Aim: Build RESTful API using MongoDB

Requirements:

- Node.js (v14 or above)
- Express.js
- MongoDB
- MongoDB Compass (optional, for GUI-based DB interaction)
- Postman (for API testing)

Theory:

A **RESTful API (Representational State Transfer)** is an architectural style that uses HTTP methods for communication between a client and server. It enables operations such as **Create, Read, Update**, and **Delete (CRUD)**.

Core Concepts:

- **Node.js:** JavaScript runtime for server-side development.
- **Express.js:** A web application framework for Node.js.
- **MongoDB:** NoSQL database that stores data in JSON-like documents.
- **Mongoose:** ODM (Object Data Modeling) library for MongoDB and Node.js, used to interact with MongoDB using schema-based models.

HTTP Methods Used:

- **GET:** Retrieve data
- **POST:** Add new data
- **PUT:** Update existing data
- **DELETE:** Remove data

Code:

Step 1: Initialize project

```
mkdir rest-api-mongo
cd rest-api-mongo
npm init -y
npm install express mongoose body-parser
```

Step 2: Basic file structure

```
rest-api-mongo/
│
├── server.js
└── models/
    └── user.js
```

Step 3: Create User Model (`models/user.js`)

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    name: String,
    email: String,
    age: Number
});
```

```
module.exports = mongoose.model('User', userSchema);
```

Step 4: Create REST API (`server.js`)

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const User = require('./models/user');

const app = express();
app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/webx0api', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => console.log('MongoDB Connected'))
  .catch(err => console.log(err));

// CREATE
app.post('/users', async (req, res) => {
    const user = new User(req.body);
    await user.save();
    res.send(user);
});

// READ
app.get('/users', async (req, res) => {
    const users = await User.find();
    res.send(users);
});

// UPDATE
app.put('/users/:id', async (req, res) => {
    const user = await User.findByIdAndUpdate(req.params.id, req.body, {
new: true });
    res.send(user);
});

// DELETE
app.delete('/users/:id', async (req, res) => {
    await User.findByIdAndDelete(req.params.id);
    res.send({ message: 'User deleted' });
});

app.listen(3000, () => {
    console.log('Server is running on port 3000');
});
```

Step 5: Run the server

```
node server.js
```

Testing API:

Use **Postman** or any API client to test the following endpoints:

- `POST /users` – Create a new user
- `GET /users` – Retrieve all users
- `PUT /users/:id` – Update a user by ID
- `DELETE /users/:id` – Delete a user by ID

Conclusion:

In this practical, we successfully created a **RESTful API using Node.js, Express, and MongoDB**. The API performs all CRUD operations and is scalable for future enhancements. This setup serves as the foundation for building modern web applications where backend APIs interact with frontend interfaces.