Unit Test - NewBooking

Test Cases

Test Case 1: Render and Category Switching

- **Objective:** Validate that the NewBooking component renders correctly and can switch between different category forms without errors.
- **Method:** `render`
- **Test Steps:**
  a. Render the NewBooking component.
  b. Verify that the initial category "Delivery" is displayed correctly by checking for the presence of "Program Stream:".
- **Expected Result:** The component should display elements specific to the "Delivery" category initially, indicating correct rendering and category state initialization.

Test Case 2: Handle School Form Input Correctly

- **Objective:** Ensure that the form for the "School" category accepts and correctly handles user input, including text inputs and toggling checkboxes.
- **Method:** `userEvent.click`, `userEvent.type`
- **Test Steps:**
  a. Click on the "School" category button to switch forms.
  b. Type into the "School Name" input field and verify the input retains the typed value.
  c. Toggle a year checkbox and verify its checked state.
- **Expected Result:** The "School Name" input should reflect the typed text, and the year checkbox should maintain its toggled state, confirming correct state management and user input handling.

Test Case 3: Submit the Bus Form Correctly

- **Objective:** Test the submission of the "Bus" form, ensuring inputs accept values and the form can submit without issues.
- **Method:** `userEvent.type`, `userEvent.click`
- **Test Steps:**
  a. Activate the "Bus" category and fill out required fields including "BUS REQ" and status.
  b. Select an option from the status dropdown and submit the form.
  c. Optionally, verify if any callbacks or effects occur upon submission.
- **Expected Result:** All form fields should accept and display the entered values, and clicking the submit button should trigger any defined actions or state changes related to form submission.

Test Case 4: Manage State Changes in the 'Others' Form

- **Objective:** Verify that the "Others" form correctly manages state changes when user input is provided, particularly for numerical inputs.
- **Method:** `userEvent.type`
- **Test Steps:**
  a. Switch to the "Others" category.
  b. Input a value into the "Profit" field and verify the field reflects this input.
- **Expected Result:** The "Profit" field should correctly update and retain the value input by the user, demonstrating effective state handling and responsiveness of the form.