**Procedure for MongoDB, MERN Stack, and React.js Development**

**1. Setting up MongoDB**

1. **Install MongoDB** on your local machine or use a cloud service like MongoDB Atlas.
2. **Create a database** to store your application data.
3. **Define collections** within the database to organize your data (e.g., users, products).
4. **Insert initial data** into collections if needed.
5. **Configure access and security** such as user authentication and network rules.
6. **Use MongoDB tools or drivers** to connect your backend application to the database.

**2. Setting up the MERN Stack Backend (MongoDB, Express, Node.js)**

1. **Initialize a Node.js project** to manage dependencies.
2. **Install Express.js** to create the backend server and handle HTTP requests.
3. **Connect Express to MongoDB** using a MongoDB driver or an Object Data Modeling (ODM) library.
4. **Define API routes** in Express to handle CRUD operations (Create, Read, Update, Delete).
5. **Implement middleware** for parsing request bodies, handling errors, and managing security.
6. **Test API endpoints** using tools like Postman or curl.
7. **Run the backend server** to listen for incoming requests.

**3. Setting up the React.js Frontend**

1. **Create a React application** using a tool like Create React App.
2. **Design UI components** to represent different parts of the user interface.
3. **Manage component state** to handle dynamic data and user interactions.
4. **Use lifecycle methods or hooks** to fetch data from the backend API.
5. **Make HTTP requests** from React to the Express backend to retrieve or send data.
6. **Display data** received from the backend in the UI components.
7. **Handle user input and events** to update state and communicate with the backend.
8. **Implement routing** if your app has multiple pages or views.

**4. Integrating Frontend and Backend**

1. **Ensure backend API is accessible** from the frontend (handle CORS if needed).
2. **Use HTTP methods** (GET, POST, PUT, DELETE) in React to interact with backend routes.
3. **Update frontend state** based on responses from the backend.
4. **Handle loading states and errors** gracefully in the UI.
5. **Deploy backend and frontend** either separately or together, ensuring proper environment configuration.

**5. Testing and Deployment**

1. **Test the full application** end-to-end to verify data flow and UI behavior.
2. **Optimize performance** on both frontend and backend.
3. **Secure the application** by implementing authentication, authorization, and data validation.
4. **Deploy the backend** to a server or cloud platform.
5. **Deploy the React frontend** to a static hosting service or alongside the backend.
6. **Monitor and maintain** the application post-deployment.