# Unit Test - Dashboard

**Test Cases**

**Test Case 1: Render Dashboard Components Correctly**

- **Objective:** Validate that the Dashboard component correctly renders all its initial UI elements.
- **Method:** `render`
- **Test Steps:**
  a. Render the Dashboard component within a BrowserRouter.
  b. Check for the presence of essential elements such as booking type filters and location filters.
- **Expected Result:** The "Booking", "All Types", and "All Locations" text elements are visible upon initial render.

**Test Case 2: Toggle Active Class on Filter Buttons**

- **Objective:** Ensure that clicking on filter buttons updates their visual state to reflect the current selection.
- **Method:** `userEvent.click`
- **Test Steps:**
  a. Render the Dashboard component.
  b. Simulate user clicks on the "Completed" filter button.
  c. Verify that the "Completed" button receives an "active" class.
  d. Verify that previously active buttons lose the "active" class.
- **Expected Result:** Only the clicked (Completed) button should have the "active" class, ensuring correct visual feedback for user interactions.

**Test Case 3: Filter Bookings by Type**

- **Objective:** Verify that the booking type filter correctly alters the displayed bookings.
- **Method:** `fireEvent.change`
- **Test Steps:**
  a. Render the Dashboard component.
  b. Select a specific booking type from the dropdown.
  c. Verify that the internal state of the component reflects this selection.
- **Expected Result:** The component should only display bookings of the selected type, and the dropdown should show the selected value.

**Test Case 4: Filter Bookings by Date Range**

- **Objective:** Ensure the date range filters adjust which bookings are displayed based on the selected range.
- **Method:** `fireEvent.change`
- **Test Steps:**
  a. Render the Dashboard component.
  b. Set a start and end date via the date picker inputs.
  c. Verify the values in the date pickers match the entered dates.
  d. Optionally, check if displayed bookings fall within the selected date range.
- **Expected Result:** The date pickers should reflect the set dates, and the displayed bookings should be limited to those within the specified range.

**Test Case 5: Sort Bookings by Date**

- **Objective:** Test the functionality of the sorting feature for displaying bookings in ascending or descending order by date.
- **Method:** `userEvent.click`

- **Test Steps:**
  a. Render the Dashboard component.
  b. Click the sort button to toggle between ascending and descending order.
  c. Verify that the button text updates to reflect the sort order.
  d. Optionally, verify that bookings are sorted correctly in the UI.
- **Expected Result:** The sort button should toggle between "Sort Date Ascending" and "Sort Date Descending," and bookings should reorder accordingly.

**Test Case 6: Filter Bookings by Location**

- **Objective:** Confirm that the location filter effectively narrows down the displayed bookings based on selected locations.
- **Method:** `userEvent.selectOptions`
- **Test Steps:**
  a. Render the Dashboard component.
  b. Select a location from the dropdown menu.
  c. Verify that only bookings from the selected location are displayed.
- **Expected Result:** The dashboard should only show bookings located in the chosen venue, with all displayed bookings meeting this criterion.