<input>` 元素，最小值为 1，并用用户输入的解析整数值更新 `numberOfPeople` 状态。

9. 要显示餐点项目，您必须在类名为 **meal_selection** 的 `<div>` 中使用 `map()` 遍历数组，就像您为附加项所做的那样。

```
<div className="meal_selection">
    {mealsItems.map((item, index) => (
        <div className="meal_item" key={index} style={{ padding: 15 }}>
            <div className="inner">
                <input type="checkbox" id={ `meal_${index}` }
                    checked={ item.selected }
                    onChange={() => handleMealSelection(index)}
                 />
                <label htmlFor={`meal_${index}`}> {item.name} </label>
            </div>
            <div className="meal_cost">${item.cost}</div>
        </div>
    ))}
</div>
```

10. 让我们逐步分析这段代码。

- `<div className="meal_selection">` 这是一个用于餐点列表的容器。
- `{mealsItems.map((item, index) => ( ... ))}` 这段代码遍历一个餐点数组，并使用提供的函数为每个项目生成HTML。
- `<div className="meal_item" key={index} style={{ padding: 15 }}>` 这是每个餐点项目的容器。`key` 属性对于React跟踪列表中的每个项目是必要的。
- `<input type="checkbox" id={meal_${index}} checked={item.selected} onChange={() => handleMealSelection(index)} />` 这是一个复选框输入元素。当前项目的选中属性控制其选中状态。当复选框状态改变时，它会触发 `handleMealSelection()` 函数，并传入当前项目的索引。
- `<label htmlFor={meal_${index}}>{item.name}</label>` 这个标签与复选框相关联。点击标签会切换复选框的状态。
- `<div className="meal_cost">${item.cost}</div>` 这显示了每个餐点项目的费用。

11. 现在你需要在 `ConferenceEvent.jsx` 文件中的 `handleMealSelection()` 函数中创建逻辑，以根据人数计算餐点小计。

```
const handleMealSelection = (index) => {
    const item = mealsItems[index];
    if (item.selected && item.type === "mealForPeople") {
        // Ensure numberOfPeople is set before toggling selection
        const newNumberOfPeople = item.selected ? numberOfPeople : 0;
        dispatch(toggleMealSelection(index, newNumberOfPeople));
    }
    else {
        dispatch(toggleMealSelection(index));
    }
};
```

12. 这是 `handleMealSelection()` 函数的解释。

- 该函数接受一个参数，表示触发选择的餐点项的索引
- 它使用提供的索引从 `mealsItems` 数组中检索餐点项对象
- 它检查检索到的项是否被选中，即 `item.selected === true`，并且其类型为 `mealForPeople`
- 如果这两个条件 **满足**，它准备更新 `numberOfPeople` 状态变量，然后切换选择
- 如果该项类型为 `mealForPeople` 并且已经被选中，`item.selected` 为 true，它将保持当前的 `numberOfPeople`
- 如果未被选中，则将 `numberOfPeople` 设置为 0
- 它使用该项的索引和（如果适用）新的 `numberOfPeople` 派发 `toggleMealSelection` 动作
- 如果该项不是 `mealForPeople` 类型或未被选中，它将派发一个动作以切换餐点选择，而不考虑其他因素

13. 在上述 `handleMealSelection()` 函数中，您正在从 `mealsSlice.jsx` 文件中派发 `toggleMealSelection` 函数。为此，请确保您已从 “./mealsSlice” 导入 `toggleMealSelection`；

```
import { toggleMealSelection } from "./mealsSlice";
```

14. 现在，在 `calculateTotalCost` 函数中创建逻辑，根据人数计算所选餐点的总费用。

```
 const calculateTotalCost = (section) => {
        let totalCost = 0;
        if (section === "venue") {
            venueItems.forEach((item) => {
                totalCost += item.cost * item.quantity;
            });
```

```
        } else if (section === "av") {
            avItems.forEach((item) => {
                totalCost += item.cost * item.quantity;
            });
        } else if (section === "meals") {
            mealsItems.forEach((item) => {
                if (item.selected) {
                  totalCost += item.cost * numberOfPeople;
                }
              });
        }
    return totalCost;
    };
```

让我们分解上述代码：

- 条件检查 (`if (section === "meals")`): 代码首先检查变量 section 是否等于字符串“meals”。如果这个条件为真，if 块中的代码将会执行。
- 遍历 mealsItems (`mealsItems.forEach((item) => { ... })`): mealsItems 被假定为一个对象数组。forEach 方法用于遍历每个元素（称为 item）。
- 检查项目是否被选中 (`if (item.selected) { ... }`): 对于 mealsItems 数组中的每个 item，代码检查该项的 selected 属性是否为真。如果该项被选中，if 块中的代码将会执行。
- 计算总成本 (`totalCost += item.cost * numberOfPeople;`): 如果该项被选中，它的成本将乘以 numberOfPeople。
- 这次乘法的结果将被加到 totalCost 中。这意味着 totalCost 正在累积所有选中项目的成本，按人数进行缩放。

15. 创建一个名为 `mealsTotalCost` 的变量，并通过传递 **meals** 作为参数来调用此函数。

```
const mealsTotalCost = calculateTotalCost("meals");
```

16. 在餐点部分包含 `mealsTotalCost` 以显示所选餐点的总费用。

```
<div className="total_cost">Total Cost: {mealsTotalCost}</div>
```

任务 6：在表格中显示选定项目

17. 接下来，您将编写代码以表格格式显示用户选择的项目和费用详细信息。当用户点击标题中的 **显示详细信息** 按钮时，这些详细信息将出现。

18. `ConferenceEvent.jsx` 组件在三元运算符 `:` 之后包含以下代码。

```
<div className="total_amount_detail">
    <TotalCost totalCosts={ totalCosts } ItemsDisplay={() => <ItemsDisplay items={ items } />} />
</div>
```

19. 让我们来看看这个代码片段。

- `TotalCost` 组件渲染在一个类名为 `total_amount_detail` 的 `<div>` 元素内。
- `TotalCost` 组件接收 `totalCosts` 和 `ItemsDisplay` 作为 props。
- `totalCosts` prop 包含成本数据，并且 `ItemsDisplay()` 组件与 `items` 作为 props 传递给 `TotalCost` 组件。

20. 接下来，让我们看看 `totalCosts` 和 `ItemsDisplay()` props。

21. 现在，您需要计算三种选择类型的小计，包括场地、附加项和餐饮。

22. 创建一个名为 `totalCosts` 的对象，其中包含所有三个小计的总成本。

```
const totalCosts = {
    venue: venueTotalCost,
    av: avTotalCost,
    meals: mealsTotalCost,
};
```

23. 在函数组件 `ConferenceEvents.jsx` 的返回之前包含上述代码。

24. 要获取选定的项目，您需要在 `getItemsFromTotalCost()` 函数中创建逻辑。您可以查看该函数的代码如下，并在代码片段后阅读代码的解释。

```
const getItemsFromTotalCost = () => {
    const items = [];
    venueItems.forEach((item) => {
      if (item.quantity > 0) {
        items.push({ ...item, type: "venue" });
      }
    });
    avItems.forEach((item) => {
      if (
        item.quantity > 0 &&
        !items.some((i) => i.name === item.name && i.type === "av")
      ) {
        items.push({ ...item, type: "av" });
      }
    });
    mealsItems.forEach((item) => {
      if (item.selected) {
        const itemForDisplay = { ...item, type: "meals" };
        if (item.numberOfPeople) {
          itemForDisplay.numberOfPeople = numberOfPeople;
        }
        items.push(itemForDisplay);
```

```
        }
      });
      return items;
    };
```

25. `getItemsFromTotalCost()` 函数将通过创建一个名为 `items` 的空数组来存储用户选择的所有项目。

- 你可以看到每种项目类型，场地、AV 和餐饮，都定义了自己的 `forEach()` 函数。这些函数只查找并包含用户选择的 `items`，并在数组中标记每个项目为“venue”、“av”或“meals”。
- 该函数返回 `items` 数组，其中包含来自三个类别的项目：场地、AV 和餐饮。

26. `getItemsFromTotalCost()` 函数在 `ConferenceEvent.jsx` 组件中被调用，代码为 `const items = getItemsFromTotalCost();`，这是你**获得的**代码。

27. 现在，你需要创建逻辑来将 `items` 详细信息显示为表格。在 `ConferenceEvent.jsx` 组件中，你会看到一个空组件 `ItemsDisplay`，其中将 `items` 变量作为 props 传递在 `{}` 内。

28. 该组件的代码**如下所示**。在查看下面的代码后，将其输入到 `ItemsDisplay` 组件中。

```
const ItemsDisplay = ({ items }) => {
    console.log(items);
    return <>
        <div className="display_box1">
            {items.length === 0 && <p>No items selected</p>}
            <table className="table_item_data">
                <thead>
                    <tr>
                        <th>Name</th>
                        <th>Unit Cost</th>
                        <th>Quantity</th>
                        <th>Subtotal</th>
                    </tr>
                </thead>
                <tbody>
                    {items.map((item, index) => (
                        <tr key={index}>
                            <td>{item.name}</td>
                            <td>${item.cost}</td>
                            <td>
                                {item.type === "meals" || item.numberOfPeople
                                ? ` For ${numberOfPeople} people`
                                : item.quantity}
                            </td>
                            <td>{item.type === "meals" || item.numberOfPeople
                                ? `${item.cost * numberOfPeople}`
                                : `${item.cost * item.quantity}`}
                            </td>
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    </>
};
```

29. Let's review how this code works.
    First, the component writes the list of items to the console, which helps with testing.

The component returns a `<div>` element with the class name `display_box1`. Within this element:

- It displays the message "No items selected" if there are no items in the `items` array.
- If the array contains items, it displays a table with the class name "table_item_data".
- The table layout has four columns: "**Name**", "**Unit Cost**", "**Quantity**", and "**Subtotal**".
- It iterates over the `items` array using the `map()` function. Each item in the `items` array creates a table row, `<tr>` element.
- Each row of the table displays the following information, respectively
  - The name of the item
  - The item's unit price with a dollar sign in front of it
  - For rooms and add-ons, it displays the quantity of the item
  - For meals, it uses the `numberOfPeople` property to display "For *x* people" where *x* is the number of people.
  - It calculates the subtotal for each item type by multiplying the item unit cost by the quantity or the number of people.

Task 7: Write the TotalCost component

1. Now, you need to create the logic to display the items in the table when the user selects the `Show Details` button.

2. Navigate to the `TotalCost.jsx` component under the **src** folder. You will see the basic layout provided. It should look like the following:

```
import React, { useState, useEffect } from 'react';
import "./TotalCost.css";
const TotalCost = ({ totalCosts, ItemsDisplay }) => {
    return (
        <div className="pricing-app">
            <div className="display_box">
                <div className="header">
                    <p className="preheading"><h3>活动的总费用</h3></p>
                </div>
                <div>
                    <h2 id="pre_fee_cost_display" className="price"></h2>
                    <div className="render_items"></div>
                </div>
            </div>
```

```
        </div>
    );
};
export default TotalCost;
```

3. Let's review this code.

- The import statements at the beginning of your file ensure access to `React`, `useState`, `useEffect`, and the stylesheet `./TotalCost.css`
- The `TotalCost()` function component takes the props `totalCosts` and `ItemsDisplay` as parameters.
- The `TotalCost` component structure includes multiple `<div>` elements along with their class names.
- In the `ConferenceEvent.jsx` component, one object named `totalCosts` aggregates the total costs for venue, audio-visual (AV), and meals.
- It assigns them to their respective properties venue, av, and meals.

4. Now, create a variable named `total_amount` to get the sum of the total costs for the venue, audio-visual (AV), and meals.
5. You can calculate this value by adding the values through the totalCosts prop parameter, `totalCosts.venue`, `totalCosts.av`, and `totalCosts.meals` as shown below:

```
const total_amount = totalCosts.venue + totalCosts.av + totalCosts.meals;
```

6. Include the above code before the return of the function component.
7. Now display the `total_amount` variable within an `<h2>` element with the id set to `pre_fee_cost_display`, using the code below:

```
<h2 id="pre_fee_cost_display" className="price">
    ${total_amount}
</h2>
```

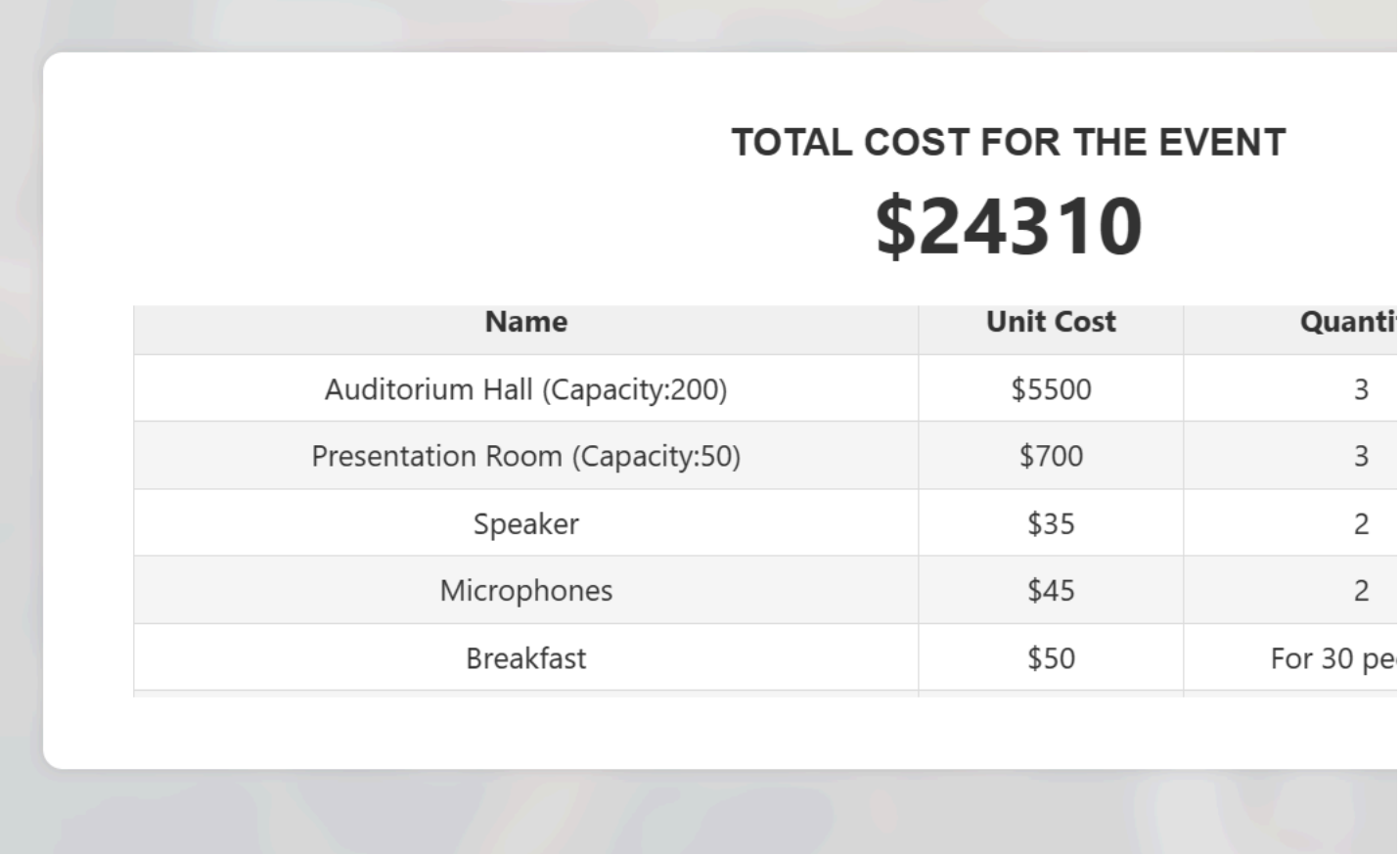
8. Next, include the `ItemsDisplay` prop within a `<div>` tag with the class name `render_items` to render items details using the `ItemsDisplay`:

```
<div className="render_items">
    <ItemsDisplay />
</div>
```

Task 8: Check The output

Now, you should have all the code you need and can check the output of your table.

1. Save all the files and re-run the application. If you already have the application open in the browser, you can refresh the application page after running it again.
2. Click the **Get Started** button.
3. Add multiple items from the venue, add-ons, and meals sections. Also, enter the number of people in the meals section.
4. Select **Show Details** in the navbar. Your output will be displayed according to what you selected on the product selection page. Here is an example of what your output might look like.



Summary and Final Solutions

In this project, you:

- Created function components
- Review the structure of the ConferenceEvent.jsx component
- Used Redux Toolkit slices to manage different parts of your application's states
- Implemented Redux actions, reducers and the Redux store to increment and decrement stored values and dynamically display those values.
- Calculated and displayed costs using arrays and the `map()` function to iterate over the arrays
- Made a dynamic table to show selected products, displaying the item names, unit cost, quantity, and subtotal for that item

Now you are prepared to develop the shopping cart application for your final project!

Solutions

Solution for `avSlice.jsx`

▼ Click here to see solution

```
    ```javascript
    import { createSlice } from "@reduxjs/toolkit";
    export const avSlice = createSlice({
      name: "av",
      initialState: [
            {
            img: "",
            name: "投影仪",
            cost: 200,
            quantity: 0,
        },
        {
            img: "",
            name: "扬声器",
            cost: 35,
            quantity: 0,
        },
        {
            img: "",
            name: "麦克风",
            cost: 45,
            quantity: 0,
        },
        {
            img: "",
            name: "白板",
            cost: 80,
            quantity: 0,
        },
        {
```
```

```
        img: "",
        name: "标识",
        cost: 80,
        quantity: 0,
    },
  ],
  reducers: {
    incrementAvQuantity: (state, action) => {
        const item = state[action.payload];
        if (item) {
            item.quantity++;
        }
    },
    decrementAvQuantity: (state, action) => {
        const item = state[action.payload];
        if (item && item.quantity > 0) {
            item.quantity--;
        }
    },
  },
});
export const { incrementAvQuantity, decrementAvQuantity } = avSlice.actions;
export default avSlice.reducer;

</details>
Solution for `ConferenceEvent.jsx`
<details>
<summary>Click here to see solution</summary>
```js
```javascript
import React, { useState } from "react";
import "./ConferenceEvent.css";
import TotalCost from "./TotalCost";
import { toggleMealSelection } from "./mealsSlice";
import { incrementAvQuantity, decrementAvQuantity } from "./avSlice";
import { useSelector, useDispatch } from "react-redux";
import { incrementQuantity, decrementQuantity } from "./venueSlice";
const ConferenceEvent = () => {
    const [showItems, setShowItems] = useState(false);
    const [numberOfPeople, setNumberOfPeople] = useState(1);
    const venueItems = useSelector((state) => state.venue);
    const avItems = useSelector((state) => state.av);
    const mealsItems = useSelector((state) => state.meals);
    const dispatch = useDispatch();
    const remainingAuditoriumQuantity = 3 - venueItems.find(item => item.name === "Auditorium Hall (Capacity:200)").quantity;

    const handleToggleItems = () => {
        console.log("handleToggleItems called");
        setShowItems(!showItems);
    };
    const handleAddToCart = (index) => {
        if (venueItems[index].name === "Auditorium Hall (Capacity:200)" && venueItems[index].quantity >= 3) {
          return;
        }
        dispatch(incrementQuantity(index));
      };

      const handleRemoveFromCart = (index) => {
        if (venueItems[index].quantity > 0) {
          dispatch(decrementQuantity(index));
        }
      };
    const handleIncrementAvQuantity = (index) => {
    dispatch(incrementAvQuantity(index));
};
const handleDecrementAvQuantity = (index) => {
    dispatch(decrementAvQuantity(index));
};
const handleMealSelection = (index) => {
    const item = mealsItems[index];
    if (item.selected && item.type === "mealForPeople") {
        // Ensure numberOfPeople is set before toggling selection
        const newNumberOfPeople = item.selected ? numberOfPeople : 0;
        dispatch(toggleMealSelection(index, newNumberOfPeople));
    }
    else {
        dispatch(toggleMealSelection(index));
    }
};
const getItemsFromTotalCost = () => {
    const items = [];
    venueItems.forEach((item) => {
      if (item.quantity > 0) {
        items.push({ ...item, type: "venue" });
      }
    });
    avItems.forEach((item) => {
      if (
        item.quantity > 0 &&
        !items.some((i) => i.name === item.name && i.type === "av")
      ) {
        items.push({ ...item, type: "av" });
      }
    });
    mealsItems.forEach((item) => {
      if (item.selected) {
        const itemForDisplay = { ...item, type: "meals" };
        if (item.numberOfPeople) {
          itemForDisplay.numberOfPeople = numberOfPeople;
        }
        items.push(itemForDisplay);
      }
    });
    return items;
```

```
  };
    const items = getItemsFromTotalCost();
    const ItemsDisplay = ({ items }) => {
        console.log(items);
        return <>
            <div className="display_box1">
                {items.length === 0 && <p>没有选择任何项目</p>}
                <table className="table_item_data">
                    <thead>
                        <tr>
                            <th>名称</th>
                            <th>单价</th>
                            <th>数量</th>
                            <th>小计</th>
                        </tr>
                    </thead>
                    <tbody>
                        {items.map((item, index) => (
                            <tr key={index}>
                                <td>{item.name}</td>
                                <td>${item.cost}</td>
                                <td>
                                    {item.type === "meals" || item.numberOfPeople
                                    ? ` ${numberOfPeople} 人`
                                    : item.quantity}
                                </td>
                                <td>{item.type === "meals" || item.numberOfPeople
                                    ? `${item.cost * numberOfPeople}`
                                    : `${item.cost * item.quantity}`}
                                </td>
                            </tr>
                        ))}
                    </tbody>
                </table>
            </div>
        </>
    };
    const calculateTotalCost = (section) => {
        let totalCost = 0;
        if (section === "venue") {
            venueItems.forEach((item) => {
                totalCost += item.cost * item.quantity;
            });
        } else if (section === "av") {
            avItems.forEach((item) => {
                totalCost += item.cost * item.quantity;
            });
        } else if (section === "meals") {
            mealsItems.forEach((item) => {
                if (item.selected) {
                  totalCost += item.cost * numberOfPeople;
                }
              });
        }
    return totalCost;
    };
    const venueTotalCost = calculateTotalCost("venue");
const avTotalCost = calculateTotalCost("av");
const mealsTotalCost = calculateTotalCost("meals");
    const navigateToProducts = (idType) => {
        if (idType == '#venue' || idType == '#addons' || idType == '#meals') {
          if (showItems) { // Check if showItems is false
            setShowItems(!showItems); // Toggle showItems to true only if it's currently false
          }
        }
      }
      const totalCosts = {
        venue: venueTotalCost,
        av: avTotalCost,
        meals: mealsTotalCost,
    };
    return (
        <>
            <navbar className="navbar_event_conference">
                <div className="company_logo">会议费用规划器</div>
                <div className="left_navbar">
                    <div className="nav_links">
                        <a href="#venue" onClick={() => navigateToProducts("#venue")} >场地</a>
                        <a href="#addons" onClick={() => navigateToProducts('#addons')}>附加项目</a>
                        <a href="#meals" onClick={() => navigateToProducts('#meals')}>餐饮</a>
                    </div>
                    <button className="details_button" onClick={() => setShowItems(!showItems)}>
                        显示详情
                    </button>
                </div>
            </navbar>
            <div className="main_container">
                {!showItems
                    ?
                    (
                        <div className="items-information">
                            <div id="venue" className="venue_container container_main">
        <div className="text">

          <h1>场地选择</h1>
        </div>
        <div className="venue_selection">
          {venueItems.map((item, index) => (
            <div className="venue_main" key={index}>
              <div className="img">
                <img src={item.img} alt={item.name} />
              </div>
```

```
              <div className="text">{item.name}</div>
              <div>${item.cost}</div>
     <div className="button_container">
        {venueItems[index].name === "Auditorium Hall (Capacity:200)" ? (
          <>
          <button
            className={venueItems[index].quantity === 0 ? "btn-warning btn-disabled" : "btn-minus btn-warning"}
            onClick={() => handleRemoveFromCart(index)}
          >
            &#8211;
          </button>
          <span className="selected_count">
            {venueItems[index].quantity > 0 ? ` ${venueItems[index].quantity}` : "0"}
          </span>
          <button
            className={remainingAuditoriumQuantity === 0? "btn-success btn-disabled" : "btn-success btn-plus"}
            onClick={() => handleAddToCart(index)}
          >
            &#43;
          </button>
        </>
        ) : (
          <div className="button_container">
           <button
              className={venueItems[index].quantity ===0 ? " btn-warning btn-disabled" : "btn-warning btn-plus"}
              onClick={() => handleRemoveFromCart(index)}
            >
               &#8211;
            </button>
            <span className="selected_count">
              {venueItems[index].quantity > 0 ? ` ${venueItems[index].quantity}` : "0"}
            </span>
            <button
              className={venueItems[index].quantity === 10 ? " btn-success btn-disabled" : "btn-success btn-plus"}
              onClick={() => handleAddToCart(index)}
            >
             &#43;
            </button>


          </div>
        )}
      </div>
            </div>
          ))}
        </div>
        <div className="total_cost">总费用: ${venueTotalCost}</div>
      </div>
                              {/*必要的附加项目*/}
                              <div id="addons" className="venue_container container_main">
                                  <div className="text">
                                      <h1> 附加项目选择</h1>
                                  </div>
                                  <div className="addons_selection">
                                  {avItems.map((item, index) => (
    <div className="av_data venue_main" key={index}>
        <div className="img">
            <img src={item.img} alt={item.name} />
        </div>
    <div className="text"> {item.name} </div>
    <div> ${item.cost} </div>
        <div className="addons_btn">
            <button className="btn-warning" onClick={() => handleDecrementAvQuantity(index)}> &ndash; </button>
            <span className="quantity-value">{item.quantity}</span>
            <button className=" btn-success" onClick={() => handleIncrementAvQuantity(index)}> &#43; </button>
        </div>
    </div>
))}
                                  </div>
<div className="total_cost">总费用: {avTotalCost}</div>
                              </div>
                              {/* 餐饮部分 */}
                              <div id="meals" className="venue_container container_main">
                                  <div className="text">
                                      <h1>餐饮选择</h1>
                                  </div>
                                  <div className="input-container venue_selection">
                                  <div className="input-container venue_selection">
    <label htmlFor="numberOfPeople"><h3>人数:</h3></label>
    <input type="number" className="input_box5" id="numberOfPeople" value={numberOfPeople}
        onChange={(e) => setNumberOfPeople(parseInt(e.target.value))}
        min="1"
    />
</div>
                                  </div>
                                  <div className="meal_selection">
    {mealsItems.map((item, index) => (
        <div className="meal_item" key={index} style={{ padding: 15 }}>
            <div className="inner">
                <input type="checkbox" id={ `meal_${index}` }
                    checked={ item.selected }
                    onChange={() => handleMealSelection(index)}
                 />
                <label htmlFor={`meal_${index}`}> {item.name} </label>
            </div>
            <div className="meal_cost">${item.cost}</div>
        </div>
    ))}
</div>
<div className="total_cost">总费用: {mealsTotalCost}</div>
                              </div>
                          </div>
                      ) : (
```

```
                        <div className="total_amount_detail">
    <TotalCost totalCosts={ totalCosts } ItemsDisplay={() => <ItemsDisplay items={ items } />} />
</div>
                    )
                }
            </div>
        </>
    );
};
export default ConferenceEvent;
```

</details>
Solution for `mealsSlice.jsx`
<details>
<summary>Click here to see solution</summary>
```js
```javascript
import { createSlice } from '@reduxjs/toolkit';
export const mealsSlice = createSlice({
  name: 'meals',
  initialState: [
    { name: '早餐', cost: 50, selected: false },
{ name: '下午茶', cost: 25, selected: false },
{ name: '午餐', cost: 65, selected: false },
{ name: '晚餐', cost: 70, selected: false },

  ],
  reducers: {
    toggleMealSelection: (state, action) => {
        state[action.payload].selected = !state[action.payload].selected;
  },
  },
});
export const { toggleMealSelection } = mealsSlice.actions;
export default mealsSlice.reducer;
```

</details>
Solution for `store.js`
<details>
<summary>Click here to see solution</summary>
```js
```javascript
import { configureStore } from '@reduxjs/toolkit';
import venueReducer from './VenueSlice';
import avReducer from './avSlice';
import mealsReducer from './mealsSlice';
export default configureStore({
  reducer: {
    venue: venueReducer,
    av: avReducer,
    meals: mealsReducer,
  },
});
```

</details>
Solution for `TotalCost.jsx`
<details>
<summary>Click here to see solution</summary>
```js
```javascript
import React, { useState, useEffect } from 'react';
import "./TotalCost.css";
const TotalCost = ({ totalCosts, ItemsDisplay }) => {
    const total_amount = totalCosts.venue + totalCosts.av + totalCosts.meals;
    return (
        <div className="pricing-app">
            <div className="display_box">
                <div className="header">
                    <p className="preheading"><h3>活动的总费用</h3></p>
                </div>
                <div>
                <h2 id="pre_fee_cost_display" className="price">
    ${total_amount}
</h2>
<div className="render_items">
    <ItemsDisplay />
</div>
                </div>
            </div>
        </div>
    );
};
export default TotalCost;
```

</details>
Solution for `VenueSlice.jsx`
<details>
<summary>Click here to see solution</summary>
```js
```javascript
// venueSlice.js
import { createSlice } from "@reduxjs/toolkit";
export const venueSlice = createSlice({
  name: "venue",
  initialState: [
    {
      img: "",
      name: "会议室 (容量:15)",
      cost: 3500,
      quantity: 0,
    },
    {
      img: "",
```

```
          name: "礼堂 (容量:200)",
          cost: 5500,
          quantity: 0,
        },
        {
          img: "",
          name: "演示室 (容量:50)",
          cost: 700,
          quantity: 0,
        },
        {
          img: "",
          name: "大会议室 (容量:10)",
          cost: 900,
          quantity: 0,
        },
        {
          img: "",
          name: "小会议室 (容量:5)",
          cost: 1100,
          quantity: 0,
        },

      ],
      reducers: {

        incrementQuantity: (state, action) => {
          const { payload: index } = action;
          if (state[index]) {
            if (state[index].name === " 礼堂 (容量:200)" && state[index].quantity >= 3) {
              return;        }
            state[index].quantity++;
          }
        },
        decrementQuantity: (state, action) => {
          const { payload: index } = action;
          if (state[index] && state[index].quantity > 0) {
            state[index].quantity--;
          }
        },
      },
    });
    export const { incrementQuantity, decrementQuantity } = venueSlice.actions;
    export default venueSlice.reducer;

```

Author(s)

Richa Arora
Bethany Hudnutt

© IBM Corporation. All rights reserved.