**Week 10 Lab: React Chat Application with**[**Socket.io**](http://socket.io/)

**Objective**

In this lab, you will create a **React-based Chat Application** that connects to a [Socket.io](http://socket.io/) server. The server is hosted at [**https://stingray-app-7s57q.ondigitalocean.app**](https://stingray-app-7s57q.ondigitalocean.app/)**.**.

Your task is to set up:

* A broadcast chat where all users can see messages.
* A private messaging system where users can directly message each other.

**Initial Setup**

1. Download the provided base files (App.jsx without [Socket.io](http://socket.io/) functionality).
2. Open a terminal, navigate to the project folder, and run:
3. npm install
4. npm run dev

This will install dependencies and start the development server.

[**Socket.io**](http://socket.io/)**Integration**

* You need to establish a connection to the provided [Socket.io](http://socket.io/) server and set up events for messaging.
* Import io from socket.io-client at the top of your App.jsx file, and connect to [**https://stingray-app-7s57q.ondigitalocean.app**](https://stingray-app-7s57q.ondigitalocean.app/)

**Socket Event Details**

To complete this lab, you’ll need to implement the following socket events:

1. **Register User**:
   * **Event Name**: "register\_user"
   * **Emit**: When a user connects, send their nickname.
   * **Data to Send**:
     + **Type**: string
     + **Example**: "JohnDoe"
   * **Description**: This event registers the user on the server so that they can participate in the chat.
2. **Broadcast Message**:
   * **Event Name**: "message"
   * **Emit**: When a user wants to send a broadcast message.
   * **Data to Send**:
     + **Type**: { message: string }
     + **Example**: { message: "Hello World"}
   * **Receive**: When a broadcast message is received.
   * **Data Received**:
     + **Type**: { user: string, message: string }
     + **Example**: { user: "JohnDoe", message: "Hello everyone!" }
   * **Description**: Send and receive messages visible to all connected users.
3. **Private Message**:
   * **Event Name**: "direct\_message"
   * **Emit**: When a user wants to send a private message to another user.
   * **Data to Send**:
     + **Type**: { recipient: string, message: string }
     + **Example**: { recipient: "JaneDoe", message: "Hey, how are you?" }
   * **Receive**: When a private message is received.
   * **Data Received**:
     + **Type**: { from: string, message: string }
     + **Example**: { from: "JaneDoe", message: "I'm good, thanks!" }
   * **Description**: Send and receive messages between specific users.

**Tasks**

* **Connect to the Server**: Set up the socket connection and emit the "register\_user" event when a user connects.
* **Broadcast Messaging**: Implement the "message" event to send and receive messages to/from all users.
* **Private Messaging**: Implement the "direct\_message" event to send messages to a specific user and receive messages meant only for you.

**Testing**

* Use multiple tabs to simulate different users and test both broadcast and private messaging.
* Assign different nicknames to each tab and verify all features work as intended.