Training Day 21

TOPICS COVERED

- **CRUD Operations using Mongoose**

CRUD stands for Create, Read, Update, and Delete which represent the four basic operations that can be performed on data in a database. These operations are fundamental to any web application that interacts with a database, such as user management, blog posting, or product inventory systems.

1. Create

The Create operation is used to add new data or records to the database. It corresponds to the HTTP POST method.

Example: Adding a new user to the system.

```
User.create({ name: "abc", email: "abc@example.com" });
```

2. Read

The Read operation is used to retrieve or fetch data from the database without modifying it. It corresponds to the HTTP GET method.

Example: Fetching all users or a specific user by ID.

```
User.find(); // All users
User.findById(id); // User by ID
```

3. Update

The Update operation is used to modify existing data in the database. It corresponds to the HTTP PUT or PATCH methods.

Example: Updating a user's name or email.

```
User.findByIdAndUpdate(id, { name: "Updated Name" }, { new: true });
```

4. Delete

The Delete operation is used to remove data from the database. It corresponds to the HTTP DELETE method.

Example: Removing a user from the system.

```
User.findByIdAndDelete(id);
```

- **Token-Based Authentication (JWT)**

JWT (JSON Web Token) is used to securely transmit information between parties. Commonly used for login sessions.

Installation:

```
npm install jsonwebtoken
```

Generate Token:

```
const jwt = require('jsonwebtoken');
const token = jwt.sign({ id: user._id }, "secretKey", { expiresIn: '1h' });
```

Verify Token:

```
const decoded = jwt.verify(token, "secretKey");
```

- **Header-Based Authentication**

Clients must send the token in the headers when accessing protected routes.

Backend Middleware:

```
const jwt = require('jsonwebtoken');

const authMiddleware = (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith("Bearer ")) {
    return res.status(401).json({ message: "Access denied. No token provided." });
  }

  const token = authHeader.split(" ")[1];

  try {
    const decoded = jwt.verify(token, "secretKey");
    req.user = decoded;
    next();
  } catch (err) {
    return res.status(401).json({ message: "Invalid token." });
  }
};
```

Use this middleware in protected routes:

```
router.get('/profile', authMiddleware, (req, res) => {
  res.json({ message: "Welcome to your profile", user: req.user });
});
```

TOOLS USED

VS Code

Postman

MongoDB Compass

jsonwebtoken

Browser Console / DevTools