

Assignment_3b\controllers\userController.js

```
1  const User = require('../models/User');
2
3  // @desc    Register user
4  // @route    POST /api/users/register
5  // @access   Public
6  exports.registerUser = async (req, res) => {
7    try {
8      const { name, email, password } = req.body;
9
10     // Check if user exists
11     const userExists = await User.findOne({ email });
12
13     if (userExists) {
14       return res.status(400).json({
15         success: false,
16         message: 'User already exists',
17       });
18     }
19
20     // Create user
21     const user = await User.create({
22       name,
23       email,
24       password,
25     });
26
27     // Generate token
28     const token = user.getSignedJwtToken();
29
30     res.status(201).json({
31       success: true,
32       token,
33       user: {
34         id: user._id,
35         name: user.name,
36         email: user.email,
37       },
38     });
39   } catch (error) {
40     res.status(500).json({
41       success: false,
42       message: 'Server Error',
43       error: error.message,
44     });
45   }
46 };
47
48 // @desc    Login user
49 // @route    POST /api/users/login
50 // @access   Public
51 exports.loginUser = async (req, res) => {
52   try {
53     const { email, password } = req.body;
54
55     // Validate email & password
56     if (!email || !password) {
57       return res.status(400).json({
```

```
58     success: false,
59     message: 'Please provide an email and password',
60   });
61 }
62
63 // Check for user
64 const user = await User.findOne({ email }).select('+password');
65
66 if (!user) {
67   return res.status(401).json({
68     success: false,
69     message: 'Invalid credentials',
70   });
71 }
72
73 // Check if password matches
74 const isMatch = await user.matchPassword(password);
75
76 if (!isMatch) {
77   return res.status(401).json({
78     success: false,
79     message: 'Invalid credentials',
80   });
81 }
82
83 // Generate token
84 const token = user.getSignedJwtToken();
85
86 res.status(200).json({
87   success: true,
88   token,
89   user: {
90     id: user._id,
91     name: user.name,
92     email: user.email,
93   },
94 });
95 } catch (error) {
96   res.status(500).json({
97     success: false,
98     message: 'Server Error',
99     error: error.message,
100   });
101 }
102 };
103
104 // @desc    Get user data
105 // @route    GET /api/users/me
106 // @access   Private
107 exports.getUserProfile = async (req, res) => {
108   try {
109     const user = await User.findById(req.user.id);
110
111     res.status(200).json({
112       success: true,
113       user: {
114         id: user._id,
115         name: user.name,
116         email: user.email,
117         createdAt: user.createdAt,
```

```
118     },
119   });
120 } catch (error) {
121   res.status(500).json({
122     success: false,
123     message: 'Server Error',
124     error: error.message,
125   });
126 }
127 };
128
129 // @desc    Update user profile
130 // @route    PUT /api/users/me
131 // @access   Private
132 exports.updateUserProfile = async (req, res) => {
133   try {
134     const updatedUser = await User.findByIdAndUpdate(
135       req.user.id,
136       { $set: req.body },
137       { new: true, runValidators: true }
138     );
139
140     res.status(200).json({
141       success: true,
142       user: {
143         id: updatedUser._id,
144         name: updatedUser.name,
145         email: updatedUser.email,
146         createdAt: updatedUser.createdAt,
147       },
148     });
149   } catch (error) {
150     res.status(500).json({
151       success: false,
152       message: 'Server Error',
153       error: error.message,
154     });
155   }
156 };
```