

GangadharGowdJogi

Final Project: MERN-Based Player Management Application

Overview:

This project is a full-stack web application built using the MERN stack (MongoDB, Express, React and Node.js) that allows users to manage a list of sports players. Users can perform the following actions:

- View all players
- Add a new player
- Update an existing player's information
- Delete a player from the list

The application has two main components:

- **Backend:** Built using Node.js and Express, with MongoDB for the database.
- **Frontend:** Built using React for an interactive and responsive user interface.

Project Structure

The project is divided into two main folders:

Backend (backend-MEN):

This is where the server-side logic resides. It includes the database connection, API routes, and player model for MongoDB.

- **models/playerModel.js:** Defines the schema (structure) for the player data stored in the MongoDB database.
- **routes/players.js:** Contains all the API routes (endpoints) for creating, reading, updating, and deleting player data.
- **server.js:** The main server file that sets up the Express application, connects to MongoDB, and handles routing.

Frontend (frontend-React+Vite):

This is where the client-side logic resides. It includes components for displaying and interacting with the player data.

- **components/PlayerComponent.jsx:** The main component that handles displaying the player table, adding players, updating players, and deleting players.
- **App.jsx:** The root component that renders the header and buttons for switching between views (view all players, add new player).
- **App.css** and **index.css:** Styling files for the frontend user interface.

Prerequisites

- Node.js and npm installed on your machine.
- MongoDB installed and running, or use a MongoDB cloud service like Atlas

Features

1. View All Players:

- When a user clicks on the "Display All Players" button, the application fetches all players from the backend and displays them in a table.
- Each row shows the player's name, number, position, and action buttons to either update or delete the player.

2. Add New Player:

- When a user clicks on the "Add New Player" button, an input form is displayed.
- The user can enter the player's name, number, and position.
- After submitting, the new player is saved to the MongoDB database via the backend API and the list of players is refreshed.

3. Update Player:

- If the user clicks the "Update" button next to a player, the form fields are pre-filled with the existing data of the player.
- The user can modify the details and submit the updated information.
- The application checks for any conflicts (e.g., if another player with the same name and number already exists) before saving the updated player.

4. Delete Player:

- The user can delete a player by clicking the "Delete" button next to the player's information in the table.
- This action sends a request to the backend to remove the player from the database, and the table is refreshed to show the updated list.

Backend (Node.js + Express + MongoDB)

Database (MongoDB)

- The application uses MongoDB as the database to store player information.
- A **player schema** is defined in the playerModel.js file, which includes the fields name, number, and position—all of which are required.

API Endpoints

1. **POST /api/players:** Adds a new player to the database. If the player already exists (same name and number), an error message is returned.
2. **GET /api/players:** Fetches all players from the database.

3. PATCH /api/players/

: Updates an existing player's details based on the provided player ID. Before updating, it checks if another player with the same name and number already exists to avoid duplicates.

4. DELETE /api/players/

: Deletes a player based on their ID.

Server (Express)

- The server is started in server.js and listens on port 3000. It connects to the MongoDB database using Mongoose and sets up the API routes to handle requests for player data.

Frontend (React + Axios)

React Components

- **App.jsx**: This is the main application file. It includes buttons to toggle between viewing all players and adding a new player. The component also manages the state for the page title (pageTitle), which is used to display different content based on user interaction.
- **PlayerComponent.jsx**:
 - This component handles the rendering of all player-related functionality, such as fetching and displaying the list of players, adding new players, updating existing ones, and deleting players.
 - **Axios** is used to make HTTP requests to the backend API to perform these actions.

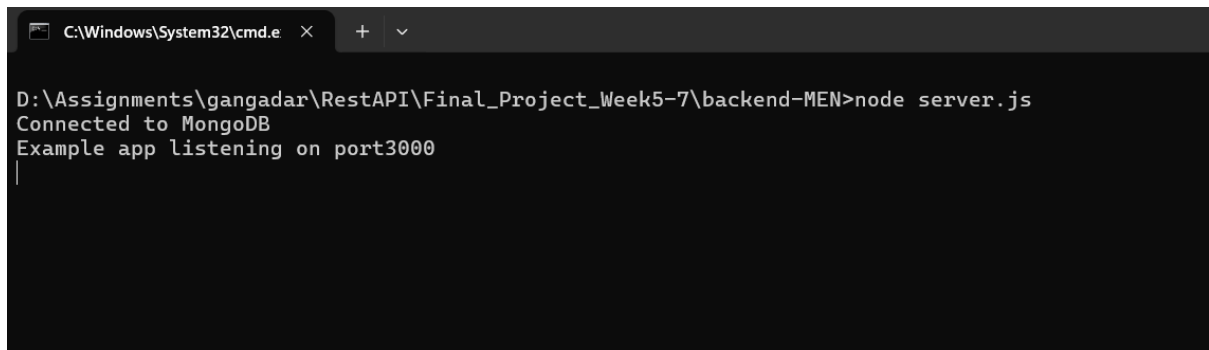
Forms for Adding, Updating, and Deleting Players

- **Add Form**: A simple form where users enter the player's name, number, and position to add a new player.
- **Update Form**: When updating, the form is pre-filled with the player's current details, allowing the user to make changes.
- **Delete Action**: The player can be deleted directly from the table with the click of a button.

In Conclusion, this project demonstrates how to build a simple MERN stack application where users can manage a list of sports players. It covers the essential CRUD (Create, Read, Update, Delete) operations using RESTful APIs and a user-friendly interface built with React. The architecture ensures clear separation of concerns, with the frontend handling the user interface and the backend managing data storage and business logic.

Screenshots:

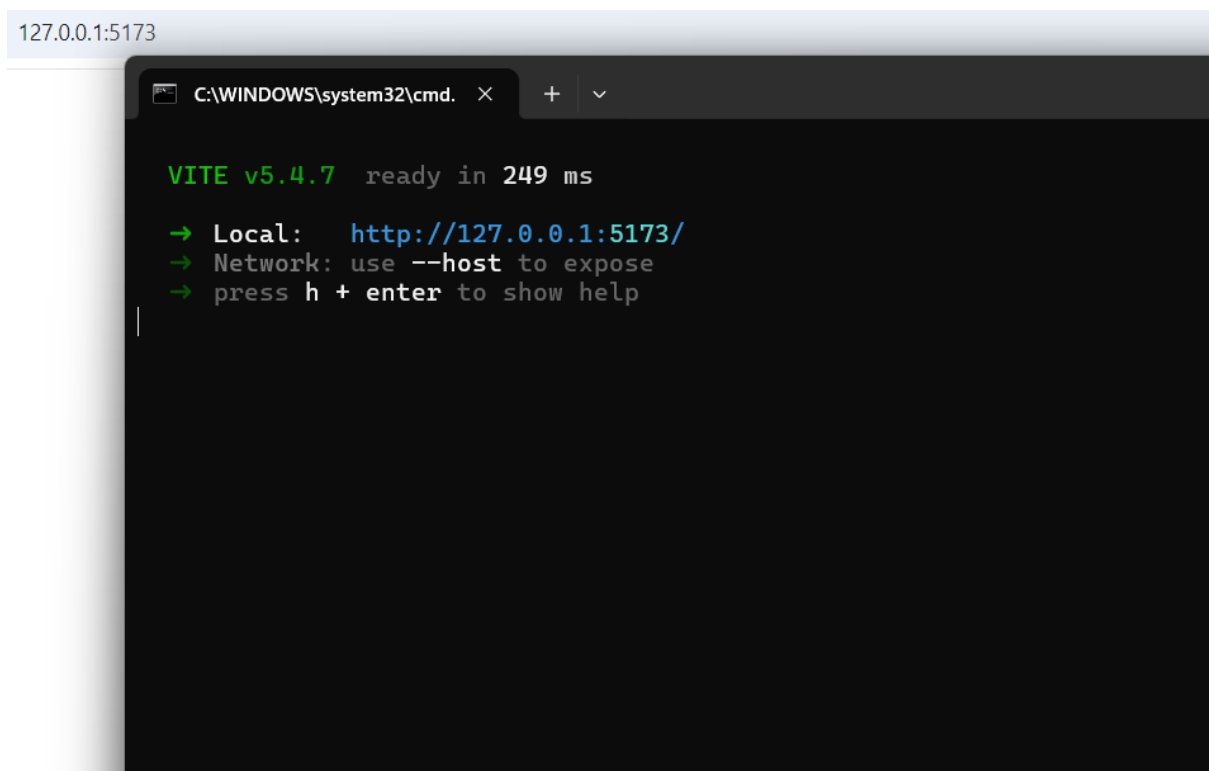
Backend node server started



```
C:\Windows\System32\cmd.e  X  +  v

D:\Assignments\gangadar\RestAPI\Final_Project_Week5-7\backend-MEN>node server.js
Connected to MongoDB
Example app listening on port3000
|
```

Front end – react+vite started



127.0.0.1:5173

```
C:\WINDOWS\system32\cmd.  X  +  v

VITE v5.4.7  ready in 249 ms

→ Local:   http://127.0.0.1:5173/
→ Network: use --host to expose
→ press h + enter to show help
|
```

Application



Sports - Players Details

Display All Players

Add New Player

Display All Players:



Sports - Players Details

Display All Players

Add New Player

All Players Information

Player Name	Player Number	Position	Action	
testuser	1	testrole	Update	Delete
testuser3Patch2	36	qwe	Update	Delete
Test123	46	Bowler	Update	Delete
Test123	465	12	Update	Delete
er	3	qw	Update	Delete
1	2	3	Update	Delete

Add new Player:



Sports - Players Details

Display All Players

Add New Player

Add New Player

<input type="text" value="Name"/>	<input type="text" value="Number"/>	<input type="text" value="Position"/>	<input type="button" value="Add Player"/>
-----------------------------------	-------------------------------------	---------------------------------------	---



Sports - Players Details

Display All Players

Add New Player

Add New Player

<input type="text" value="NewTestUser"/>	<input type="text" value="125"/>	<input type="text" value="Batsman"/>	<input type="button" value="Add Player"/>
--	----------------------------------	--------------------------------------	---

Display All Players



Sports - Players Details

Display All Players Add New Player

All Players Information

Player Name	Player Number	Position	Action	
testuser	1	testrole	Update	Delete
testuser3Patch2	36	qwe	Update	Delete
Test123	46	Bowler	Update	Delete
Test123	465	12	Update	Delete
er	3	qw	Update	Delete
1	2	3	Update	Delete
NewTestUser	125	Batsman	Update	Delete

Player already Exists alert



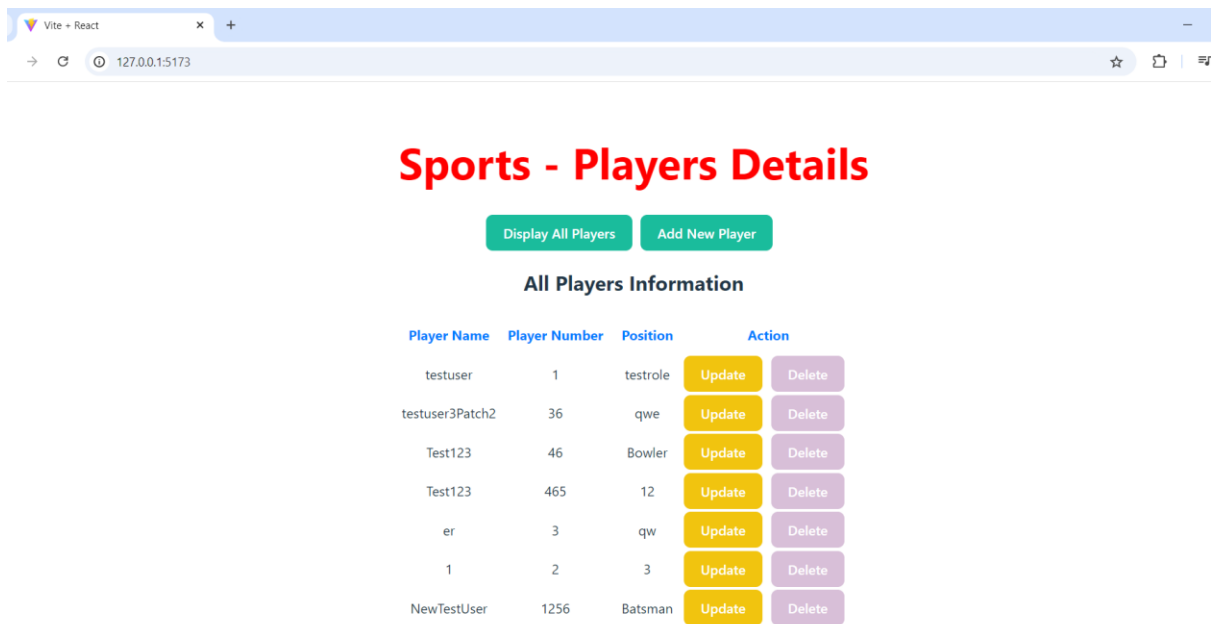
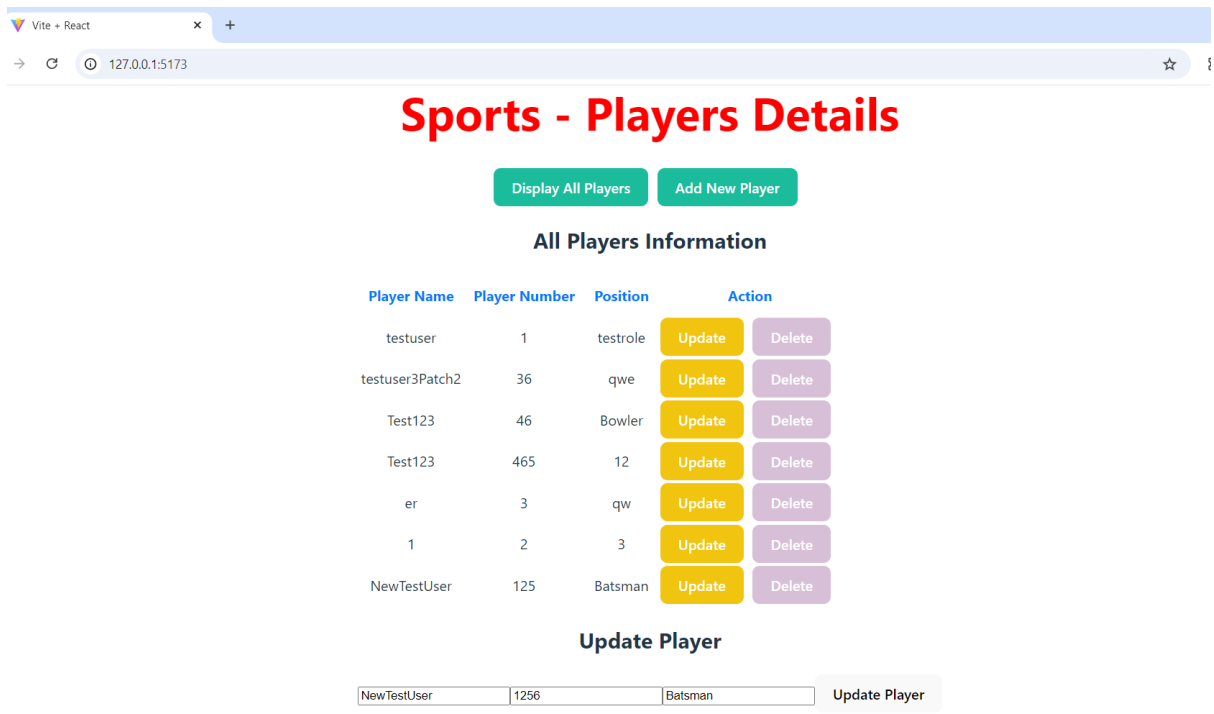
Spils

Display All Players Add New Player

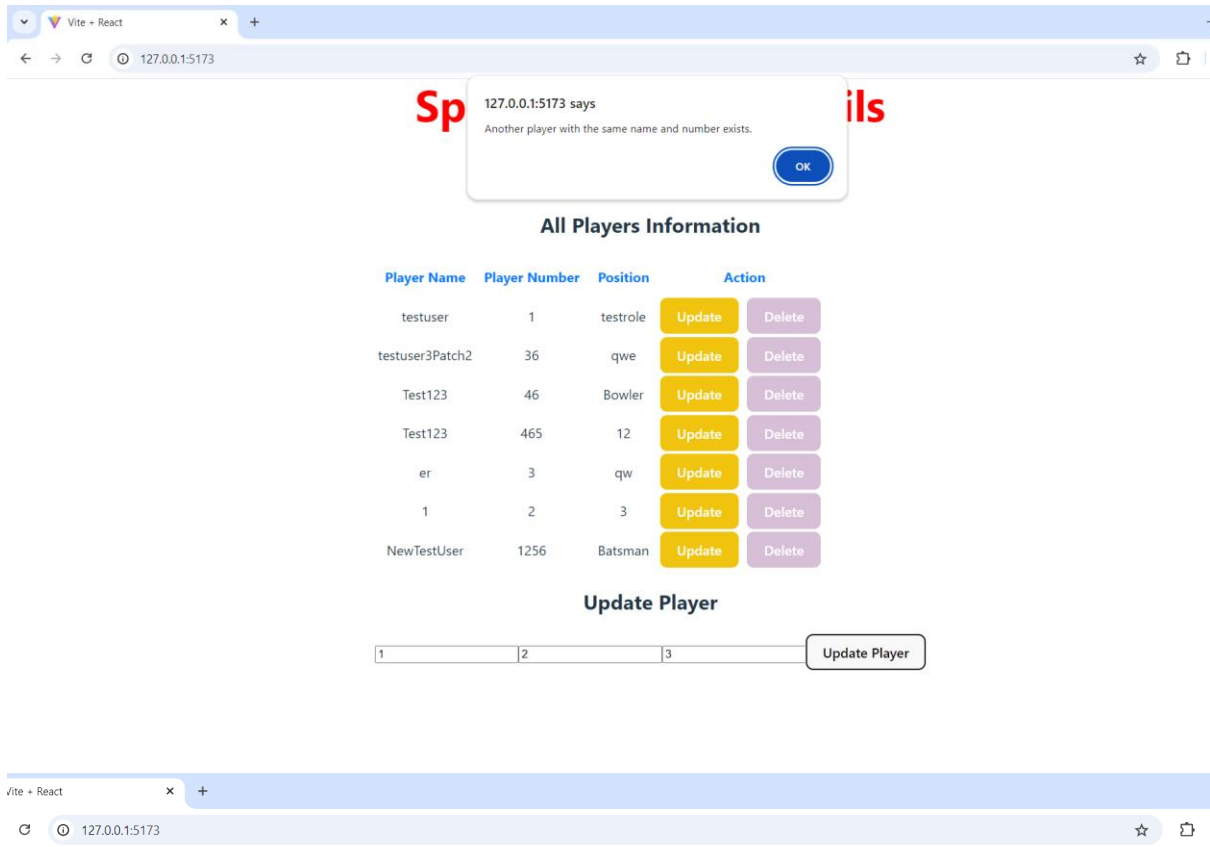
Add New Player

NewTestUser	125	Batsman	Add Player
-------------	-----	---------	------------

Update Player



Already Exists



Sports - Players Details

Display All Players

Add New Player

All Players Information

Player Name	Player Number	Position	Action
testuser	1	testrole	<button>Update</button> <button>Delete</button>
testuser3Patch2	36	qwe	<button>Update</button> <button>Delete</button>
Test123	46	Bowler	<button>Update</button> <button>Delete</button>
Test123	465	12	<button>Update</button> <button>Delete</button>
er	3	qw	<button>Update</button> <button>Delete</button>
1	2	3	<button>Update</button> <button>Delete</button>
14	2	34	<button>Update</button> <button>Delete</button>

Delete



Sports - Players Details

Display All Players

Add New Player

All Players Information

Player Name	Player Number	Position	Action	
testuser	1	testrole	<button>Update</button>	<button>Delete</button>
testuser3Patch2	36	qwe	<button>Update</button>	<button>Delete</button>
Test123	46	Bowler	<button>Update</button>	<button>Delete</button>
Test123	465	12	<button>Update</button>	<button>Delete</button>

```
C:\Windows\System32\cmd.e x + v
D:\Assignments\gangadar\RestAPI\Final_Project_Week5-7\backend-MEN>node server.js
Connected to MongoDB
Example app listening on port3000
/api/players GET
/api/players POST
/api/players GET
/api/players GET
/api/players POST
/api/players GET
/api/players/670bd1d6f4f96956091bf60b PATCH
/api/players GET
/api/players/670bd1d6f4f96956091bf60b PATCH
/api/players GET
/api/players/670bd1d6f4f96956091bf60b PATCH
/api/players GET
/api/players/670bd1d6f4f96956091bf60b PATCH
/api/players/670bd1d6f4f96956091bf60b PATCH
/api/players GET
/api/players/670bd0daee44f17e9765439d DELETE
/api/players GET
/api/players/670bc173ee44f17e9765436e DELETE
/api/players GET
/api/players/670bd1d6f4f96956091bf60b DELETE
/api/players GET
```