# React – Json-Server And Firebase Real Time Database

```
npx create-react-app react-firebase-jsonserver

cd react-firebase-jsonserver
```

Step 2: Set Up JSON-Server

```
npm install -g json-server
```

Next, create a `db.json` file in the root of your project to act as the database:

```
{
  "todos": [
    { "id": 1, "title": "Learn React", "completed": false },
    { "id": 2, "title": "Learn Firebase", "completed": false },
    { "id": 3, "title": "Integrate JSON-Server", "completed": false }
  ]
}
```

To start JSON-Server, add the following script to your `package.json`:

```
"scripts": {
  "start": "react-scripts start",
  "json-server": "json-server --watch db.json --port 5000"
}
```

Run JSON-Server:

The server will be running on http://localhost:5000.

## Step 3: Set Up Firebase Real Time Database

1. **Create a Firebase Project**:
   - Go to the Firebase Console.
   - Click on "Add project" and follow the setup steps.
2. **Add Firebase to Your Web App**:
   - In the project overview, click on the web icon (</>) to set up Firebase for web.
   - Copy the Firebase configuration object.
3. **Install Firebase in Your React Project**:

```
npm install firebase
```

**Configure Firebase**: Create a `firebase.js` file in the `src` folder and add your Firebase configuration:

```
import firebase from 'firebase/app';
import 'firebase/database';

const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  databaseURL: "YOUR_DATABASE_URL",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_STORAGE_BUCKET",
  messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
  appId: "YOUR_APP_ID"
};

firebase.initializeApp(firebaseConfig);

const database = firebase.database();

export default database;
```

## Step 4: Create Components

*App Component*

Update `App.js` to fetch data from both JSON-Server and Firebase:

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
```

```
import database from './firebase';
import TodoList from './components/TodoList';
import TodoForm from './components/TodoForm';

const App = () => {
  const [todos, setTodos] = useState([]);

  useEffect(() => {
    // Fetch data from JSON-Server
    axios.get('http://localhost:5000/todos')
      .then(response => setTodos(response.data))
      .catch(error => console.error('Error fetching data from JSON-Server: ',
error));

    // Fetch data from Firebase Real Time Database
    database.ref('todos').on('value', (snapshot) => {
      const firebaseTodos = snapshot.val() ? Object.values(snapshot.val()) : [];
      setTodos(firebaseTodos);
    });
  }, []);

  const addTodo = (newTodo) => {
    // Add to JSON-Server
    axios.post('http://localhost:5000/todos', newTodo)
      .then(response => setTodos([...todos, response.data]))
      .catch(error => console.error('Error adding todo to JSON-Server: ', error));
```

```
    // Add to Firebase Real Time Database
    const newTodoRef = database.ref('todos').push();
    newTodoRef.set(newTodo);
  };

  const removeTodo = (id) => {
    // Remove from JSON-Server
    axios.delete(`http://localhost:5000/todos/${id}`)
      .then(() => setTodos(todos.filter(todo => todo.id !== id)))
      .catch(error => console.error('Error removing todo from JSON-Server: ', error));

    // Remove from Firebase Real Time Database
    database.ref('todos').orderByChild('id').equalTo(id).once('value', (snapshot) => {
      const key = Object.keys(snapshot.val())[0];
      database.ref(`todos/${key}`).remove();
    });
  };

  return (
    <div className="App">
      <h1>Todo List</h1>
      <TodoForm addTodo={addTodo} />
      <TodoList todos={todos} removeTodo={removeTodo} />
    </div>
  );
```

```
};

export default App;
```

*TodoList Component*

Create `TodoList.js` in the `components` folder:

`jsx`

```
import React from 'react';
import './TodoList.css';

const TodoList = ({ todos, removeTodo }) => {
  return (
    <ul>
      {todos.map(todo => (
        <li key={todo.id}>
          {todo.title}
          <button onClick={() => removeTodo(todo.id)}>Remove</button>
        </li>
      ))}
    </ul>
  );
};

export default TodoList;
```

*TodoForm Component*

Create `TodoForm.js` in the `components` folder:

```
import React, { useState } from 'react';
```

```
const TodoForm = ({ addTodo }) => {
  const [title, setTitle] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    if (title.trim()) {
      const newTodo = {
        id: Date.now(),
        title: title,
        completed: false
      };
      addTodo(newTodo);
      setTitle('');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        value={title}
        onChange={(e) => setTitle(e.target.value)}
        placeholder="Add new todo"
      />
      <button type="submit">Add</button>
```

```
    </form>
  );
};

export default TodoForm;
```

### Step 5: Add CSS Styling

Create a `TodoList.css` file in the `components` folder:

```
ul {
  list-style-type: none;
  padding: 0;
}

li {
  display: flex;
  justify-content: space-between;
  margin: 5px 0;
  padding: 10px;
  background: #f4f4f4;
  border-radius: 5px;
}

button {
  background: #ff4d4d;
  color: white;
  border: none;
  padding: 5px 10px;
```

```
  cursor: pointer;
  border-radius: 5px;
}

button:hover {
  background: #ff0000;
}
```

npm start npm run json-server