Data Flow Diagram (DFD) for To-Do List React application

The To-Do List Application is a simple task management system built using React.js. It allows users to add, store, and display tasks dynamically. The application features a user-friendly interface where users can enter tasks, which are then stored in the application's state and displayed in a list format.

Level 0 (Context Diagram)

At this level, we represent the system as a whole and its interaction with external entities.

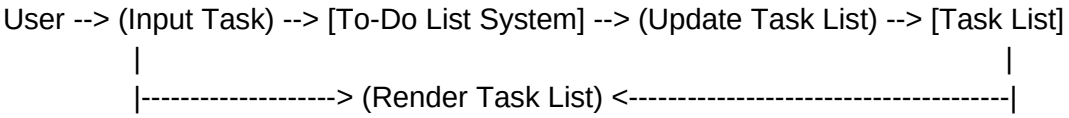
User --> [To-Do List System] --> Task List

Explanation:

- **External Entity:** User
- **Process:** To-Do List System
- **Data Store:** Task List

Level 1 (Detailed Breakdown)

At this level, we break down the main process into smaller components.



Explanation:

1. **User Inputs Task**
    - User enters a new task in the input field.
2. **Process Task Addition**
    - The system updates the task list with the new entry.
3. **Store Tasks**
    - Tasks are stored in the task list state (useState).
4. **Display Tasks**
    - The system renders the list of tasks in the UI.

Data Flow

The data flow in the To-Do List Application follows these steps:

1. **User Inputs Task:** The user enters a new task in the input field.
2. **Process Task Addition:** The system processes and updates the task list with the new entry.
3. **Store Tasks:** The task is stored in the application state (useState).
4. **Display Tasks:** The updated task list is rendered dynamically for the user.

Additional Notes

- **State Management:** The application uses React's useState hook to manage task data efficiently.
- **Dynamic Rendering:** The task list updates instantly upon adding a new task.
- **Scalability:** Future enhancements may include task deletion, task editing, and persistent storage using a database or local storage.
- **User Experience:** The UI is designed to be minimalistic and user-friendly for ease of interaction.