Correct it And also add All crud operation Put post delete

###### **ChatGPT said:**

Here is the **fully corrected SPController** with **all CRUD operations** (POST, GET, PUT, DELETE):

### **Corrected SPController (Full CRUD)**

csharp

CopyEdit

1. using Microsoft.AspNetCore.Mvc;
2. using Dapper;
3. using PES.WebAPI.Data;
4. using PES.WebAPI.Models;
5. [ApiController]
6. [Route("api/[controller]")]
7. public class SPController : ControllerBase
8. {
9. private readonly DapperContext \_context;
10. public SPController(DapperContext context)
11. {
12. \_context = context;
13. }
14. // ➡ CREATE (POST)
15. [HttpPost]
16. public async Task<IActionResult> Create([FromBody] SPModel model)
17. {
18. using var conn = \_context.CreateConnection();
19. // Validate Trustee\_ID exists
20. var trusteeExists = await conn.ExecuteScalarAsync<int>(
21. "SELECT COUNT(1) FROM Deepanshu\_Trustee WHERE Trustee\_ID = @Trustee\_ID",
22. new { Trustee\_ID = model.Trustee\_ID }
23. );
24. if (trusteeExists == 0)
25. {
26. return BadRequest($"Trustee\_ID {model.Trustee\_ID} does not exist.");
27. }
28. var sql = @"INSERT INTO Deepanshu\_SP (Name, Service\_Type, Trustee\_ID)
29. VALUES (@Name, @Service\_Type, @Trustee\_ID)";
30. await conn.ExecuteAsync(sql, model);
31. return Ok("SP record inserted successfully");
32. }
33. // ➡ READ (GET All)
34. [HttpGet]
35. public async Task<IActionResult> GetAll()
36. {
37. using var conn = \_context.CreateConnection();
38. var sql = "SELECT \* FROM Deepanshu\_SP WITH (NOLOCK)";
39. var data = await conn.QueryAsync(sql);
40. return Ok(data);
41. }
42. // ➡ READ (GET by ID)
43. [HttpGet("{id}")]
44. public async Task<IActionResult> GetById(int id)
45. {
46. using var conn = \_context.CreateConnection();
47. var sql = "SELECT \* FROM Deepanshu\_SP WITH (NOLOCK) WHERE SP\_ID = @SP\_ID";
48. var sp = await conn.QueryFirstOrDefaultAsync(sql, new { SP\_ID = id });
49. if (sp == null)
50. return NotFound($"SP record with ID {id} not found.");
51. return Ok(sp);
52. }
53. // ➡ UPDATE (PUT)
54. [HttpPut("{id}")]
55. public async Task<IActionResult> Update(int id, [FromBody] SPModel model)
56. {
57. using var conn = \_context.CreateConnection();
58. // Check if SP record exists
59. var exists = await conn.ExecuteScalarAsync<int>(
60. "SELECT COUNT(1) FROM Deepanshu\_SP WHERE SP\_ID = @SP\_ID", new { SP\_ID = id });
61. if (exists == 0)
62. return NotFound($"SP record with ID {id} not found.");
63. // Check Trustee\_ID exists
64. var trusteeExists = await conn.ExecuteScalarAsync<int>(
65. "SELECT COUNT(1) FROM Deepanshu\_Trustee WHERE Trustee\_ID = @Trustee\_ID",
66. new { Trustee\_ID = model.Trustee\_ID });
67. if (trusteeExists == 0)
68. return BadRequest($"Trustee\_ID {model.Trustee\_ID} does not exist.");
69. var sql = @"UPDATE Deepanshu\_SP
70. SET Name = @Name, Service\_Type = @Service\_Type, Trustee\_ID = @Trustee\_ID
71. WHERE SP\_ID = @SP\_ID";
72. await conn.ExecuteAsync(sql, new
73. {
74. model.Name,
75. model.Service\_Type,
76. model.Trustee\_ID,
77. SP\_ID = id
78. });
79. return Ok($"SP record with ID {id} updated successfully");
80. }
81. // ➡ DELETE
82. [HttpDelete("{id}")]
83. public async Task<IActionResult> Delete(int id)
84. {
85. using var conn = \_context.CreateConnection();
86. var sql = "DELETE FROM Deepanshu\_SP WHERE SP\_ID = @SP\_ID";
87. var rows = await conn.ExecuteAsync(sql, new { SP\_ID = id });
88. if (rows == 0)
89. return NotFound($"SP record with ID {id} not found.");
90. return Ok($"SP record with ID {id} deleted successfully");
91. }
92. }

### **SPModel.cs**

Make sure you have the **model class** for binding request body:

csharp

CopyEdit

1. namespace PES.WebAPI.Models
2. {
3. public class SPModel
4. {
5. public string Name { get; set; }
6. public string Service\_Type { get; set; }
7. public int Trustee\_ID { get; set; }
8. }
9. }

### **Endpoints available now**

1. **POST:** api/SP – Create
2. **GET:** api/SP – Get All
3. **GET:** api/SP/{id} – Get by ID
4. **PUT:** api/SP/{id} – Update
5. **DELETE:** api/SP/{id} – Delete

👉 Do you also want me to **do the same full CRUD for TrusteeController and AMLController** (with validation)?  
 Or should I **link them together** (e.g., deleting a Trustee deletes its SP records)?

##### **You said:**

Also for trustee and aml contorller

###### **ChatGPT said:**

Great! Here's the **full CRUD for TrusteeController and AMLController** with proper validation and relationships:

## **1️⃣ TrusteeController.cs**

csharp

CopyEdit

1. using Microsoft.AspNetCore.Mvc;
2. using Dapper;
3. using PES.WebAPI.Data;
4. using PES.WebAPI.Models;
5. [ApiController]
6. [Route("api/[controller]")]
7. public class TrusteeController : ControllerBase
8. {
9. private readonly DapperContext \_context;
10. public TrusteeController(DapperContext context)
11. {
12. \_context = context;
13. }
14. // ➡ CREATE (POST)
15. [HttpPost]
16. public async Task<IActionResult> Create([FromBody] TrusteeModel model)
17. {
18. using var conn = \_context.CreateConnection();
19. var sql = @"INSERT INTO Deepanshu\_Trustee (Name, Email, Phone, Plan\_Number)
20. VALUES (@Name, @Email, @Phone, @Plan\_Number)";
21. await conn.ExecuteAsync(sql, model);
22. return Ok("Trustee record inserted successfully");
23. }
24. // ➡ READ (GET All)
25. [HttpGet]
26. public async Task<IActionResult> GetAll()
27. {
28. using var conn = \_context.CreateConnection();
29. var sql = "SELECT \* FROM Deepanshu\_Trustee WITH (NOLOCK)";
30. var data = await conn.QueryAsync(sql);
31. return Ok(data);
32. }
33. // ➡ READ (GET by ID)
34. [HttpGet("{id}")]
35. public async Task<IActionResult> GetById(int id)
36. {
37. using var conn = \_context.CreateConnection();
38. var sql = "SELECT \* FROM Deepanshu\_Trustee WITH (NOLOCK) WHERE Trustee\_ID = @Trustee\_ID";
39. var trustee = await conn.QueryFirstOrDefaultAsync(sql, new { Trustee\_ID = id });
40. if (trustee == null)
41. return NotFound($"Trustee with ID {id} not found.");
42. return Ok(trustee);
43. }
44. // ➡ UPDATE (PUT)
45. [HttpPut("{id}")]
46. public async Task<IActionResult> Update(int id, [FromBody] TrusteeModel model)
47. {
48. using var conn = \_context.CreateConnection();
49. var exists = await conn.ExecuteScalarAsync<int>(
50. "SELECT COUNT(1) FROM Deepanshu\_Trustee WHERE Trustee\_ID = @Trustee\_ID", new { Trustee\_ID = id });
51. if (exists == 0)
52. return NotFound($"Trustee with ID {id} not found.");
53. var sql = @"UPDATE Deepanshu\_Trustee
54. SET Name = @Name, Email = @Email, Phone = @Phone, Plan\_Number = @Plan\_Number
55. WHERE Trustee\_ID = @Trustee\_ID";
56. await conn.ExecuteAsync(sql, new
57. {
58. model.Name,
59. model.Email,
60. model.Phone,
61. model.Plan\_Number,
62. Trustee\_ID = id
63. });
64. return Ok($"Trustee with ID {id} updated successfully");
65. }
66. // ➡ DELETE
67. [HttpDelete("{id}")]
68. public async Task<IActionResult> Delete(int id)
69. {
70. using var conn = \_context.CreateConnection();
71. // Delete SP records linked to this Trustee first (FK constraint)
72. await conn.ExecuteAsync("DELETE FROM Deepanshu\_SP WHERE Trustee\_ID = @Trustee\_ID", new { Trustee\_ID = id });
73. var sql = "DELETE FROM Deepanshu\_Trustee WHERE Trustee\_ID = @Trustee\_ID";
74. var rows = await conn.ExecuteAsync(sql, new { Trustee\_ID = id });
75. if (rows == 0)
76. return NotFound($"Trustee with ID {id} not found.");
77. return Ok($"Trustee with ID {id} deleted successfully");
78. }
79. }

## **2️⃣ AMLController.cs**

csharp

CopyEdit

1. using Microsoft.AspNetCore.Mvc;
2. using Dapper;
3. using PES.WebAPI.Data;
4. using PES.WebAPI.Models;
5. [ApiController]
6. [Route("api/[controller]")]
7. public class AMLController : ControllerBase
8. {
9. private readonly DapperContext \_context;
10. public AMLController(DapperContext context)
11. {
12. \_context = context;
13. }
14. // ➡ CREATE (POST)
15. [HttpPost]
16. public async Task<IActionResult> Create([FromBody] AMLModel model)
17. {
18. using var conn = \_context.CreateConnection();
19. var sql = @"INSERT INTO Deepanshu\_AML (Plan\_Number, Status, Compliance\_Date, Comments)
20. VALUES (@Plan\_Number, @Status, @Compliance\_Date, @Comments)";
21. await conn.ExecuteAsync(sql, model);
22. return Ok("AML record inserted successfully");
23. }
24. // ➡ READ (GET All)
25. [HttpGet]
26. public async Task<IActionResult> GetAll()
27. {
28. using var conn = \_context.CreateConnection();
29. var sql = "SELECT \* FROM Deepanshu\_AML WITH (NOLOCK)";
30. var data = await conn.QueryAsync(sql);
31. return Ok(data);
32. }
33. // ➡ READ (GET by ID)
34. [HttpGet("{id}")]
35. public async Task<IActionResult> GetById(int id)
36. {
37. using var conn = \_context.CreateConnection();
38. var sql = "SELECT \* FROM Deepanshu\_AML WITH (NOLOCK) WHERE AML\_ID = @AML\_ID";
39. var aml = await conn.QueryFirstOrDefaultAsync(sql, new { AML\_ID = id });
40. if (aml == null)
41. return NotFound($"AML record with ID {id} not found.");
42. return Ok(aml);
43. }
44. // ➡ UPDATE (PUT)
45. [HttpPut("{id}")]
46. public async Task<IActionResult> Update(int id, [FromBody] AMLModel model)
47. {
48. using var conn = \_context.CreateConnection();
49. var exists = await conn.ExecuteScalarAsync<int>(
50. "SELECT COUNT(1) FROM Deepanshu\_AML WHERE AML\_ID = @AML\_ID", new { AML\_ID = id });
51. if (exists == 0)
52. return NotFound($"AML record with ID {id} not found.");
53. var sql = @"UPDATE Deepanshu\_AML
54. SET Plan\_Number = @Plan\_Number, Status = @Status, Compliance\_Date = @Compliance\_Date, Comments = @Comments
55. WHERE AML\_ID = @AML\_ID";
56. await conn.ExecuteAsync(sql, new
57. {
58. model.Plan\_Number,
59. model.Status,
60. model.Compliance\_Date,
61. model.Comments,
62. AML\_ID = id
63. });
64. return Ok($"AML record with ID {id} updated successfully");
65. }
66. // ➡ DELETE
67. [HttpDelete("{id}")]
68. public async Task<IActionResult> Delete(int id)
69. {
70. using var conn = \_context.CreateConnection();
71. var sql = "DELETE FROM Deepanshu\_AML WHERE AML\_ID = @AML\_ID";
72. var rows = await conn.ExecuteAsync(sql, new { AML\_ID = id });
73. if (rows == 0)
74. return NotFound($"AML record with ID {id} not found.");
75. return Ok($"AML record with ID {id} deleted successfully");
76. }
77. }

## **3️⃣ Model Classes**

Create models for Trustee and AML:

csharp

CopyEdit

1. namespace PES.WebAPI.Models
2. {
3. public class TrusteeModel
4. {
5. public string Name { get; set; }
6. public string Email { get; set; }
7. public string? Phone { get; set; }
8. public string Plan\_Number { get; set; }
9. }
10. public class AMLModel
11. {
12. public string Plan\_Number { get; set; }
13. public string Status { get; set; }
14. public DateTime Compliance\_Date { get; set; }
15. public string? Comments { get; set; }
16. }
17. }

## **Now Available Endpoints**

### **Trustee**

* POST /api/Trustee
* GET /api/Trustee
* GET /api/Trustee/{id}
* PUT /api/Trustee/{id}
* DELETE /api/Trustee/{id} (also deletes child SP records)

### **AML**

* POST /api/AML
* GET /api/AML
* GET /api/AML/{id}
* PUT /api/AML/{id}
* DELETE /api/AML/{id}

👉 Do you also want me to **update the SPController delete logic** so that if the linked Trustee is deleted, all related SP records are handled properly **(cascading)**?

Or should I make **Swagger show sample JSON schema for all 3 controllers**?

##### **You said:**

And there is no schema for auth Also no page of login there for react app

###### **ChatGPT said:**

Good catch! ✅ Let's **fix Auth schema for Swagger** and **add a React Login page** with working Redux + API call.

## **1️⃣ Fix Auth schema in Swagger (.NET)**

Your current AuthController probably returns JWT without a proper model. We’ll add **models and attributes** so Swagger shows the schema:

### **Auth Models**

csharp

CopyEdit

1. namespace PES.WebAPI.Models
2. {
3. public class LoginRequest
4. {
5. public string Username { get; set; }
6. public string Password { get; set; }
7. }
8. public class LoginResponse
9. {
10. public string Token { get; set; }
11. public string[] Roles { get; set; }
12. }
13. }

### **AuthController**

csharp

CopyEdit

1. using Microsoft.AspNetCore.Mvc;
2. using PES.WebAPI.Models;
3. [ApiController]
4. [Route("api/[controller]")]
5. public class AuthController : ControllerBase
6. {
7. [HttpPost("login")]
8. [ProducesResponseType(typeof(LoginResponse), 200)] // <-- Adds schema to Swagger
9. public IActionResult Login([FromBody] LoginRequest request)
10. {
11. if (request.Username == "admin" && request.Password == "password")
12. {
13. // Generate JWT token logic here
14. return Ok(new LoginResponse
15. {
16. Token = "FAKE\_JWT\_TOKEN",
17. Roles = new[] { "Admin" }
18. });
19. }
20. return Unauthorized("Invalid credentials");
21. }
22. }

Now Swagger will show the schema for both **request** and **response** for /auth/login.

## **2️⃣ React Login Page (Redux + API)**

Here’s a **Login page with Redux Toolkit** to store token and roles.

### **src/redux/slices/authSlice.js**

*(Already present, we just confirm)*

javascript

CopyEdit

1. import { createSlice } from "@reduxjs/toolkit";
2. const authSlice = createSlice({
3. name: "auth",
4. initialState: { token: null, roles: [] },
5. reducers: {
6. setAuth: (state, action) => {
7. state.token = action.payload.token;
8. state.roles = action.payload.roles;
9. },
10. logout: (state) => {
11. state.token = null;
12. state.roles = [];
13. },
14. },
15. });
16. export const { setAuth, logout } = authSlice.actions;
17. export default authSlice.reducer;

### **src/redux/store.js**

javascript

CopyEdit

1. import { configureStore } from "@reduxjs/toolkit";
2. import authReducer from "./slices/authSlice";
3. export const store = configureStore({
4. reducer: {
5. auth: authReducer,
6. },
7. });

### **src/components/Login.jsx**

jsx

CopyEdit

1. import React, { useState } from "react";
2. import { Button, TextField, Container, Typography } from "@mui/material";
3. import { useDispatch } from "react-redux";
4. import { setAuth } from "../redux/slices/authSlice";
5. import axios from "axios";
6. export default function Login() {
7. const [username, setUsername] = useState("");
8. const [password, setPassword] = useState("");
9. const dispatch = useDispatch();
10. const handleLogin = async () => {
11. try {
12. const res = await axios.post("http://localhost:5841/api/Auth/login", {
13. username,
14. password,
15. });
16. localStorage.setItem("token", res.data.token);
17. localStorage.setItem("roles", JSON.stringify(res.data.roles));
18. dispatch(setAuth({ token: res.data.token, roles: res.data.roles }));
19. alert("Login Successful!");
20. } catch (err) {
21. alert("Invalid credentials!");
22. }
23. };
24. return (
25. <Container maxWidth="sm" style={{ marginTop: "80px" }}>
26. <Typography variant="h5" gutterBottom>
27. Login
28. </Typography>
29. <TextField
30. label="Username"
31. fullWidth
32. margin="normal"
33. value={username}
34. onChange={(e) => setUsername(e.target.value)}
35. />
36. <TextField
37. label="Password"
38. fullWidth
39. margin="normal"
40. type="password"
41. value={password}
42. onChange={(e) => setPassword(e.target.value)}
43. />
44. <Button
45. variant="contained"
46. fullWidth
47. style={{ marginTop: "20px" }}
48. onClick={handleLogin}
49. >
50. Login
51. </Button>
52. </Container>
53. );
54. }

### **src/App.js (Routing)**

jsx

CopyEdit

1. import React from "react";
2. import { BrowserRouter, Routes, Route } from "react-router-dom";
3. import Login from "./components/Login";
4. function App() {
5. return (
6. <BrowserRouter>
7. <Routes>
8. <Route path="/" element={<Login />} />
9. </Routes>
10. </BrowserRouter>
11. );
12. }
13. export default App;

## **3️⃣ What Happens Now?**

* Swagger **shows schema** for Auth login.
* React **has a working login page** (http://localhost:3000/) that calls api/Auth/login.
* Token & roles are stored in **Redux + localStorage**.
* You can protect other pages using useSelector((state) => state.auth.token).

👉 Do you want me to **add Protected Routes (redirect if not logged in)** and a **Logout button** in React?

Or should I also **connect this login with the .NET Identity (real DB users)** instead of hardcoded username/password?

##### **You said:**

there is no schema for auth And also give database table And also add some data in all table

###### **ChatGPT said:**

Perfect! Let’s **add the missing schema for Auth**, create a **database table for users**, and **insert sample data for all your tables** (AML, Trustee, SP, and Users).

## **1️⃣ Database Table for Auth (Users Table)**

Create a table to store login credentials:

sql

CopyEdit

1. CREATE TABLE Deepanshu\_Users (
2. User\_ID INT IDENTITY(1,1) PRIMARY KEY,
3. Username VARCHAR(50) UNIQUE NOT NULL,
4. PasswordHash VARCHAR(255) NOT NULL,
5. Role VARCHAR(50) NOT NULL,
6. Created\_On DATETIME DEFAULT GETDATE()
7. );

## **2️⃣ Update Your AuthController for DB Users**

Instead of hardcoding credentials, validate against Deepanshu\_Users table:

csharp

CopyEdit

1. using Microsoft.AspNetCore.Mvc;
2. using Dapper;
3. using PES.WebAPI.Data;
4. using PES.WebAPI.Models;
5. [ApiController]
6. [Route("api/[controller]")]
7. public class AuthController : ControllerBase
8. {
9. private readonly DapperContext \_context;
10. public AuthController(DapperContext context)
11. {
12. \_context = context;
13. }
14. [HttpPost("login")]
15. [ProducesResponseType(typeof(LoginResponse), 200)]
16. public async Task<IActionResult> Login([FromBody] LoginRequest request)
17. {
18. using var conn = \_context.CreateConnection();
19. var user = await conn.QueryFirstOrDefaultAsync<dynamic>(
20. "SELECT \* FROM Deepanshu\_Users WHERE Username = @Username", new { request.Username });
21. if (user == null)
22. return Unauthorized("User not found!");
23. // For simplicity, comparing plain password (later hash it)
24. if (user.PasswordHash != request.Password)
25. return Unauthorized("Invalid password!");
26. return Ok(new LoginResponse
27. {
28. Token = "FAKE\_JWT\_" + user.Username, // Replace with real JWT
29. Roles = new[] { user.Role }
30. });
31. }
32. }

### **Models (same as earlier)**

csharp

CopyEdit

1. public class LoginRequest
2. {
3. public string Username { get; set; }
4. public string Password { get; set; }
5. }
6. public class LoginResponse
7. {
8. public string Token { get; set; }
9. public string[] Roles { get; set; }
10. }

## **3️⃣ Sample Data for All Tables**

### **Users**

sql

CopyEdit

1. INSERT INTO Deepanshu\_Users (Username, PasswordHash, Role)
2. VALUES
3. ('admin', 'admin123', 'Admin'),
4. ('manager', 'manager123', 'Manager');

### **Trustee**

sql

CopyEdit

1. INSERT INTO Deepanshu\_Trustee (Name, Email, Phone, Plan\_Number)
2. VALUES
3. ('John Trustee', 'john@example.com', '9876543210', 'P123'),
4. ('Mary Trustee', 'mary@example.com', '9988776655', 'P124');

### **AML**

sql

CopyEdit

1. INSERT INTO Deepanshu\_AML (Plan\_Number, Status, Compliance\_Date, Comments)
2. VALUES
3. ('P123', 'Compliant', GETDATE(), 'No issues'),
4. ('P124', 'Pending', GETDATE(), 'Audit scheduled');

### **SP**

sql

CopyEdit

1. INSERT INTO Deepanshu\_SP (Name, Service\_Type, Trustee\_ID)
2. VALUES
3. ('Service Partner A', 'Investment', 1),
4. ('Service Partner B', 'Consulting', 2);

## **4️⃣ Swagger Will Show Auth Schema**

Because we used:

csharp

CopyEdit

1. [ProducesResponseType(typeof(LoginResponse), 200)]

Swagger will **show request & response schema** for /api/Auth/login.

## **Next Steps**

1. Do you want me to **update CRUD Controllers (Trustee, AML, SP)** so they return foreign key joined data?
2. Should I **hash passwords** (instead of plain text) for Deepanshu\_Users?
3. Do you want **React Login** to validate with this real DB user table (with JWT)?

👉 Also, should I **add default Admin login in React app** with redirect to dashboard after login?

Tools

ChatGPT can make mistakes. Check important info. See Cookie Preferences.