

## server.js

```
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const cors = require('cors');
4  const User = require('./models/User');
5  // const uri =
   "mongodb+srv://tanishkhot29:XvyCNzRNtwkKm0Q1@wadl.cxhbx.mongodb.net/userdb?
   retryWrites=true&w=majority";
6
7  const uri = "mongodb://localhost:27017/userdb";
8
9  const app = express();
10 app.use(express.json());
11 app.use(cors({
12   origin: 'http://localhost:3000',
13   methods: ['GET', 'POST', 'PUT', 'DELETE'],
14   credentials: true,
15   allowedHeaders: ['Content-Type', 'Authorization']
16 }));
17
18
19 // Update the mongoose connection
20 // mongoose.connect(uri, {
21 //   serverSelectionTimeoutMS: 5000,
22 //   socketTimeoutMS: 45000,
23 //   family: 4
24 // })
25 // .then(() => console.log('Connected to MongoDB'))
26 // .catch(err => console.error('MongoDB connection error:', err));
27
28 mongoose.connect(uri)
29   .then(() => console.log('Connected to MongoDB'))
30   .catch(err => console.error('MongoDB connection error:', err));
31
32 app.post('/api/register', async (req, res) => {
33   try {
34     const { name, email, password } = req.body;
35
36     const existingUser = await User.findOne({ email });
37     if (existingUser) {
38       return res.status(400).json({ message: 'User already exists' });
39     }
40
41     const user = new User({
42       name,
43       email,
44       password
45     });
46
47     await user.save();
48     res.status(201).json({ message: 'User registered successfully' });
49   } catch (error) {
50     res.status(500).json({ message: 'Server error' });
```

```
51   }
52 });
53
54 app.post('/api/login', async (req, res) => {
55   try {
56     const { email, password } = req.body;
57
58     const user = await User.findOne({ email, password });
59     if (!user) {
60       return res.status(400).json({ message: 'Invalid credentials' });
61     }
62
63     res.json({ message: 'Login successful', userId: user._id });
64   } catch (error) {
65     res.status(500).json({ message: 'Server error' });
66   }
67 });
68
69 app.get('/api/user/:id', async (req, res) => {
70   try {
71     const user = await User.findById(req.params.id);
72     if (!user) {
73       return res.status(404).json({ message: 'User not found' });
74     }
75     res.json(user);
76   } catch (error) {
77     res.status(500).json({ message: 'Server error' });
78   }
79 });
80
81 app.put('/api/user/:id', async (req, res) => {
82   try {
83     const { name, email } = req.body;
84     const user = await User.findByIdAndUpdate(
85       req.params.id,
86       { name, email },
87       { new: true }
88     );
89
90     if (!user) {
91       return res.status(404).json({ message: 'User not found' });
92     }
93     res.json(user);
94   } catch (error) {
95     res.status(500).json({ message: 'Server error' });
96   }
97 });
98
99 app.get('/api/users', async (req, res) => {
100   try {
101     const users = await User.find().select('-password');
102     res.json(users);
103   } catch (error) {
104     res.status(500).json({ message: 'Server error' });
105   }
106 });
```

```
105 |     }  
106 |   });  
107 |  
108 |   app.use(express.static('public'));  
109 |  
110 |   const PORT = 3000;  
111 |   app.listen(PORT, () => {  
112 |     console.log(`Server running on port ${PORT}`);  
113 |   });
```