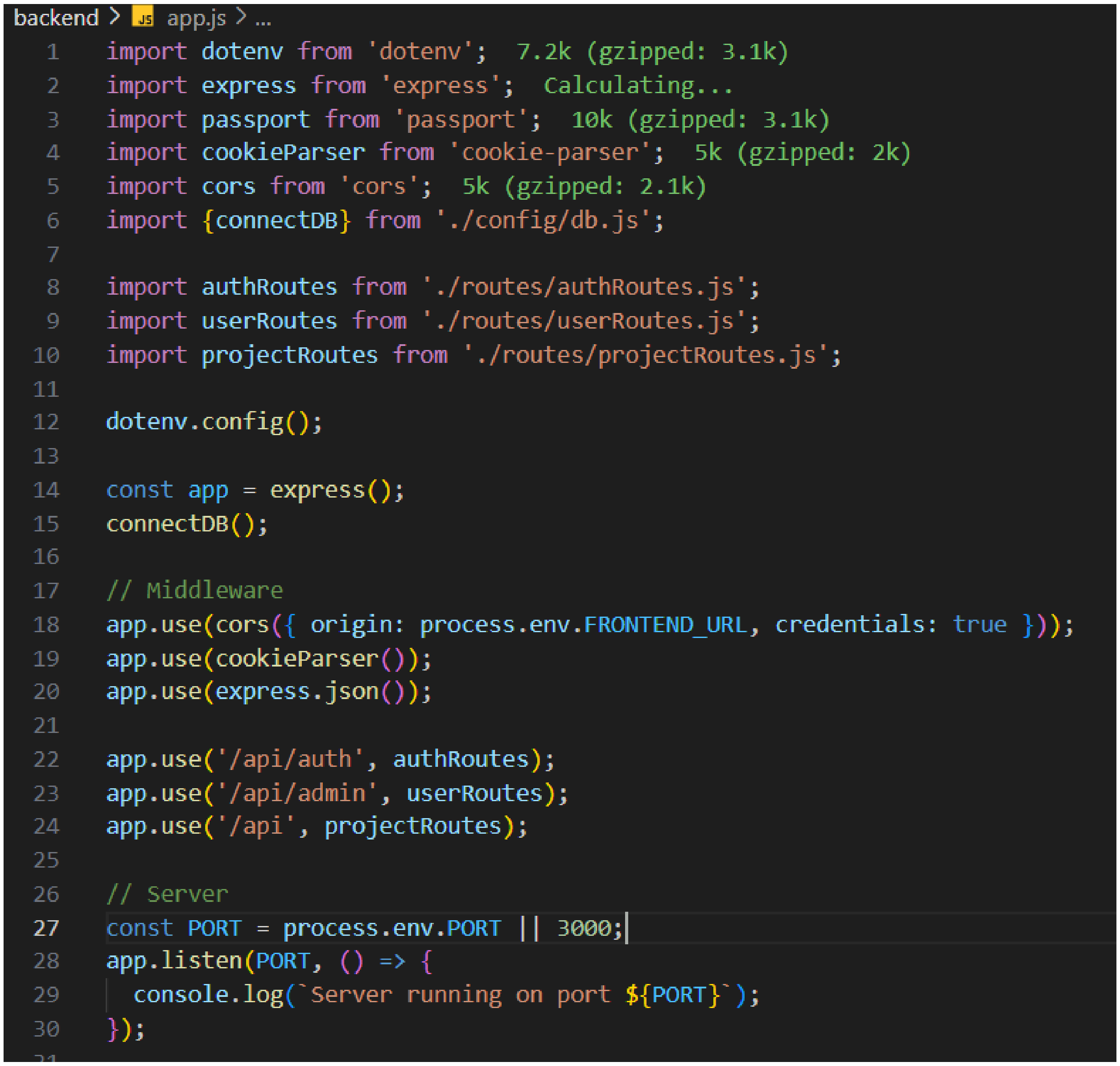
Experiment No - 05

Roll No.	52
Name	Kshitij Nangare
Class	D15B
Subject	Full Stack Development
Lab Outcome	L1, L2, L3
Date of Performance / Submission	18/08/2025 22/09/2025
Signature & Grades	

Experiment 5

Aim : Create secure, production-ready RESTful APIs

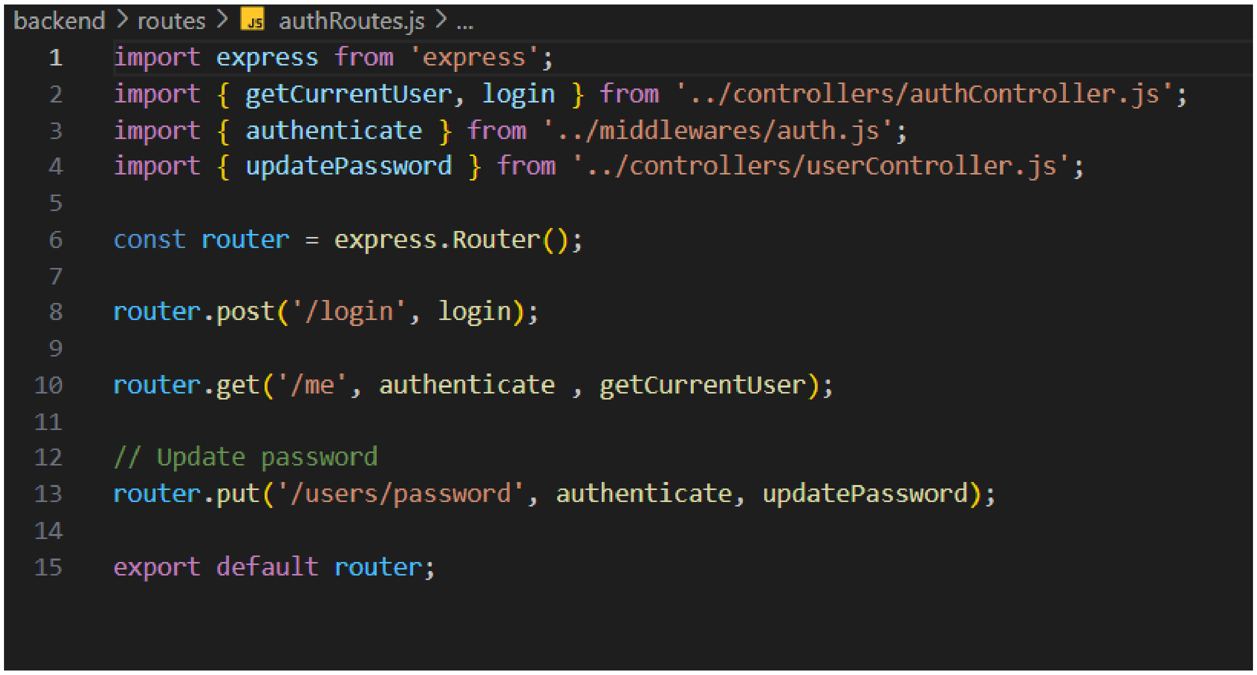
Code:



backend > app.js > ...

```
import dotenv from 'dotenv';  7.2k (gzipped: 3.1k)
import express from 'express';  Calculating...
import passport from 'passport';  10k (gzipped: 3.1k)
import cookieParser from 'cookie-parser';  5k (gzipped: 2k)
import cors from 'cors';  5k (gzipped: 2.1k)
import {connectDB} from './config/db.js';

import authRoutes from './routes/authRoutes.js';
import userRoutes from './routes/userRoutes.js';
import projectRoutes from './routes/projectRoutes.js';

dotenv.config();

const app = express();
connectDB();

// Middleware
app.use(cors({ origin: process.env.FRONTEND_URL, credentials: true }));
app.use(cookieParser());
app.use(express.json());

app.use('/api/auth', authRoutes);
app.use('/api/admin', userRoutes);
app.use('/api', projectRoutes);

// Server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

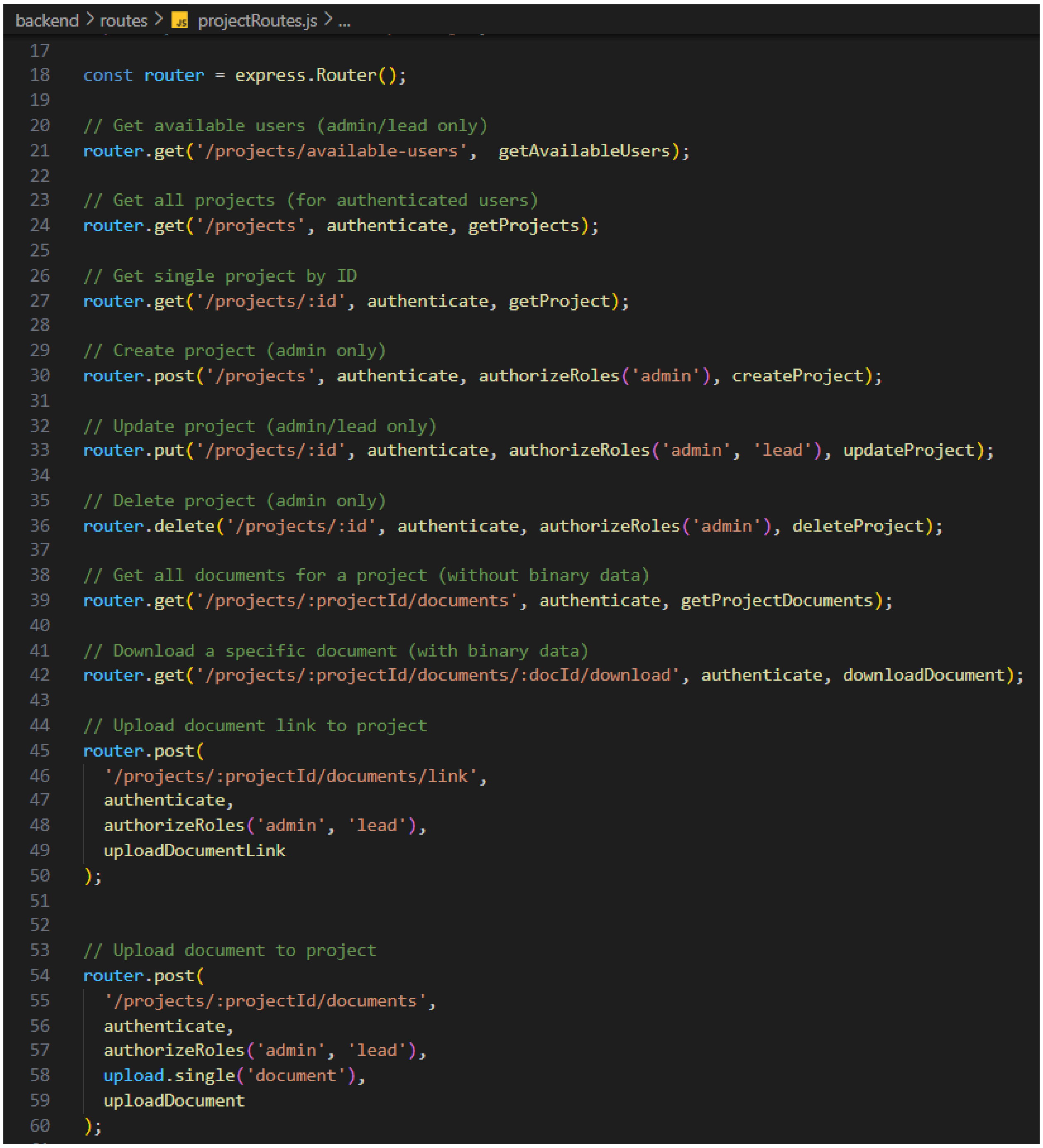
Figure 1



backend > routes > authRoutes.js > ...

```javascript
import express from 'express';
import { getCurrentUser, login } from '../controllers/authController.js';
import { authenticate } from '../middlewares/auth.js';
import { updatePassword } from '../controllers/userController.js';

const router = express.Router();

router.post('/login', login);

router.get('/me', authenticate , getCurrentUser);

// Update password
router.put('/users/password', authenticate, updatePassword);

export default router;
```

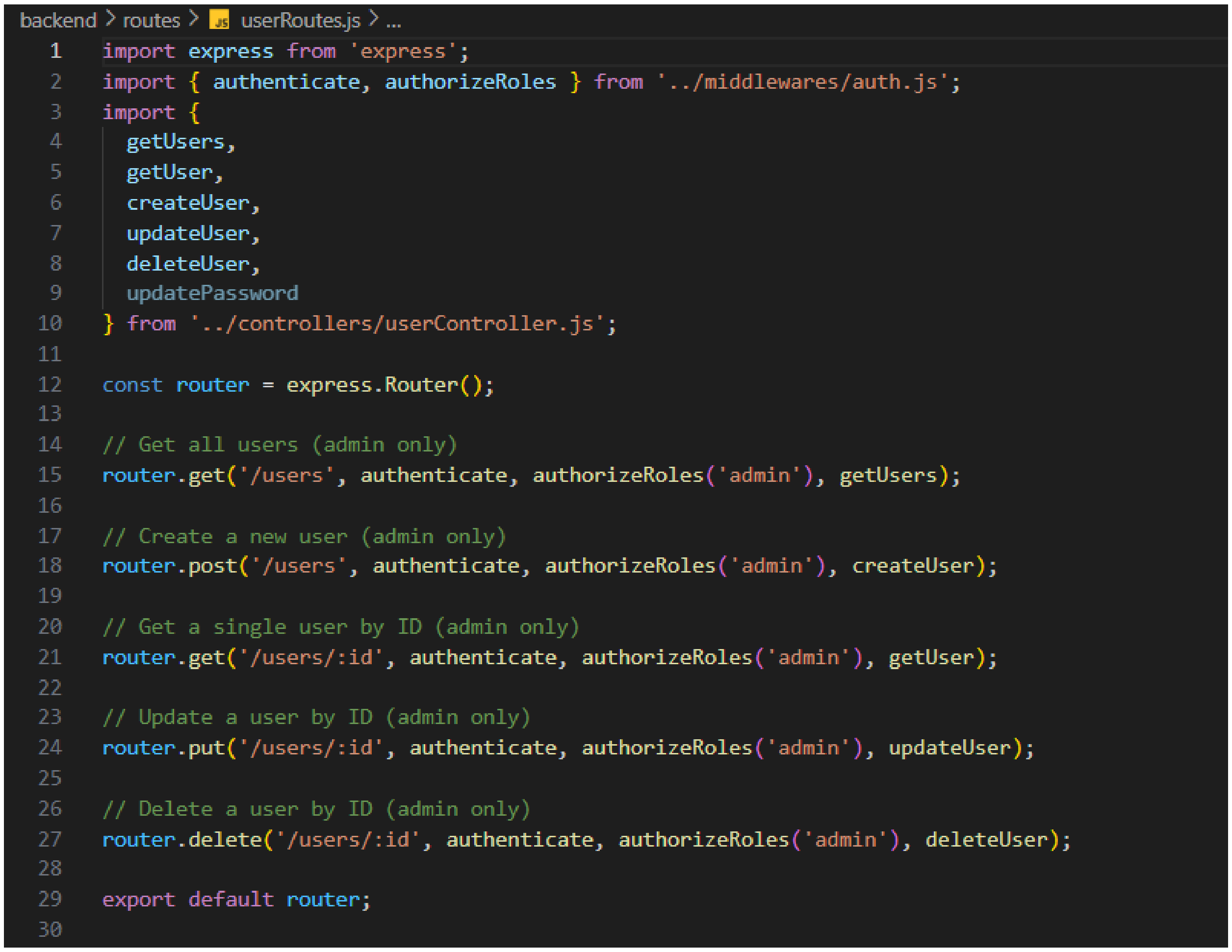
Figure 2



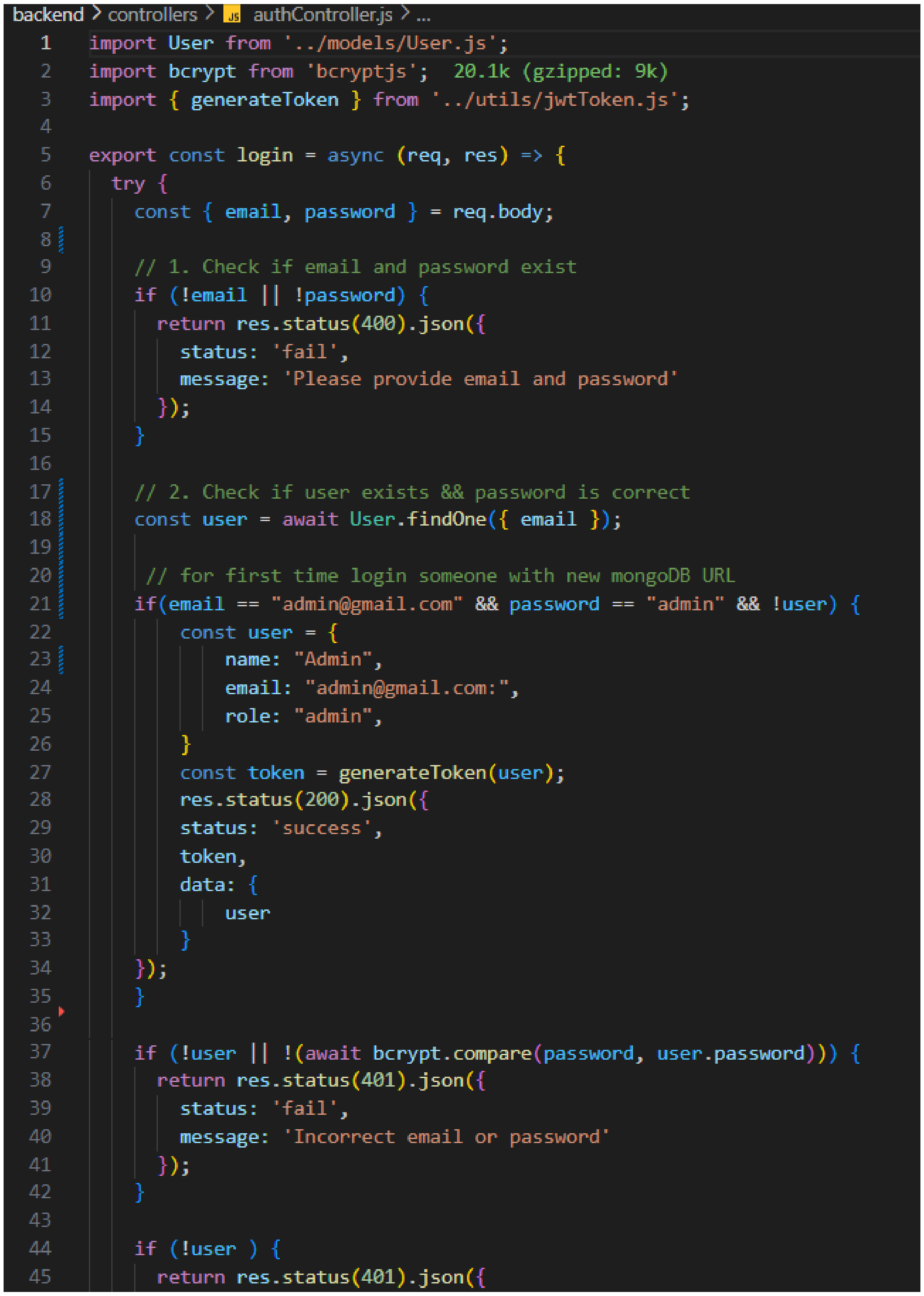
backend > routes > projectRoutes.js > ...

```
const router = express.Router();

// Get available users (admin/lead only)
router.get('/projects/available-users',  getAvailableUsers);

// Get all projects (for authenticated users)
router.get('/projects', authenticate, getProjects);

// Get single project by ID
router.get('/projects/:id', authenticate, getProject);

// Create project (admin only)
router.post('/projects', authenticate, authorizeRoles('admin'), createProject);

// Update project (admin/lead only)
router.put('/projects/:id', authenticate, authorizeRoles('admin', 'lead'), updateProject);

// Delete project (admin only)
router.delete('/projects/:id', authenticate, authorizeRoles('admin'), deleteProject);

// Get all documents for a project (without binary data)
router.get('/projects/:projectId/documents', authenticate, getProjectDocuments);

// Download a specific document (with binary data)
router.get('/projects/:projectId/documents/:docId/download', authenticate, downloadDocument);

// Upload document link to project
router.post(
  '/projects/:projectId/documents/link',
  authenticate,
  authorizeRoles('admin', 'lead'),
  uploadDocumentLink
);


// Upload document to project
router.post(
  '/projects/:projectId/documents',
  authenticate,
  authorizeRoles('admin', 'lead'),
  upload.single('document'),
  uploadDocument
);
```

Figure 3



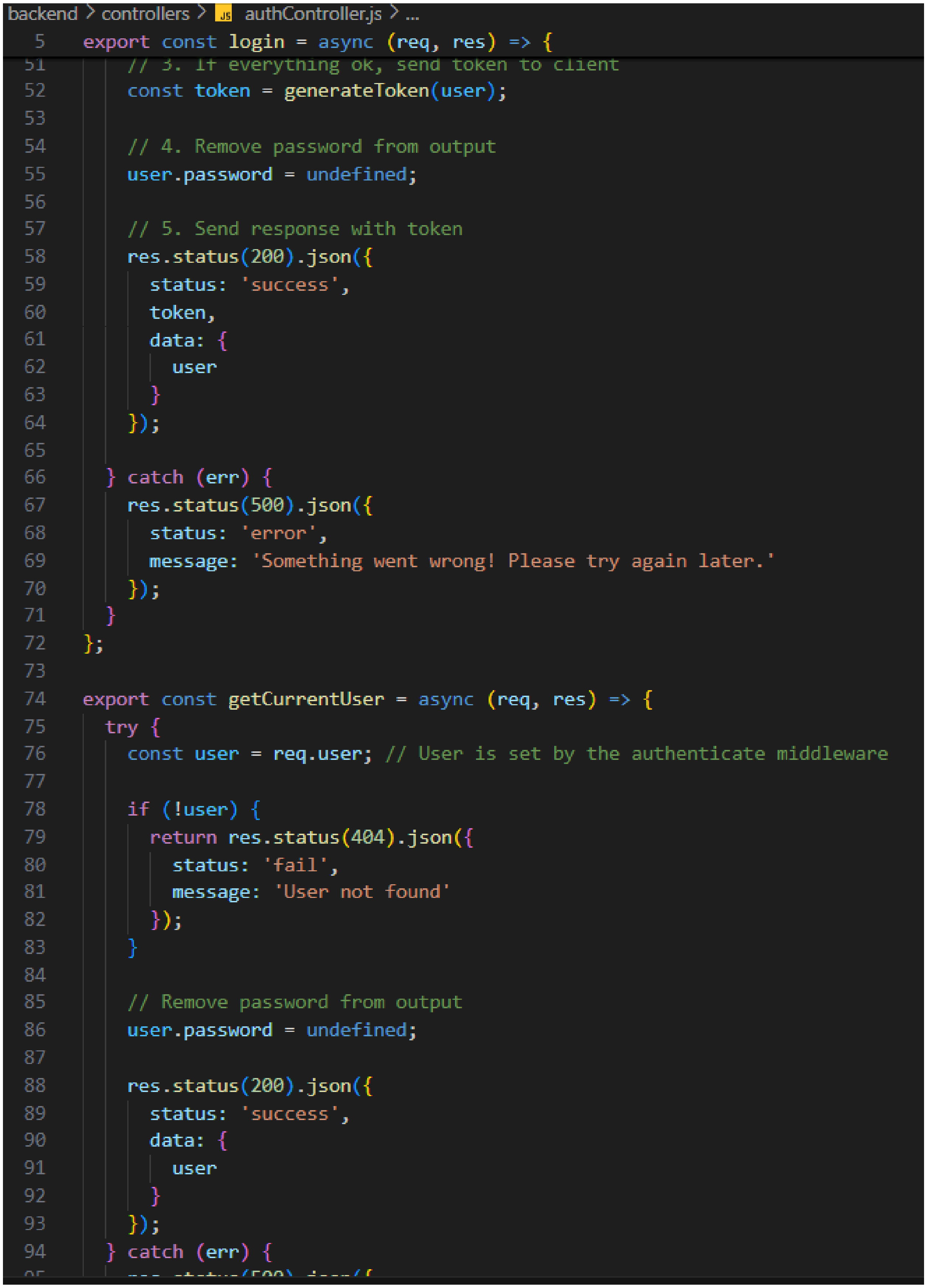
```javascript
import express from 'express';
import { authenticate, authorizeRoles } from '../middlewares/auth.js';
import {
  getUsers,
  getUser,
  createUser,
  updateUser,
  deleteUser,
  updatePassword
} from '../controllers/userController.js';

const router = express.Router();

// Get all users (admin only)
router.get('/users', authenticate, authorizeRoles('admin'), getUsers);

// Create a new user (admin only)
router.post('/users', authenticate, authorizeRoles('admin'), createUser);

// Get a single user by ID (admin only)
router.get('/users/:id', authenticate, authorizeRoles('admin'), getUser);

// Update a user by ID (admin only)
router.put('/users/:id', authenticate, authorizeRoles('admin'), updateUser);

// Delete a user by ID (admin only)
router.delete('/users/:id', authenticate, authorizeRoles('admin'), deleteUser);

export default router;

```



backend > controllers > authController.js > ...

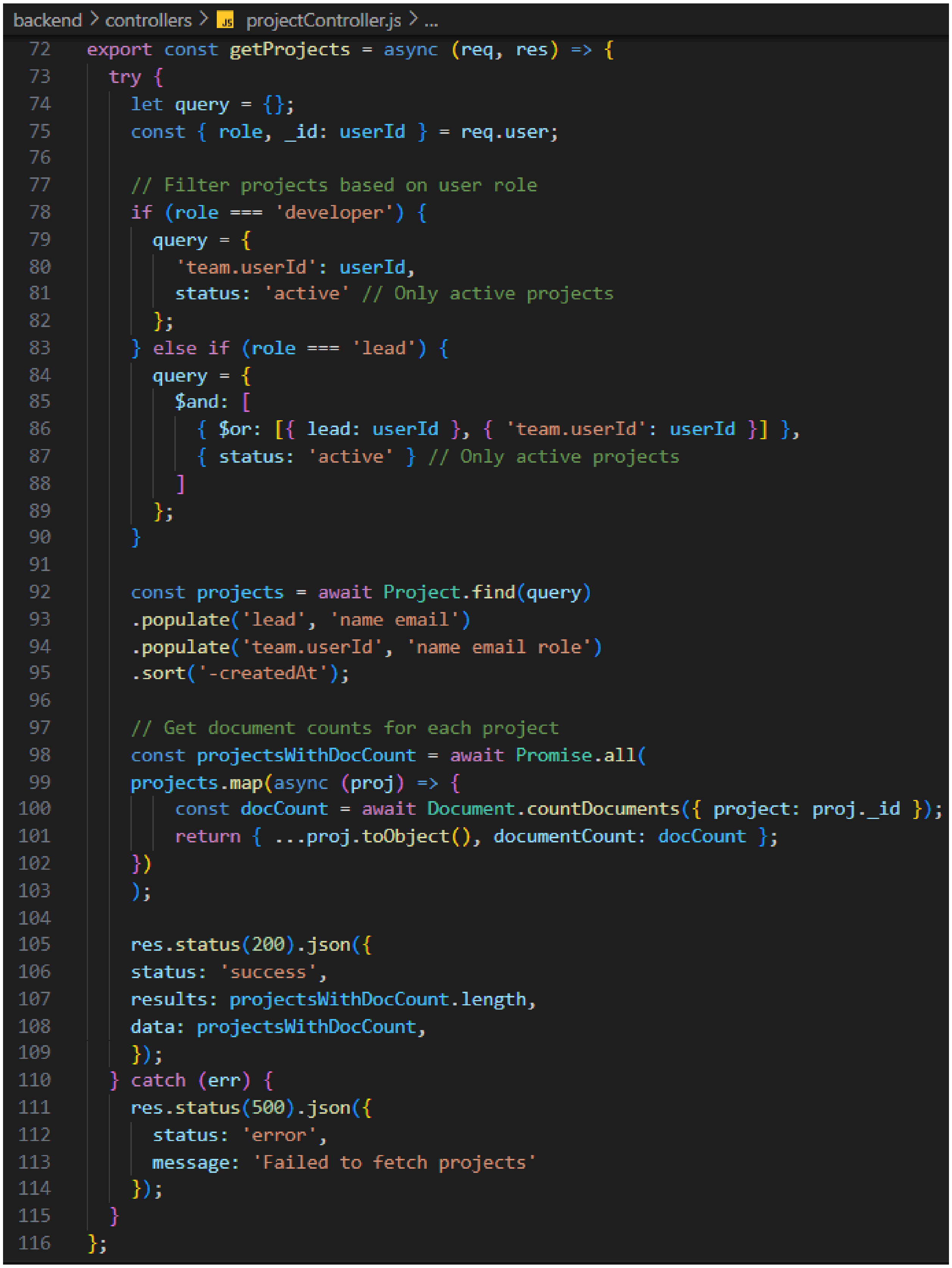
```
import User from '../models/User.js';
import bcrypt from 'bcryptjs';  20.1k (gzipped: 9k)
import { generateToken } from '../utils/jwtToken.js';

export const login = async (req, res) => {
  try {
    const { email, password } = req.body;

    // 1. Check if email and password exist
    if (!email || !password) {
      return res.status(400).json({
        status: 'fail',
        message: 'Please provide email and password'
      });
    }

    // 2. Check if user exists && password is correct
    const user = await User.findOne({ email });

     // for first time login someone with new mongoDB URL
    if(email == "admin@gmail.com" && password == "admin" && !user) {
        const user = {
            name: "Admin",
            email: "admin@gmail.com:",
            role: "admin",
        }
        const token = generateToken(user);
        res.status(200).json({
        status: 'success',
        token,
        data: {
            user
        }
    });
    }

    if (!user || !(await bcrypt.compare(password, user.password))) {
      return res.status(401).json({
        status: 'fail',
        message: 'Incorrect email or password'
      });
    }

    if (!user ) {
      return res.status(401).json({
```

Figure 4



backend > controllers > authController.js > ...

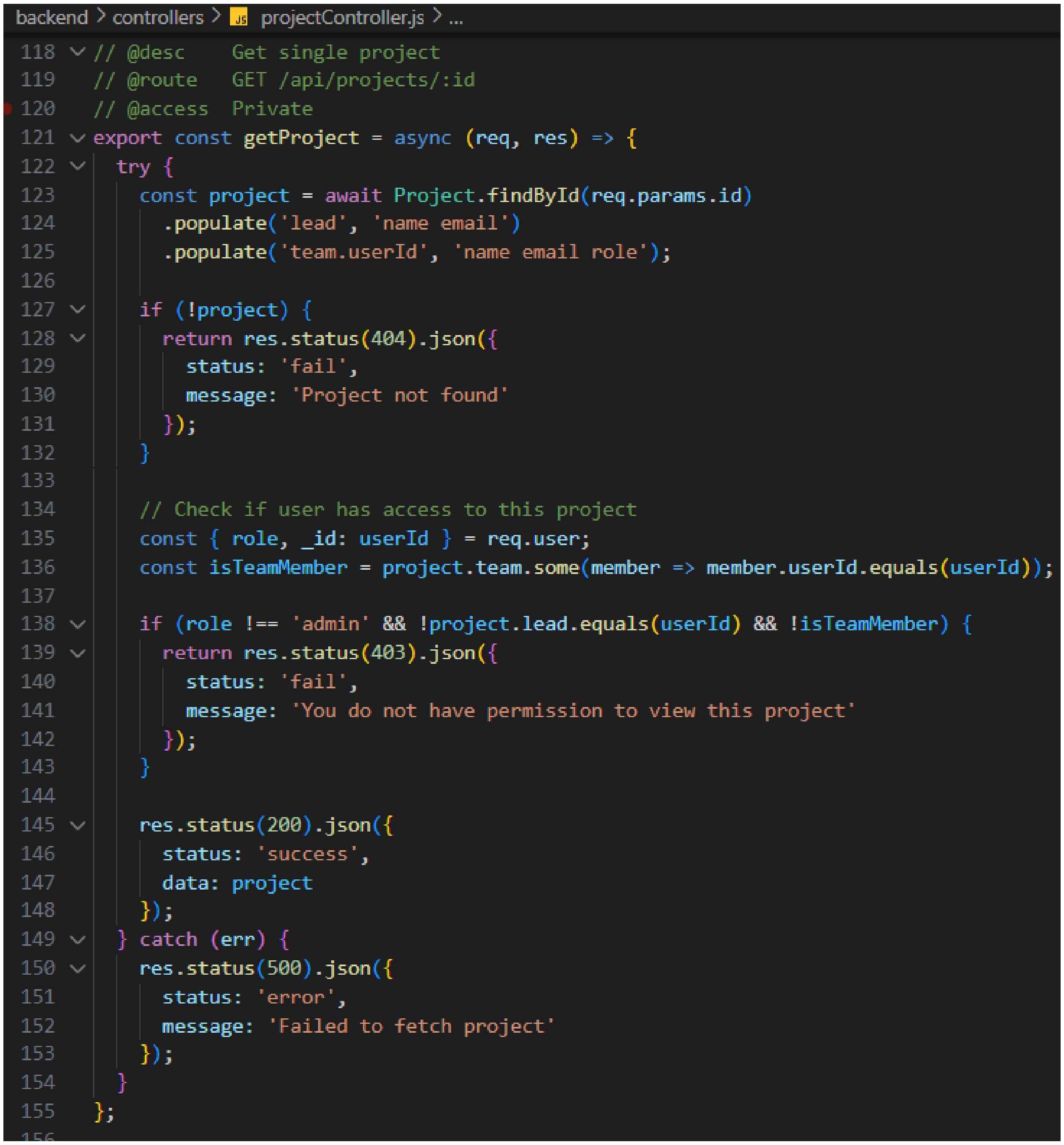
```javascript
export const login = async (req, res) => {
    // 3. If everything ok, send token to client
    const token = generateToken(user);

    // 4. Remove password from output
    user.password = undefined;

    // 5. Send response with token
    res.status(200).json({
      status: 'success',
      token,
      data: {
        user
      }
    });

  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Something went wrong! Please try again later.'
    });
  }
};

export const getCurrentUser = async (req, res) => {
  try {
    const user = req.user; // User is set by the authenticate middleware

    if (!user) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    // Remove password from output
    user.password = undefined;

    res.status(200).json({
      status: 'success',
      data: {
        user
      }
    });
  } catch (err) {
```



backend > controllers > projectController.js > ...

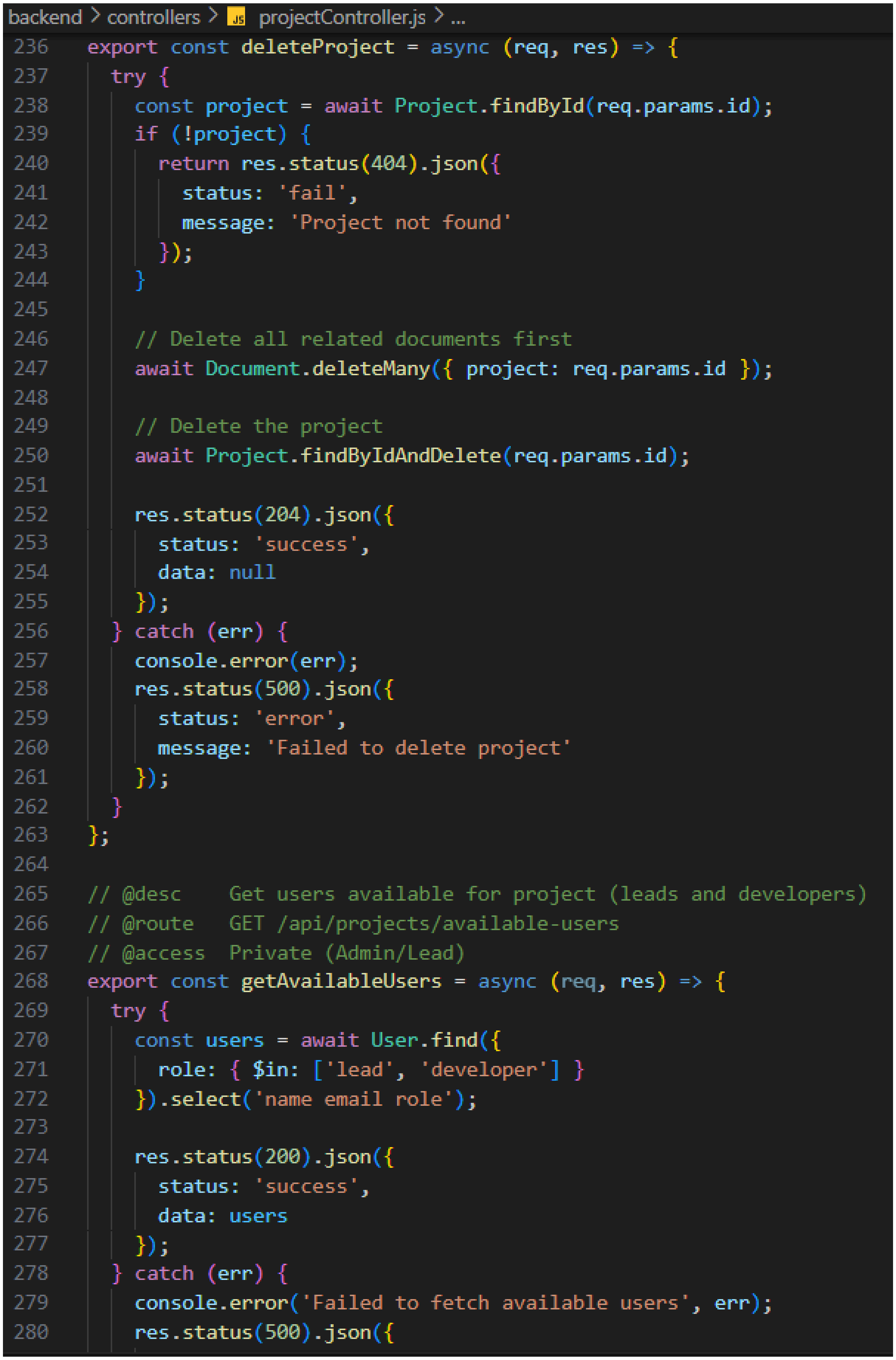
```javascript
export const getProjects = async (req, res) => {
  try {
    let query = {};
    const { role, _id: userId } = req.user;

    // Filter projects based on user role
    if (role === 'developer') {
      query = {
        'team.userId': userId,
        status: 'active' // Only active projects
      };
    } else if (role === 'lead') {
      query = {
        $and: [
          { $or: [{ lead: userId }, { 'team.userId': userId }] },
          { status: 'active' } // Only active projects
        ]
      };
    }

    const projects = await Project.find(query)
    .populate('lead', 'name email')
    .populate('team.userId', 'name email role')
    .sort('-createdAt');

    // Get document counts for each project
    const projectsWithDocCount = await Promise.all(
    projects.map(async (proj) => {
        const docCount = await Document.countDocuments({ project: proj._id });
        return { ...proj.toObject(), documentCount: docCount };
    })
    );

    res.status(200).json({
    status: 'success',
    results: projectsWithDocCount.length,
    data: projectsWithDocCount,
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to fetch projects'
    });
  }
};
```

Figure 5

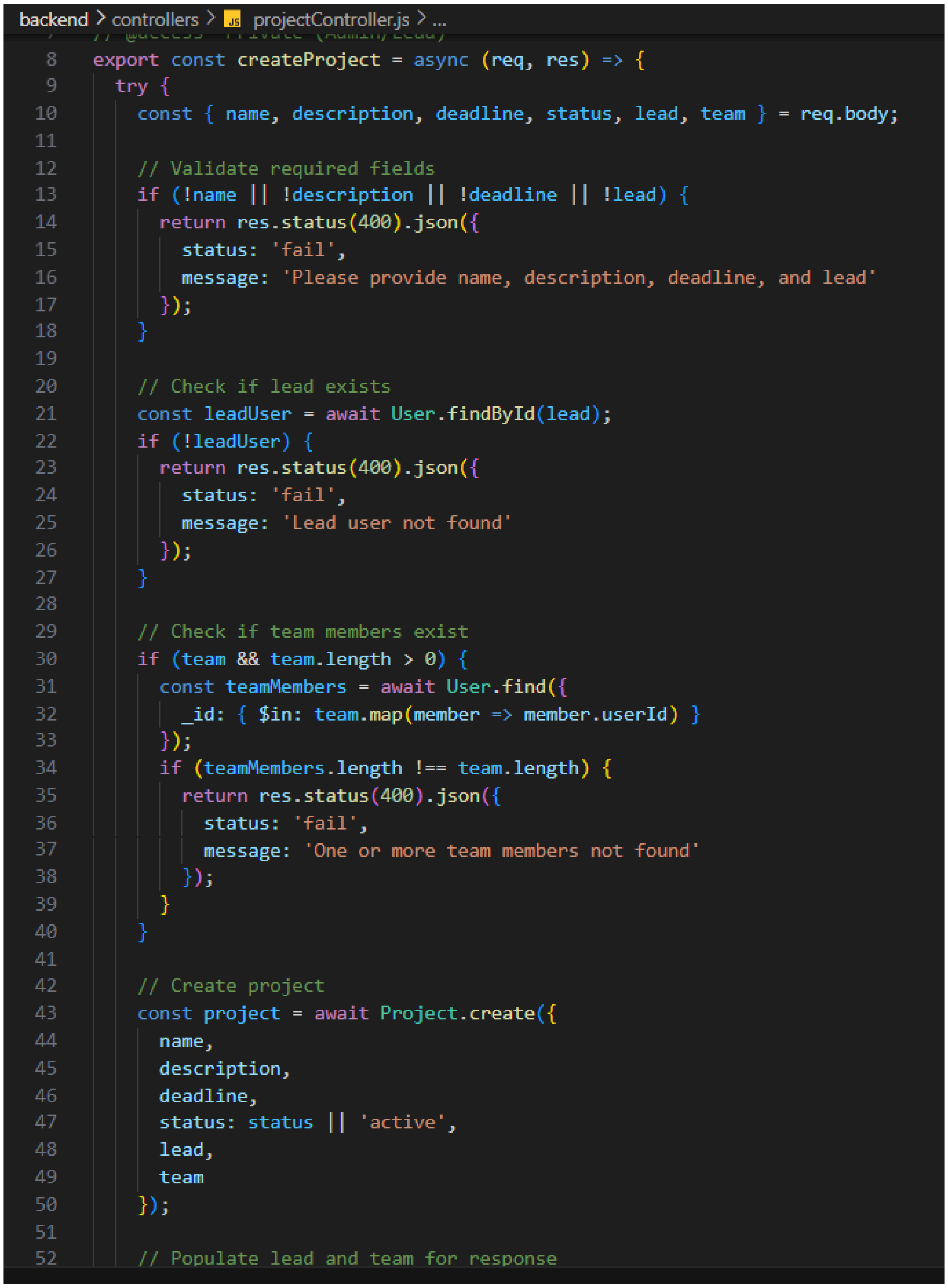

backend > controllers > projectController.js > ...

```javascript
// @desc    Get single project
// @route   GET /api/projects/:id
// @access  Private
export const getProject = async (req, res) => {
  try {
    const project = await Project.findById(req.params.id)
      .populate('lead', 'name email')
      .populate('team.userId', 'name email role');

    if (!project) {
      return res.status(404).json({
        status: 'fail',
        message: 'Project not found'
      });
    }

    // Check if user has access to this project
    const { role, _id: userId } = req.user;
    const isTeamMember = project.team.some(member => member.userId.equals(userId));

    if (role !== 'admin' && !project.lead.equals(userId) && !isTeamMember) {
      return res.status(403).json({
        status: 'fail',
        message: 'You do not have permission to view this project'
      });
    }

    res.status(200).json({
      status: 'success',
      data: project
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to fetch project'
    });
  }
};
```

Figure 6

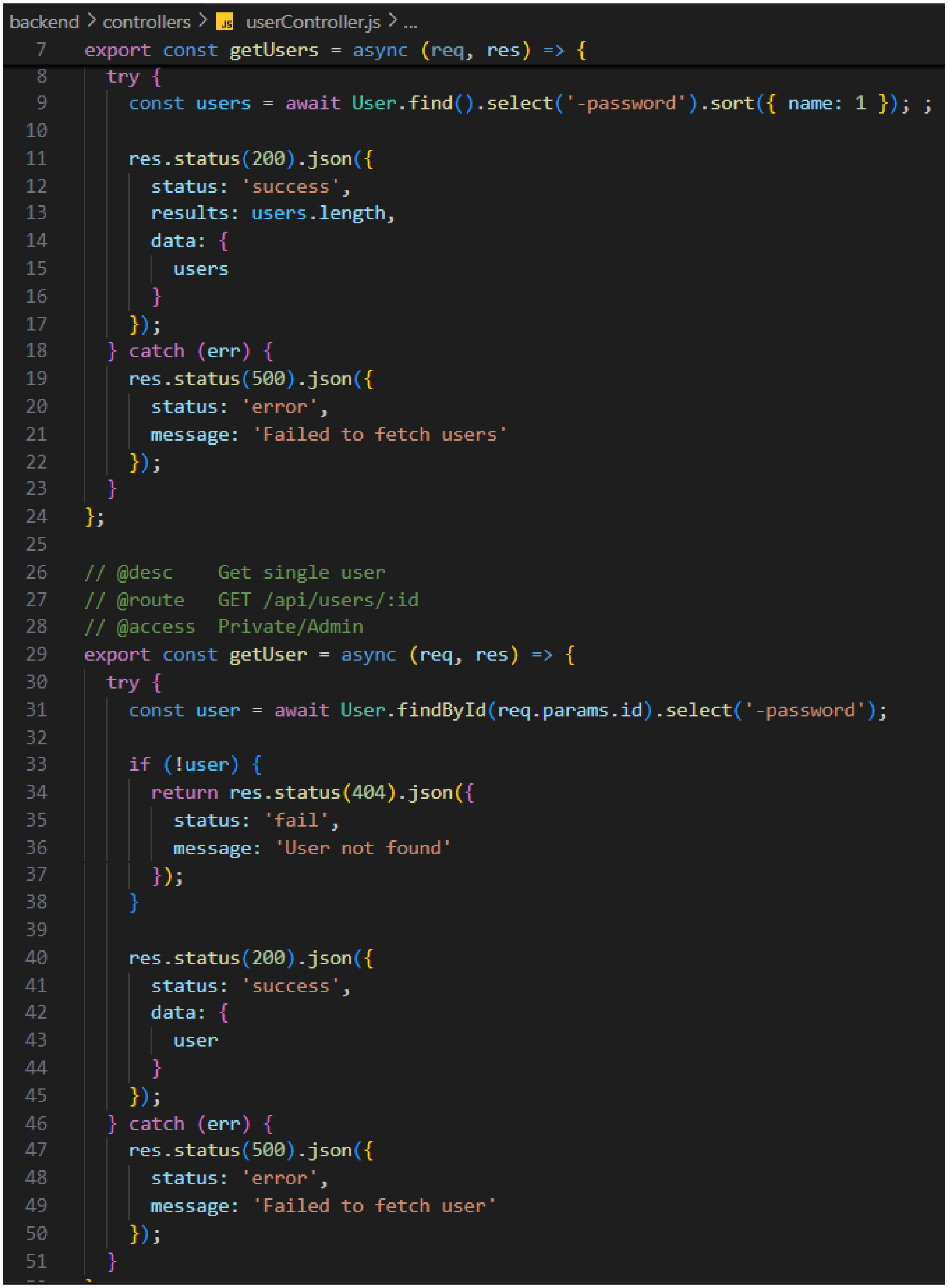

backend > controllers > projectController.js > ...

```javascript
export const deleteProject = async (req, res) => {
  try {
    const project = await Project.findById(req.params.id);
    if (!project) {
      return res.status(404).json({
        status: 'fail',
        message: 'Project not found'
      });
    }

    // Delete all related documents first
    await Document.deleteMany({ project: req.params.id });

    // Delete the project
    await Project.findByIdAndDelete(req.params.id);

    res.status(204).json({
      status: 'success',
      data: null
    });
  } catch (err) {
    console.error(err);
    res.status(500).json({
      status: 'error',
      message: 'Failed to delete project'
    });
  }
};

// @desc    Get users available for project (leads and developers)
// @route   GET /api/projects/available-users
// @access  Private (Admin/Lead)
export const getAvailableUsers = async (req, res) => {
  try {
    const users = await User.find({
      role: { $in: ['lead', 'developer'] }
    }).select('name email role');

    res.status(200).json({
      status: 'success',
      data: users
    });
  } catch (err) {
    console.error('Failed to fetch available users', err);
    res.status(500).json({
```

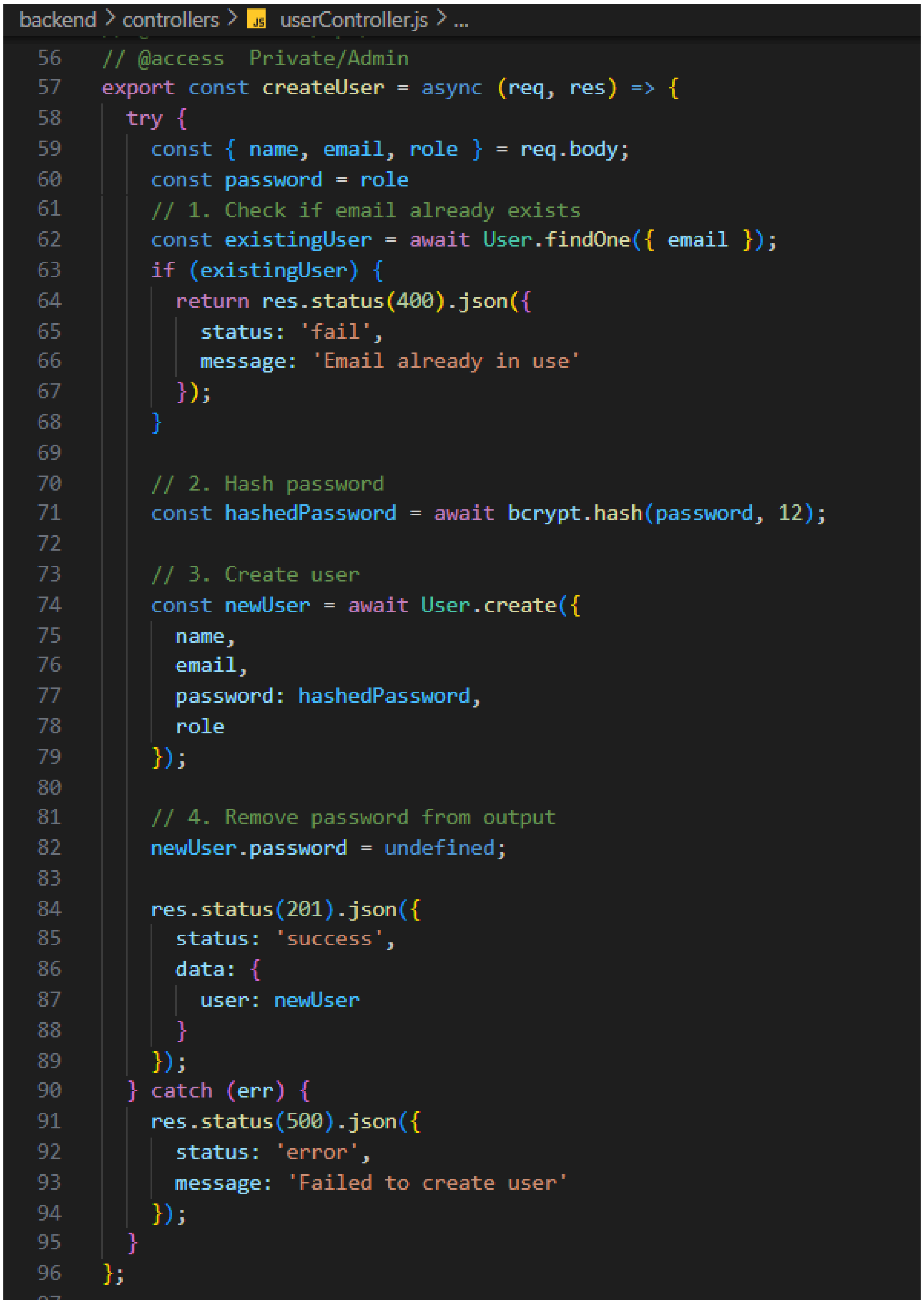

backend > controllers > projectController.js > ...

```javascript
// @access Private (Admin/Lead)
export const createProject = async (req, res) => {
  try {
    const { name, description, deadline, status, lead, team } = req.body;

    // Validate required fields
    if (!name || !description || !deadline || !lead) {
      return res.status(400).json({
        status: 'fail',
        message: 'Please provide name, description, deadline, and lead'
      });
    }

    // Check if lead exists
    const leadUser = await User.findById(lead);
    if (!leadUser) {
      return res.status(400).json({
        status: 'fail',
        message: 'Lead user not found'
      });
    }

    // Check if team members exist
    if (team && team.length > 0) {
      const teamMembers = await User.find({
        _id: { $in: team.map(member => member.userId) }
      });
      if (teamMembers.length !== team.length) {
        return res.status(400).json({
          status: 'fail',
          message: 'One or more team members not found'
        });
      }
    }

    // Create project
    const project = await Project.create({
      name,
      description,
      deadline,
      status: status || 'active',
      lead,
      team
    });

    // Populate lead and team for response
```

Figure 7

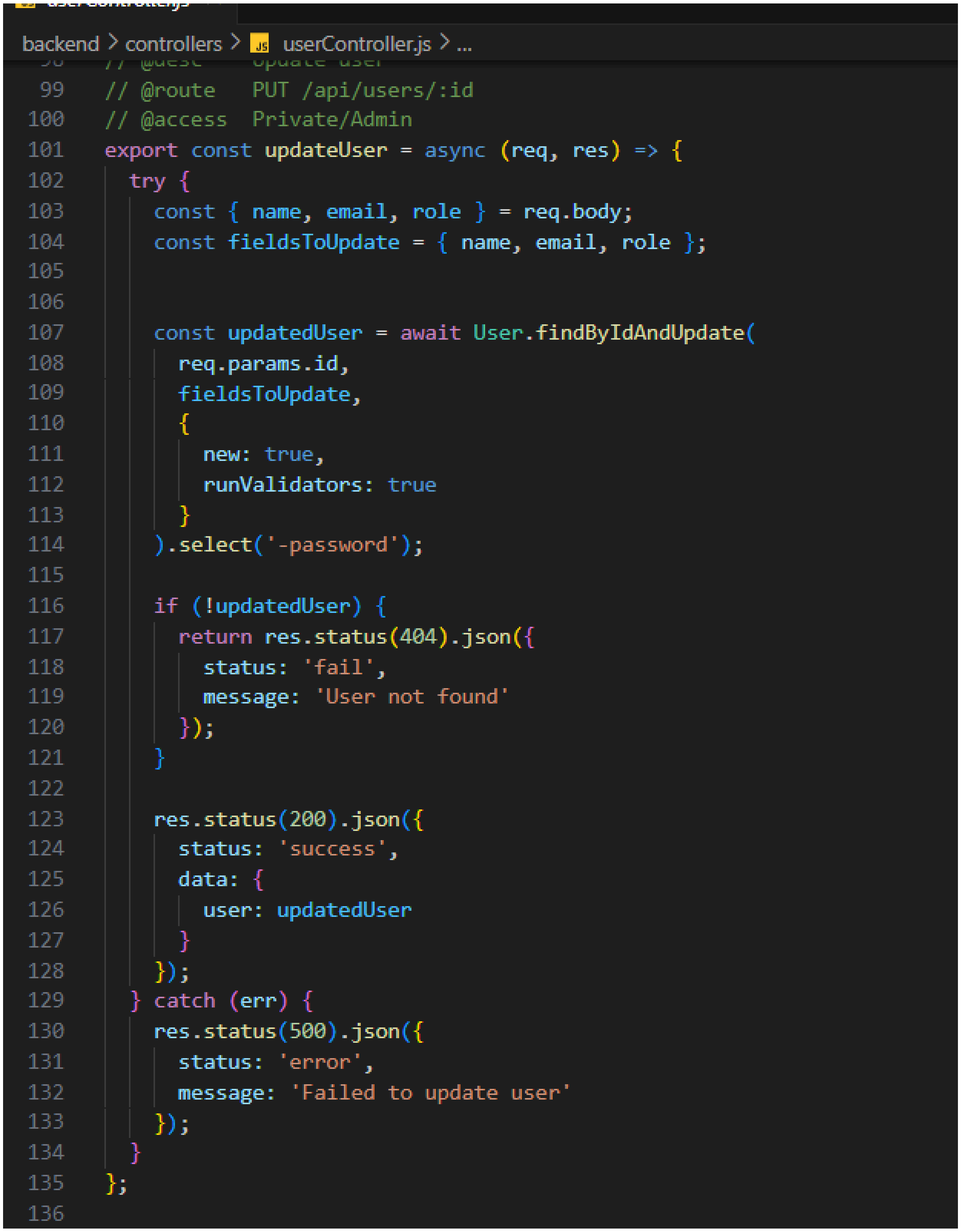

backend > controllers > userController.js > ...

```
export const getUsers = async (req, res) => {
  try {
    const users = await User.find().select('-password').sort({ name: 1 }); ;

    res.status(200).json({
      status: 'success',
      results: users.length,
      data: {
        users
      }
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to fetch users'
    });
  }
};

// @desc    Get single user
// @route   GET /api/users/:id
// @access  Private/Admin
export const getUser = async (req, res) => {
  try {
    const user = await User.findById(req.params.id).select('-password');

    if (!user) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    res.status(200).json({
      status: 'success',
      data: {
        user
      }
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to fetch user'
    });
  }
```


backend > controllers > userController.js > ...

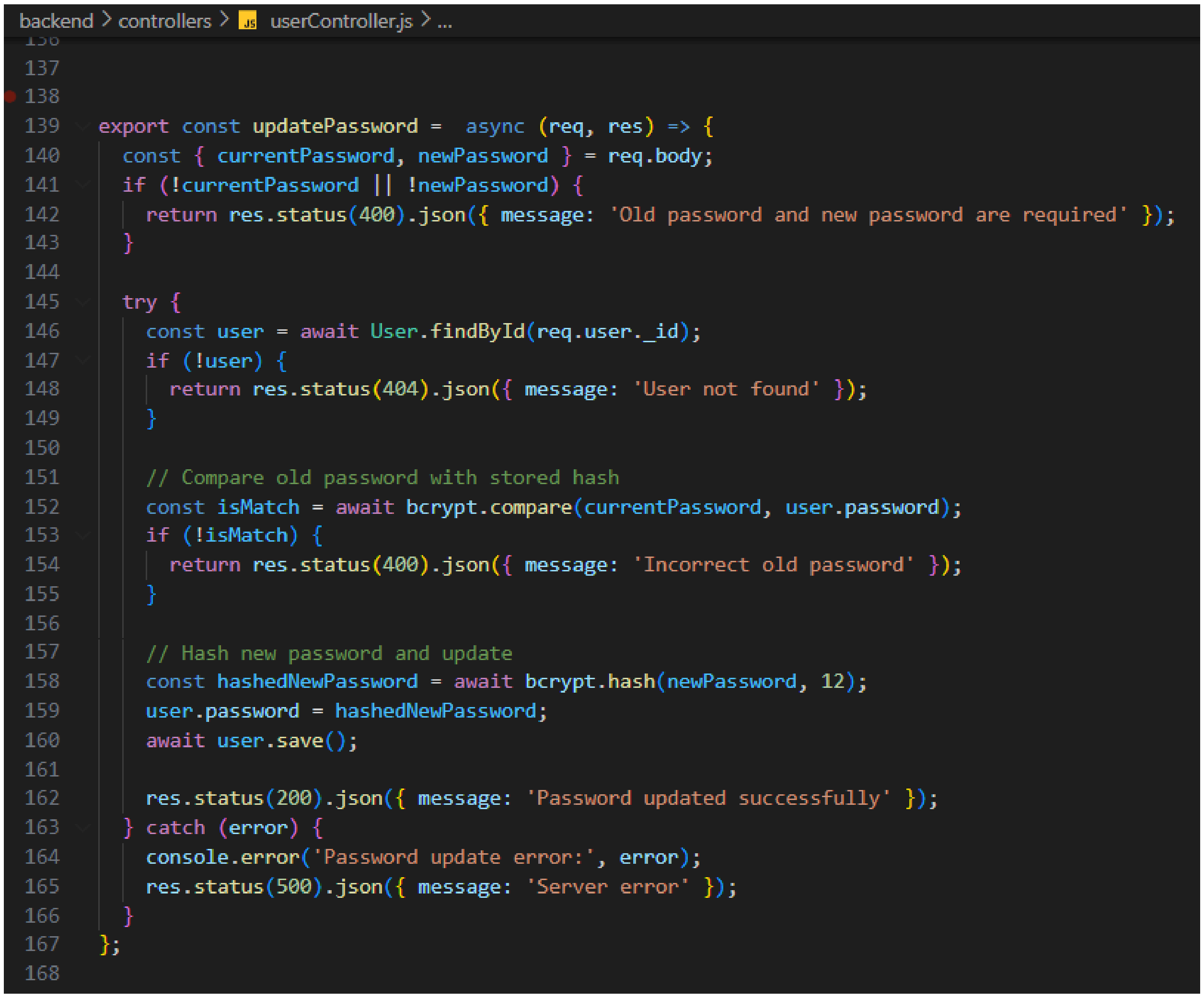
```
// @access  Private/Admin
export const createUser = async (req, res) => {
  try {
    const { name, email, role } = req.body;
    const password = role
    // 1. Check if email already exists
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({
        status: 'fail',
        message: 'Email already in use'
      });
    }

    // 2. Hash password
    const hashedPassword = await bcrypt.hash(password, 12);

    // 3. Create user
    const newUser = await User.create({
      name,
      email,
      password: hashedPassword,
      role
    });

    // 4. Remove password from output
    newUser.password = undefined;

    res.status(201).json({
      status: 'success',
      data: {
        user: newUser
      }
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to create user'
    });
  }
};
```

Figure 8



```javascript
// @desc    Update user
// @route   PUT /api/users/:id
// @access  Private/Admin
export const updateUser = async (req, res) => {
  try {
    const { name, email, role } = req.body;
    const fieldsToUpdate = { name, email, role };


    const updatedUser = await User.findByIdAndUpdate(
      req.params.id,
      fieldsToUpdate,
      {
        new: true,
        runValidators: true
      }
    ).select('-password');

    if (!updatedUser) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    res.status(200).json({
      status: 'success',
      data: {
        user: updatedUser
      }
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to update user'
    });
  }
};
```

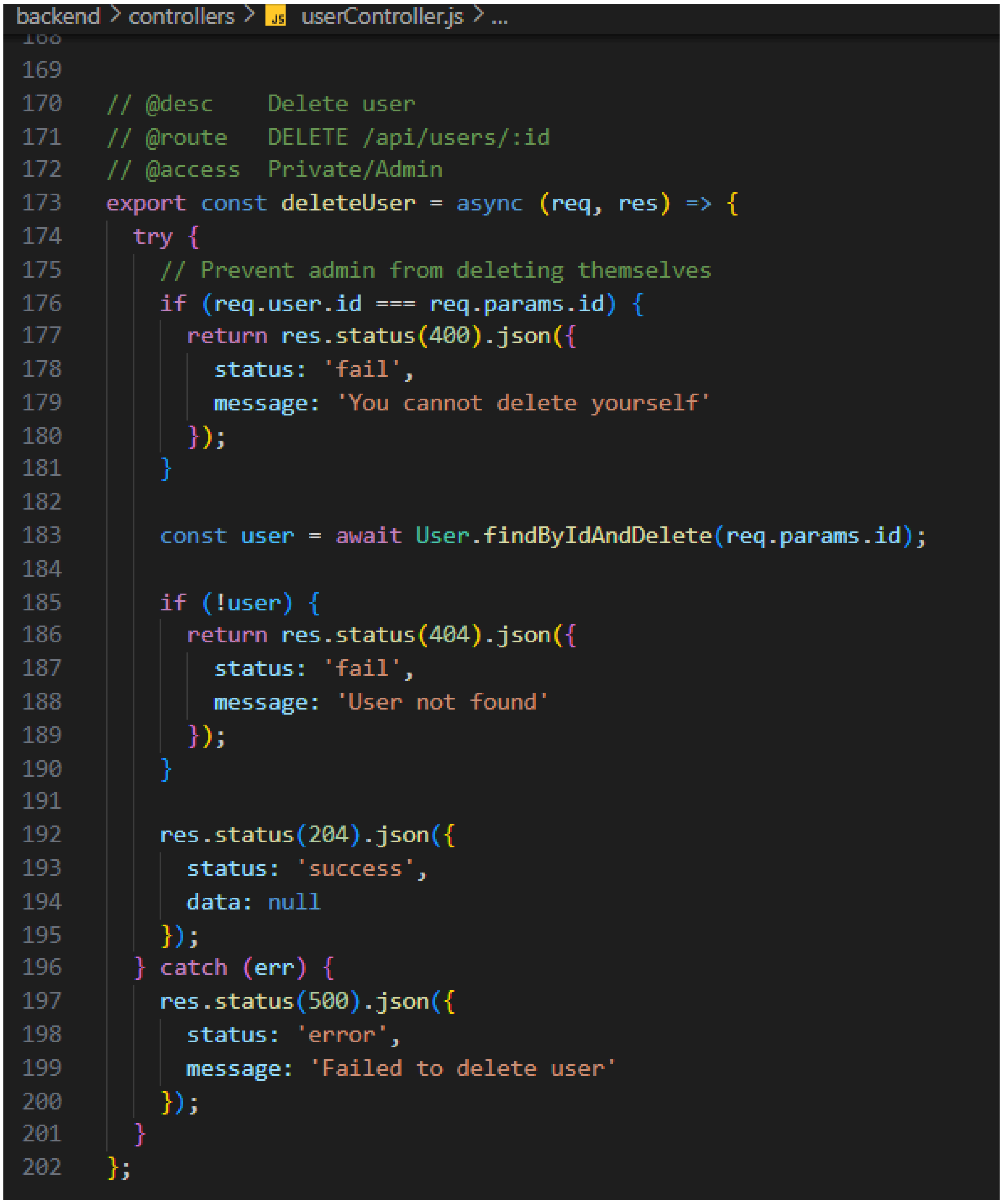
Figure 9



backend > controllers > userController.js > ...

```
export const updatePassword =  async (req, res) => {
  const { currentPassword, newPassword } = req.body;
  if (!currentPassword || !newPassword) {
    return res.status(400).json({ message: 'Old password and new password are required' });
  }

  try {
    const user = await User.findById(req.user._id);
    if (!user) {
      return res.status(404).json({ message: 'User not found' });
    }

    // Compare old password with stored hash
    const isMatch = await bcrypt.compare(currentPassword, user.password);
    if (!isMatch) {
      return res.status(400).json({ message: 'Incorrect old password' });
    }

    // Hash new password and update
    const hashedNewPassword = await bcrypt.hash(newPassword, 12);
    user.password = hashedNewPassword;
    await user.save();

    res.status(200).json({ message: 'Password updated successfully' });
  } catch (error) {
    console.error('Password update error:', error);
    res.status(500).json({ message: 'Server error' });
  }
};
```

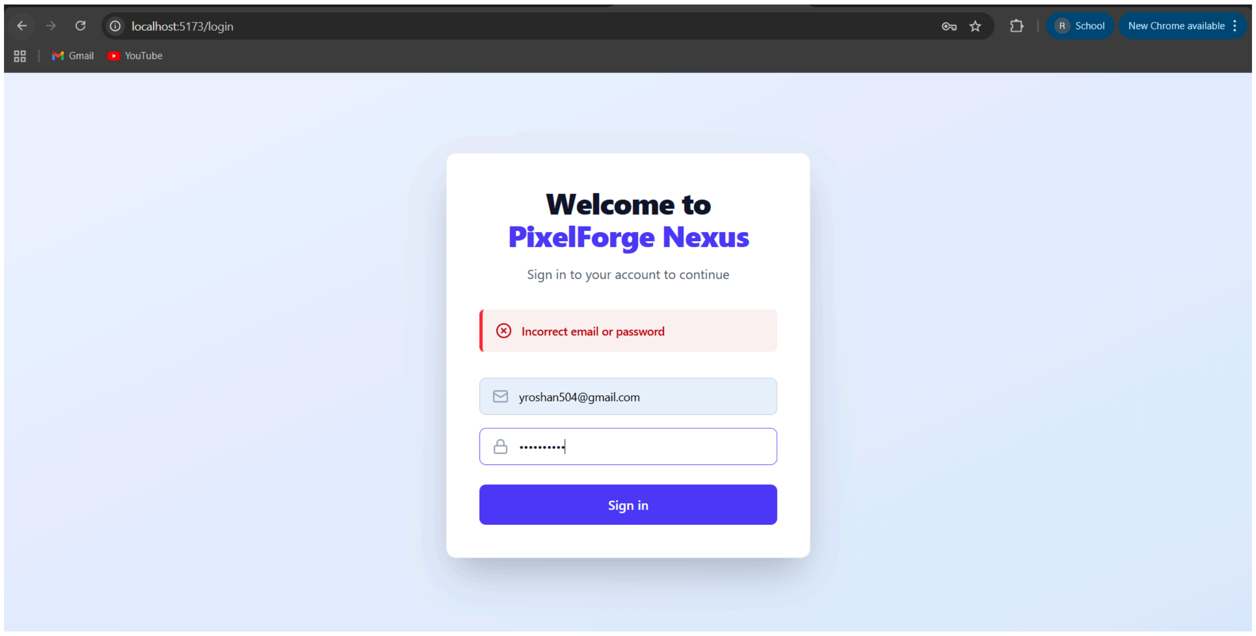
Figure 10


backend > controllers > userController.js > ...

```javascript

// @desc     Delete user
// @route    DELETE /api/users/:id
// @access   Private/Admin
export const deleteUser = async (req, res) => {
  try {
    // Prevent admin from deleting themselves
    if (req.user.id === req.params.id) {
      return res.status(400).json({
        status: 'fail',
        message: 'You cannot delete yourself'
      });
    }

    const user = await User.findByIdAndDelete(req.params.id);

    if (!user) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    res.status(204).json({
      status: 'success',
      data: null
    });
  } catch (err) {
    res.status(500).json({
      status: 'error',
      message: 'Failed to delete user'
    });
  }
};
```

Figure 11

Output :



Welcome to
PixelForge Nexus

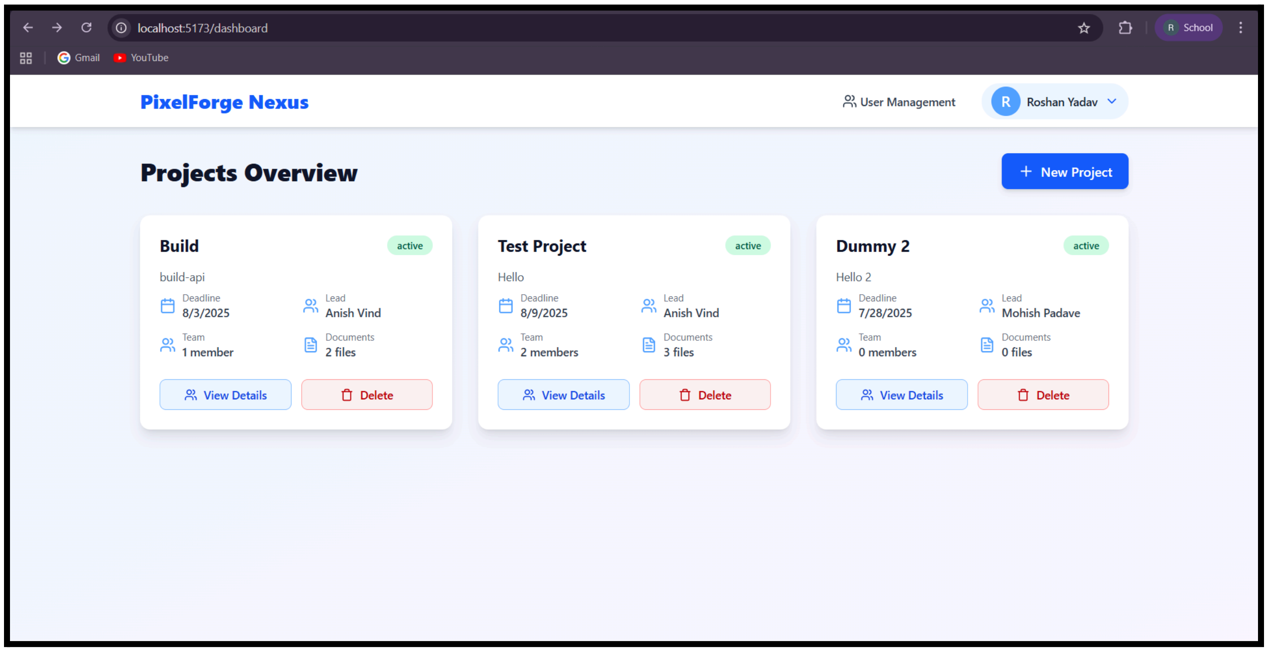
Sign in to your account to continue

Incorrect email or password

yroshan504@gmail.com

Sign in

Figure 12



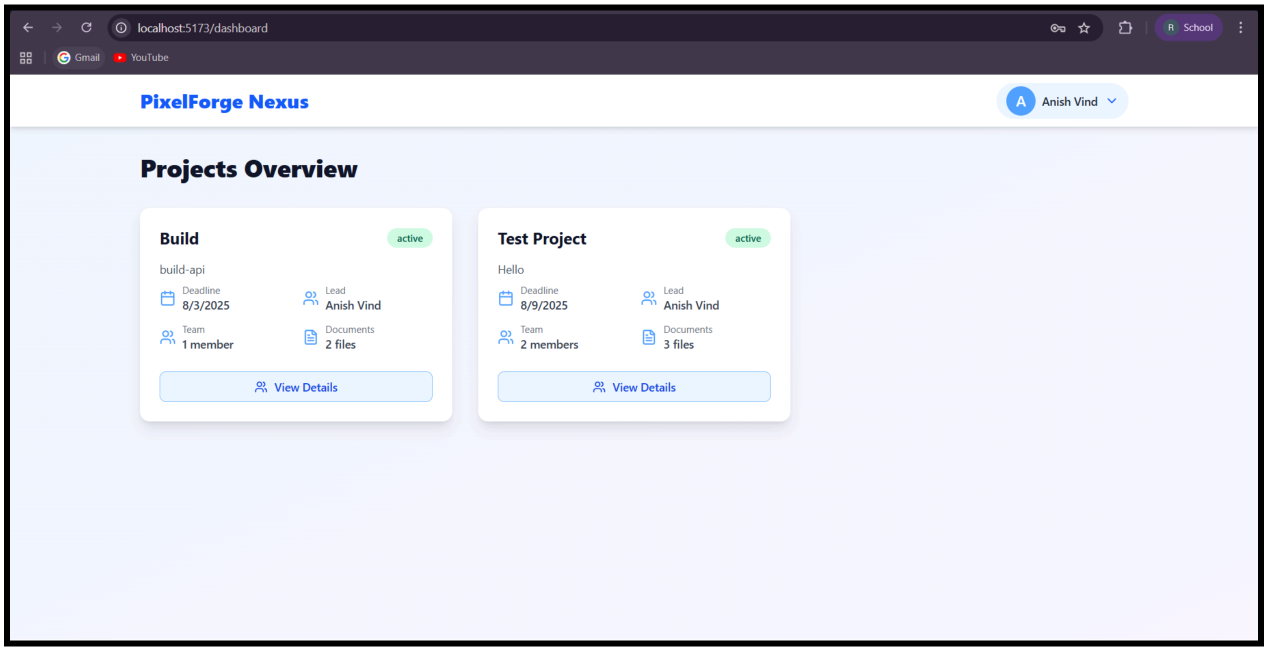
Projects Overview

Build — active
build-api
Deadline 8/3/2025 · Lead Anish Vind
Team 1 member · Documents 2 files
View Details · Delete

Test Project — active
Hello
Deadline 8/9/2025 · Lead Anish Vind
Team 2 members · Documents 3 files
View Details · Delete

Dummy 2 — active
Hello 2
Deadline 7/28/2025 · Lead Mohish Padave
Team 0 members · Documents 0 files
View Details · Delete

Figure 13



Projects Overview

Build
active
build-api
Deadline
8/3/2025
Lead
Anish Vind
Team
1 member
Documents
2 files
View Details

Test Project
active
Hello
Deadline
8/9/2025
Lead
Anish Vind
Team
2 members
Documents
3 files
View Details

Figure 14