

MERN Stack with Login and Cricket Team CRUD Operations

This MERN stack application includes a login page that navigates to a Cricket Team CRUD page upon form submission. The backend provides open endpoints for creating, reading, updating, and deleting cricket teams.

Setup Instructions

1. Create a react app

```
npx create-react-app your_appname
```

2. Frontend codes

```
npm install axios react-router-dom  
npm start
```

3. Create a backend folder

4. Create Models, Routes

5. Terminal => cd backend

6. Backend codes

```
npm install express mongoose cors bcryptjs jsonwebtoken
```

```
npm init -y
```

```
node server.js
```

Frontend Code

public/index.html

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>MERN Cricket Team CRUD App</title>
<link rel="stylesheet" href="/src/styles.css">
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

src/App.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Login from './components/Login';
import CricketTeamCRUD from './components/CricketTeamCRUD';

function App() {
  return (
    <Router>
      <div className="min-h-screen bg-gray-100">
        <Routes>
          <Route path="/" element={<Login />} />
          <Route path="/dashboard" element={<CricketTeamCRUD />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

src/components/Login.js

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

import { useNavigate } from 'react-router-dom';

import './styles.css';

const Login = () => {

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [error, setError] = useState("");
```

```
const navigate = useNavigate();

const handleSubmit = (e) => {
    e.preventDefault();

    // Validation: check if fields are empty
    if (!email.trim() || !password.trim()) {
        setError('Both fields are required');
        return;
    }

    setError('');

    // No authentication, directly navigate to dashboard
    navigate('/dashboard');

};

return (
<div className="login-container">

    <div className="login-card">

        <h2 className="login-title">Welcome</h2>

        <p className="login-subtitle">Enter details to manage cricket teams</p>

        <form onSubmit={handleSubmit} className="login-form">

            {error && <p className="error-message">{error}</p>}

            <div className="form-group">

                <label htmlFor="email">Email</label>

                <input
                    type="email"
                    id="email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}>

            </div>
        </form>
    </div>
</div>
);
```

```

    className="input-field"
  />

</div>

<div className="form-group">

  <label htmlFor="password">Password</label>

  <input
    type="password"
    id="password"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
    className="input-field"
  />

</div>

<button type="submit" className="btn btn-login">
  Enter
</button>

</form>

</div>

</div>

);

};

export default Login;

```

src/components/CricketTeamCRUD.js

```

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

const CricketTeamCRUD = () => {
  const [teams, setTeams] = useState([]);
  const [formData, setFormData] = useState({ name: "", coach: "", captain: "" });
  const [editingId, setEditingId] = useState(null);
  const [message, setMessage] = useState("");

```

```

useEffect(() => {
  fetchTeams();
}, []);

const fetchTeams = async () => {
  try {
    const response = await axios.get('http://localhost:5000/api/cricketTeams');
    setTeams(response.data);
  } catch (err) {
    setMessage('Failed to fetch teams');
  }
};

const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    if (editingId) {
      await axios.put(`http://localhost:5000/api/cricketTeams/${editingId}`, formData);
      setMessage('Team updated successfully');
    } else {
      await axios.post('http://localhost:5000/api/cricketTeams', formData);
      setMessage('Team added successfully');
    }
    setFormData({ name: '', coach: '', captain: '' });
    setEditingId(null);
    fetchTeams();
  } catch (err) {
    setMessage(err.response?.data?.message || 'Operation failed');
  }
};

const handleEdit = (team) => {
  setFormData({ name: team.name, coach: team.coach, captain: team.captain });
  setEditingId(team._id);
};

const handleDelete = async (id) => {
  try {
    await axios.delete(`http://localhost:5000/api/cricketTeams/${id}`);
    setMessage('Team deleted successfully');
    fetchTeams();
  } catch (err) {
    setMessage('Failed to delete team');
  }
};

return (
  <div className="container">
    <h2 className="page-title">Cricket Team Management</h2>
    {message && <p className="message">{message}</p>}
    <div className="form-container">
      <h3>{editingId ? 'Edit Team' : 'Add Team'}</h3>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="name">Team Name</label>
          <input
            type="text"
            id="name"
            value={formData.name}
            onChange={(e) => setFormData({ ...formData, name: e.target.value })}>
        </div>
    
```

```

        className="input-field"
      />
    </div>
    <div className="form-group">
      <label htmlFor="coach">Coach</label>
      <input
        type="text"
        id="coach"
        value={formData.coach}
        onChange={(e) => setFormData({ ...formData, coach: e.target.value })}
        className="input-field"
      />
    </div>
    <div className="form-group">
      <label htmlFor="captain">Captain</label>
      <input
        type="text"
        id="captain"
        value={formData.captain}
        onChange={(e) => setFormData({ ...formData, captain: e.target.value })}
        className="input-field"
      />
    </div>
    <button type="submit" className="btn btn-primary">
      {editingId ? 'Update' : 'Add'} Team
    </button>
  </form>
</div>
<div className="table-container">
  <h3>Team List</h3>
  <table>
    <thead>
      <tr>
        <th>Team Name</th>
        <th>Coach</th>
        <th>Captain</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {teams.map((team) => (
        <tr key={team._id}>
          <td>{team.name}</td>
          <td>{team.coach}</td>
          <td>{team.captain}</td>
          <td>
            <button className="btn btn-edit" onClick={() => handleEdit(team)}>
              Edit
            </button>
            <button className="btn btn-delete" onClick={() => handleDelete(team._id)}>
              Delete
            </button>
          </td>
        </tr>
      )));
    </tbody>
  </table>
</div>
</div>
);

```

```
};

export default CricketTeamCRUD;
```

src/styles.css

```
/* General Styles */
body {
  font-family: 'Poppins', sans-serif;
  margin: 0;
  padding: 0;
  background: linear-gradient(135deg, #e0e7ff, #e6f3fa);
  min-height: 100vh;
}

/* Login Page Styles */
.login-container {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  background: linear-gradient(135deg, #6b7280, #3b82f6);
  padding: 20px;
}

.login-card {
  background: #fff;
  border-radius: 16px;
  padding: 40px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
  width: 100%;
  max-width: 400px;
  text-align: center;
  animation: fadeIn 0.5s ease-in-out;
}

.login-title {
  font-size: 2rem;
  color: #1e3a8a;
  margin-bottom: 10px;
  font-weight: 600;
}

.login-subtitle {
  font-size: 1rem;
  color: #64748b;
  margin-bottom: 20px;
}

.login-form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

.form-group {
  text-align: left;
}
```

```
.form-group label {
  font-size: 1rem;
  color: #1e3a8a;
  margin-bottom: 8px;
  display: block;
  font-weight: 500;
}

.input-field {
  width: 100%;
  padding: 12px;
  border: 2px solid #e2e8f0;
  border-radius: 8px;
  font-size: 1rem;
  transition: all 0.3s ease;
}

.input-field:focus {
  border-color: #3b82f6;
  box-shadow: 0 0 8px rgba(59, 130, 246, 0.3);
  outline: none;
}

.btn-login {
  background: linear-gradient(90deg, #3b82f6, #1e3a8a);
  color: #fff;
  padding: 12px;
  border: none;
  border-radius: 8px;
  font-size: 1.1rem;
  font-weight: 500;
  cursor: pointer;
  transition: transform 0.2s, box-shadow 0.2s;
}

.btn-login:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);
}

@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

/* Cricket Team CRUD Page Styles */
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}

.page-title {
  font-size: 2.5rem;
```

```
color: #1e3a8a;
text-align: center;
margin-bottom: 30px;
text-transform: uppercase;
font-weight: 600;
}

.form-container {
background: #fff;
padding: 20px;
border-radius: 8px;
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
margin-bottom: 30px;
}

.form-container h3 {
font-size: 1.5rem;
color: #1e3a8a;
margin-bottom: 20px;
}

.form-group {
margin-bottom: 15px;
}

.form-group label {
display: block;
font-size: 1rem;
color: #1e3a8a;
margin-bottom: 5px;
font-weight: 500;
}

.form-group input {
width: 100%;
padding: 10px;
border: 2px solid #e2e8f0;
border-radius: 5px;
font-size: 1rem;
transition: border-color 0.3s, box-shadow 0.3s;
}

.form-group input:focus {
border-color: #3b82f6;
box-shadow: 0 0 8px rgba(59, 130, 246, 0.3);
outline: none;
}

.btn {
padding: 10px 20px;
border: none;
border-radius: 5px;
font-size: 1rem;
cursor: pointer;
transition: background-color 0.3s, transform 0.2s;
}

.btn-primary {
background: linear-gradient(90deg, #3b82f6, #1e3a8a);
color: #fff;
```

```
}

.btn-primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(59, 130, 246, 0.4);
}

.btn-edit {
  background-color: #f59e0b;
  color: #fff;
  margin-right: 10px;
}

.btn-edit:hover {
  background-color: #d97706;
  transform: translateY(-2px);
}

.btn-delete {
  background-color: #ef4444;
  color: #fff;
}

.btn-delete:hover {
  background-color: #b91c1c;
  transform: translateY(-2px);
}

.table-container {
  background: #fff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

.table-container h3 {
  font-size: 1.5rem;
  color: #1e3a8a;
  margin-bottom: 20px;
}

table {
  width: 100%;
  border-collapse: collapse;
}

th, td {
  padding: 12px;
  text-align: left;
  border-bottom: 1px solid #e2e8f0;
}

th {
  background: linear-gradient(90deg, #3b82f6, #1e3a8a);
  color: #fff;
  font-weight: 600;
}

tr:hover {
  background-color: #f1f5f9;
```

```
}
```

Backend Code

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const cricketTeamRoutes = require('./routes/cricketTeams');

const app = express();

// Middleware
app.use(cors({ origin: 'http://localhost:3000' }));
app.use(express.json());

// Debug middleware
app.use((req, res, next) => {
  console.log(`Request: ${req.method} ${req.url}`);
  next();
});

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/mern-cricket', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('MongoDB connected successfully'))
.catch((err) => console.error('MongoDB connection error:', err.message));

// Routes
app.use('/api/cricketTeams', cricketTeamRoutes);

// Test route
app.get('/api/test', (req, res) => res.send('API is working'));

// Start server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

models/CricketTeam.js

```
const mongoose = require('mongoose');
const cricketTeamSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    trim: true,
  },
  coach: {
    type: String,
    required: true,
    trim: true,
  },
  captain: {
    type: String,
    required: true,
```

```

    trim: true,
  },
});

module.exports = mongoose.model('CricketTeam', cricketTeamSchema);

```

routes/cricketTeams.js

```

const express = require('express');
const router = express.Router();
const CricketTeam = require('../models/CricketTeam');

// Get all teams
router.get('/', async (req, res) => {
  try {
    const teams = await CricketTeam.find();
    res.json(teams);
  } catch (err) {
    res.status(500).json({ message: 'Server error' });
  }
});

// Create a team
router.post('/', async (req, res) => {
  try {
    const team = new CricketTeam(req.body);
    await team.save();
    res.status(201).json(team);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// Update a team
router.put('/:id', async (req, res) => {
  try {
    const team = await CricketTeam.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
      runValidators: true,
    });
    if (!team) {
      return res.status(404).json({ message: 'Team not found' });
    }
    res.json(team);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// Delete a team
router.delete('/:id', async (req, res) => {
  try {
    const team = await CricketTeam.findByIdAndDelete(req.params.id);
    if (!team) {
      return res.status(404).json({ message: 'Team not found' });
    }
    res.json({ message: 'Team deleted' });
  } catch (err) {
    res.status(500).json({ message: 'Server error' });
  }
});

```

```
module.exports = router;
```

NOTES :

NPM is a package manager used to install, delete, and update Javascript packages on your machine. **NPX** is a package executer, and it is used to execute javascript packages directly, without installing them.

Axios is a popular JavaScript library used to make HTTP requests from the browser and Node.js to interact with APIs, while a **router** is a software component that manages the navigation within a web application by associating specific URL paths with their corresponding views or components. In essence, Axios handles communication with external servers, sending and receiving data, whereas a router handles internal navigation, showing different content to the user based on the URL they request or enter.

useState

- **Definition:** A React hook that lets you add and manage variables (state) inside functional components.
- **Analogy:**
 - Think of a **whiteboard in a classroom**. You can write something (initial state), then erase & update it (setState).

useNavigate (from React Router)

- **Definition:** A hook to navigate between different pages (routes) in React.
- **Analogy:**
 - Think of **Google Maps** – when you click “Go to Home”, it takes you to that location.

useEffect

- **Definition:** A hook to perform side effects (code that runs after render, like fetching data, setting timers, etc.).
- **Analogy:**
 - Imagine a **student entering a classroom**. After they enter (render), the **teacher gives homework** (side effect).

State Variables

- **Definition:** Variables that React “remembers” and re-renders UI when they change.
- **Normal variable** = like notes on a rough page (if you change them, nobody knows).

- **State variable** = like notes on a projector (when updated, everyone sees change immediately).

Stateful vs Stateless Components

- **Stateful Component**
 - A component that manages its own data (state).
 - Example: A **Login Form** that remembers username and password typed by the user.
- **Stateless Component**
 - A component that does not manage state; it just displays info.
 - Example: A **Navbar** that just shows links (Home, About, Contact).

Analogy:

- **Stateful** = A **water bottle** you carry, which can be filled/emptied (changes state).
- **Stateless** = A **paper cup** given at canteen, used once (no memory of previous state).

Const

- `const` is used to declare a **constant variable**.
- Once a variable is declared with `const`, you **cannot reassign** it to a different value.
- However, if it's an **object or array**, the **contents can be changed**, but the variable itself cannot point to a new object/array.

Analogy

- Imagine you're given a **permanent seat number in your exam hall**.
 - That seat number (like roll no. 21) can **never change** → this is `const`.
 - But you can **keep exam paper, hall ticket, stationary items, question paper** → this is modifying the contents inside (objects/arrays).

onChange

```
onChange={ (e) => setEmail(e.target.value) }
```

- Whenever user types, this function runs.
- `e.target.value` = whatever is typed in the box.
- `setEmail(...)` updates the React state with the new value.
- Keeps **UI (input box) and State (variable) always in sync**.

- Imagine a **notebook** (React state) and a **whiteboard** (input field).

- Whatever the teacher writes on the **whiteboard**, the student immediately copies into the **notebook** (`setEmail`).
- Whatever is in the notebook is always what's shown on the whiteboard (`value={email}`).

What is **async**?

- `async` is a keyword used in **JavaScript functions**.
 - When you mark a function as `async`, it means:
 - 👉 This function will **always return a Promise**.
 - 👉 Inside it, you can use `await` to pause execution until a task (like fetching data) is done.
-

What is **await**?

- `await` can **only be used inside an async function**.
- It tells JavaScript:
 - 👉 “*Wait for this Promise to finish, then continue with the next line.*”

Without `await`, the code runs immediately and may not wait for the result.

Analogy

- **Async function = You ordering popcorn 🍿 at the counter.**
- **Await = You wait at the counter until the popcorn is ready.**
- Without `await`, you order popcorn but walk into the theater → later someone shouts your name when popcorn is ready.