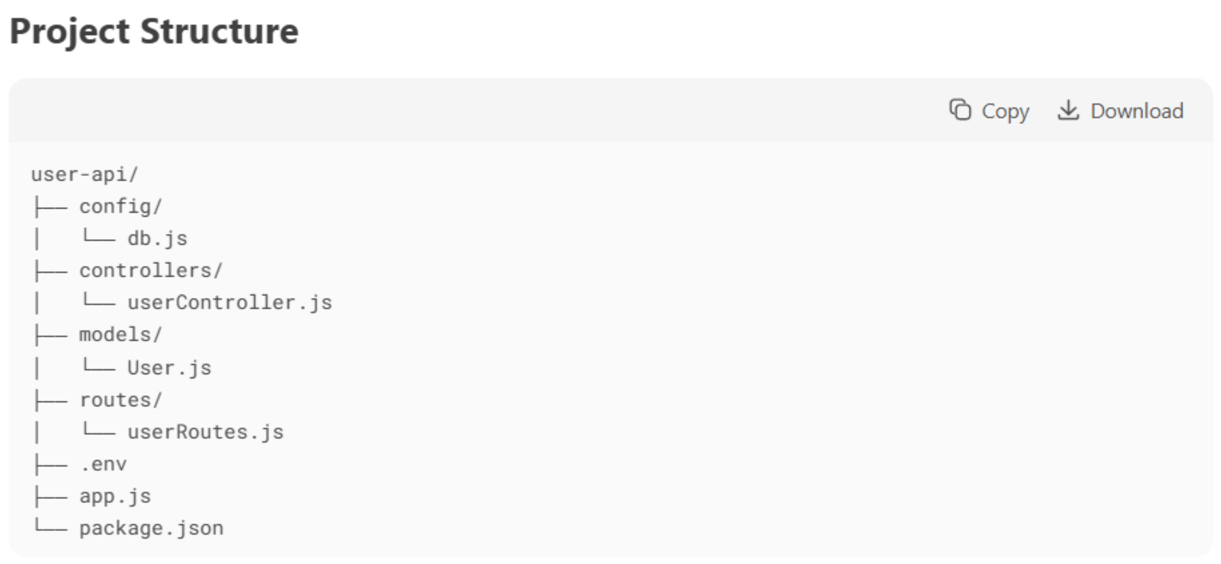
Complete Node.js + Express + MongoDB CRUD API

Here's a **full project** with user **Create, Read, Update, Delete (CRUD)** and **Search** functionality using MongoDB Atlas.



Project Structure

```
user-api/
├── config/
│   └── db.js
├── controllers/
│   └── userController.js
├── models/
│   └── User.js
├── routes/
│   └── userRoutes.js
├── .env
├── app.js
└── package.json
```

\

1. Setup & Installation

```
# Create project
Create a new folder user-api, and open this folder in VS Code
cd user-api
npm init -y

# Install dependencies
npm install express mongoose dotenv cors
```

2. Configure MongoDB Atlas

1. Get your connection string from MongoDB Atlas
   (Format: `mongodb+srv://<username>:<password>@cluster0.xxx.mongodb.net/<dbname>?retryWrites=true&w=majority`)

2. Create `.env` file:

```
MONGO_URI=
    mongodb+srv://username:password@cluster0.q0wlyup.mongodb.net/UserDataBase?retryWrites=tru
    e&w=majority&appName=Cluster0
    PORT=5000
```

3. Database Connection (`config/db.js`)

javascript

Copy

Download

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

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log('MongoDB Connected!');
  } catch (err) {
    console.error('Connection error:', err.message);
    process.exit(1);
  }
};

module.exports = connectDB;
```

4. User Model (`models/User.js`)

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

const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    lowercase: true
  },
  age: {
    type: Number,
    min: 18
  }
}, { timestamps: true });

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

5. User Controller (`controllers/userController.js`)

```
const User = require('../models/User');

// Create User
exports.createUser = async (req, res) => {
  try {
    const user = new User(req.body);
    await user.save();
    res.status(201).json(user);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
};

// Get All Users
exports.getAllUsers = async (req, res) => {
  try {
    const users = await User.find();
    res.json(users);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// Get Single User
exports.getUser = async (req, res) => {
  try {
    const user = await User.findById(req.params.id);
    if (!user) return res.status(404).json({ error: 'User not found' });
    res.json(user);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// Update User
```

```
exports.updateUser = async (req, res) => {
  try {
    const user = await User.findByIdAndUpdate(req.params.id, req.body, {
      new: true,
      runValidators: true
    });
    if (!user) return res.status(404).json({ error: 'User not found' });
    res.json(user);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
};

// Delete User
exports.deleteUser = async (req, res) => {
  try {
    const user = await User.findByIdAndDelete(req.params.id);
    if (!user) return res.status(404).json({ error: 'User not found' });
    res.json({ message: 'User deleted successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// Search Users
exports.searchUsers = async (req, res) => {
  try {
    const { name, email } = req.query;
    const filter = {};
    if (name) filter.name = { $regex: name, $options: 'i' };
    if (email) filter.email = { $regex: email, $options: 'i' };

    const users = await User.find(filter);
    res.json(users);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

6. User Routes (`routes/userRoutes.js`)

```
const express = require('express');
const router = express.Router();
const {
  createUser,
  getAllUsers,
  getUser,
  updateUser,
  deleteUser,
  searchUsers
} = require('../controllers/userController');

// CRUD Routes
router.post('/', createUser);
router.get('/', getAllUsers);
router.get('/search', searchUsers); // /api/users/search?name=John
router.get('/:id', getUser);
router.patch('/:id', updateUser);
router.delete('/:id', deleteUser);

module.exports = router;
```

7. Main Server (`app.js`)

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const connectDB = require('./config/db');
const userRoutes = require('./routes/userRoutes');

const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// Database Connection
connectDB();

// Routes
app.use('/api/users', userRoutes);

// Health Check
app.get('/', (req, res) => {
  res.send('User API is running!');
});

// Start Server
```

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

8. Run the Project

```
node app.js
```

API Endpoints

Method	Endpoint	Description
POST	`/api/users`	Create a new user
GET	`/api/users`	Get all users
GET	`/api/users/search?name=John`	Search users by name/email
GET	`/api/users/:id`	Get single user
PATCH	`/api/users/:id`	Update user
DELETE	`/api/users/:id`	Delete user

9. Test API

Download Postman https://www.postman.com/downloads/

10. Consume API with front End using React

Last step is to implement this API with your front-end build in lecture # 13