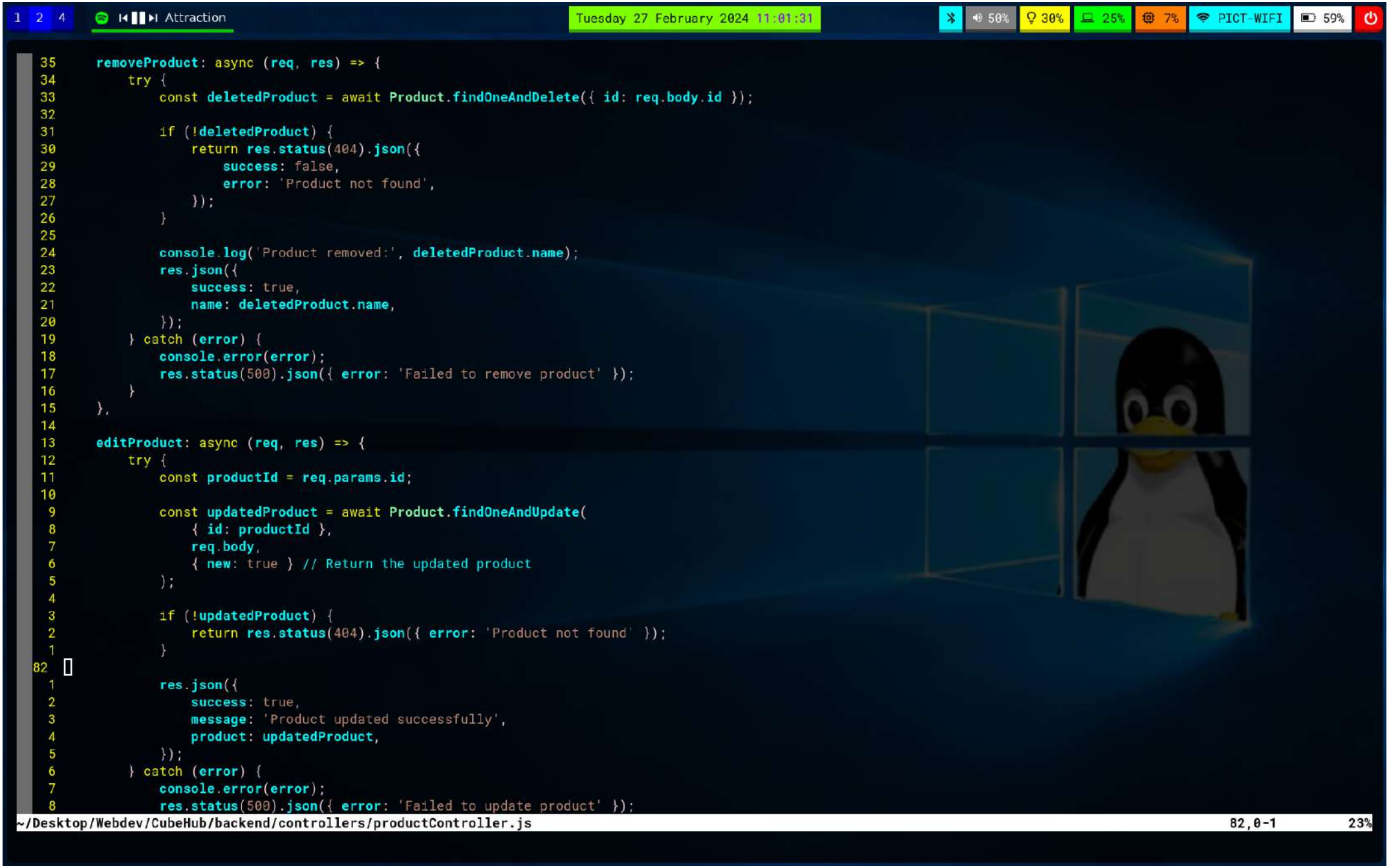
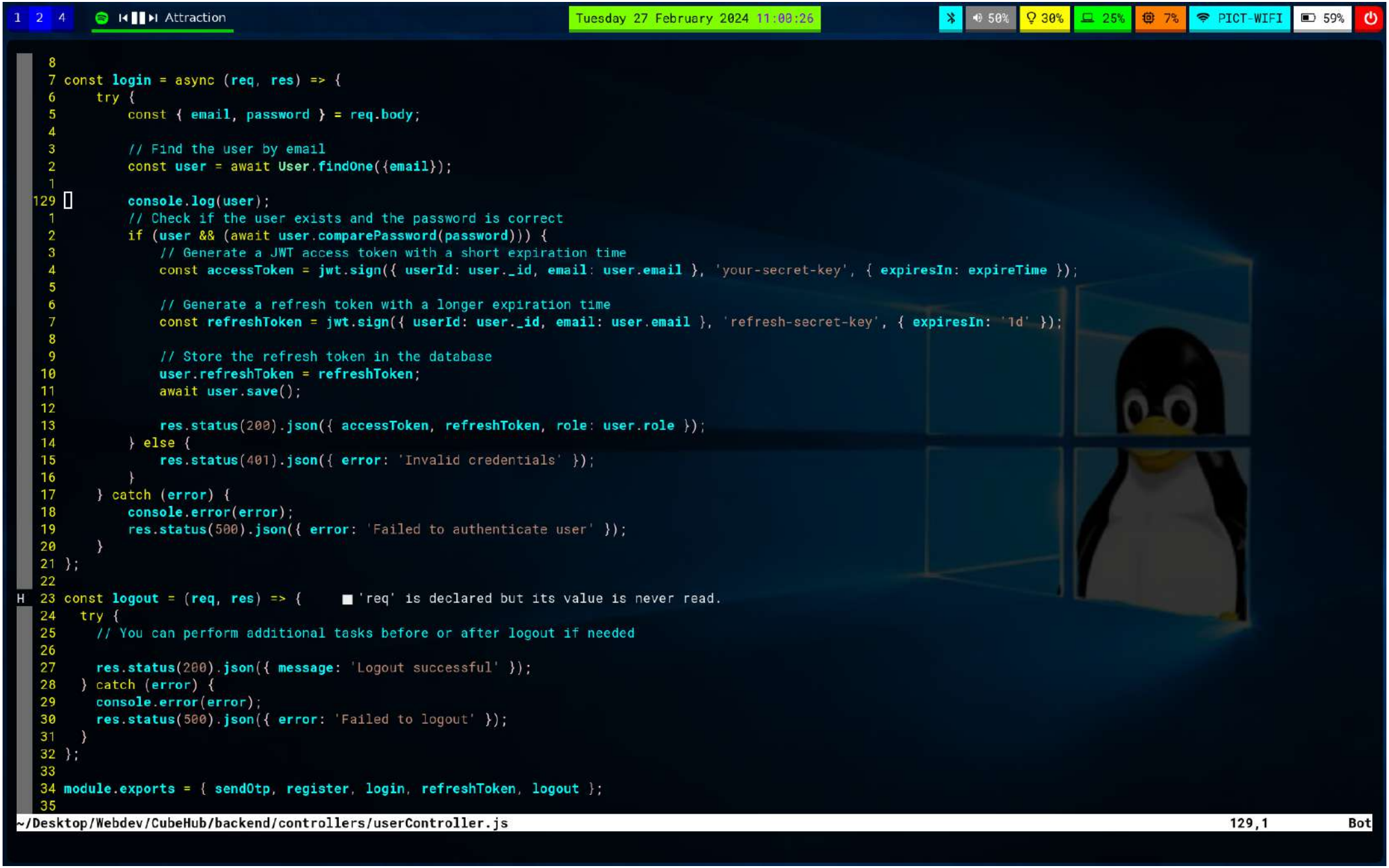
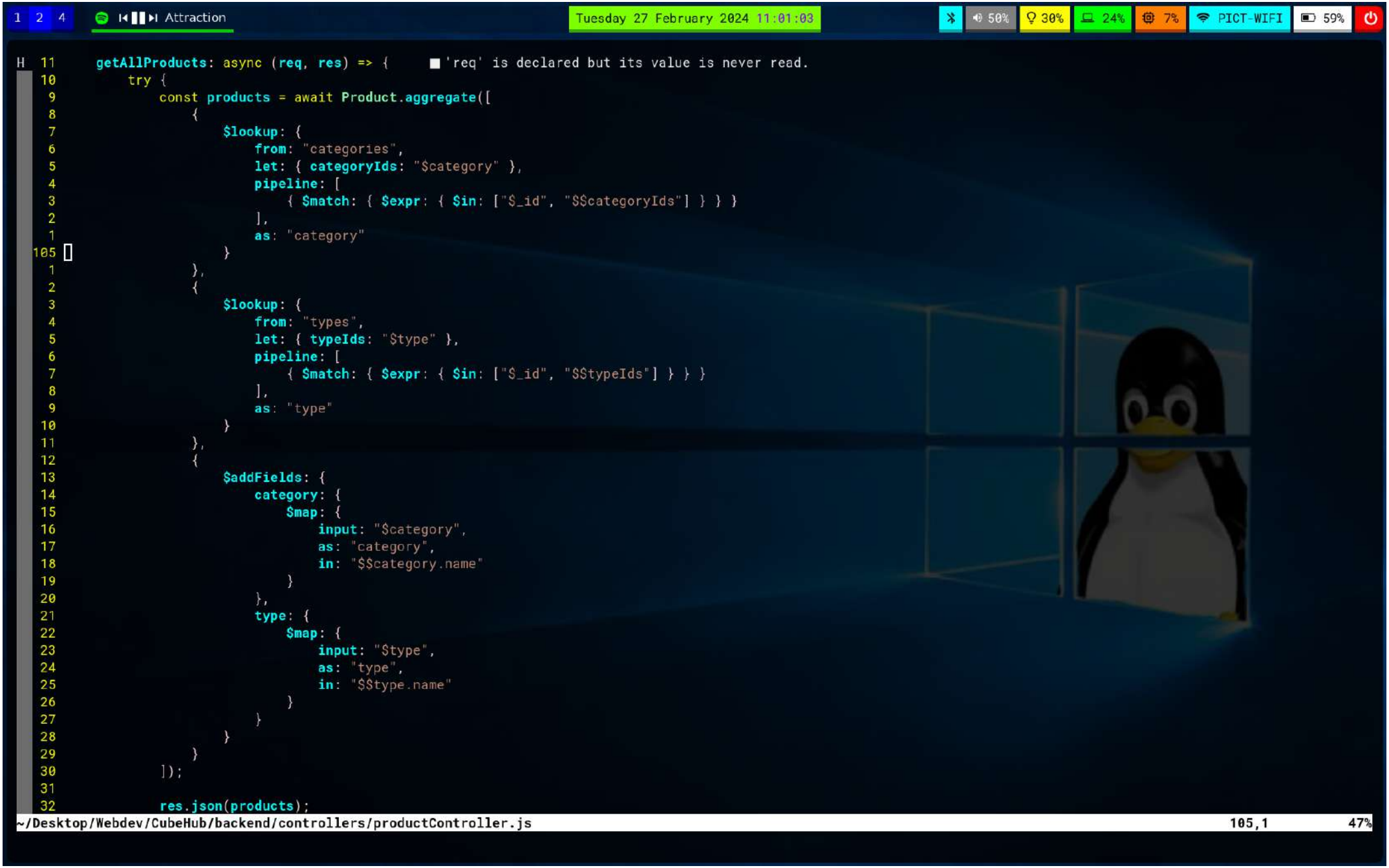

1 2 4
Attraction
Tuesday 27 February 2024 11:01:31
50%
30%
25%
7%
PICT-WIFI
59%
removeProduct: async (req, res) => {
try {
const deletedProduct = await Product.findOneAndDelete({ id: req.body.id });
if (!deletedProduct) {
return res.status(404).json({
success: false,
error: 'Product not found',
});
}
console.log('Product removed:', deletedProduct.name);
res.json({
success: true,
name: deletedProduct.name,
});
} catch (error) {
console.error(error);
res.status(500).json({ error: 'Failed to remove product' });
}
},
editProduct: async (req, res) => {
try {
const productId = req.params.id;
const updatedProduct = await Product.findOneAndUpdate(
{ id: productId },
req.body,
{ new: true } // Return the updated product
);
if (!updatedProduct) {
return res.status(404).json({ error: 'Product not found' });
}
res.json({
success: true,
message: 'Product updated successfully',
product: updatedProduct,
});
} catch (error) {
console.error(error);
res.status(500).json({ error: 'Failed to update product' });
~/Desktop/Webdev/CubeHub/backend/controllers/productController.js
82,0-1
23%


```
};

const register = async (req, res) => {
    try {
        const { username, email, password, otp } = req.body;

        const existingUser = await User.findOne({email});
        if(existingUser){
            return res.status(400).json({
                success: false,
                message: 'user already exists',
            });
        }

        const recentOtp = await OTP.find({email}).sort({createdAt: -1}).limit(1);

        if(recentOtp.length === 0){
            return res.status(400).json({
                success:false,
                message: 'otp not found'
            });
        }
        else if(otp !== recentOtp[0].otp){
            return res.status(400).json({
                success: false,
                message: 'invalid otp',
            });
        }

        // Create a new user in the database
        const newUser = new User({ username, email, password, role: 'customer'});
        await newUser.save();

        res.status(201).json({ message: 'User registered successfully' });
    } catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Failed to register user' });
    }
};

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

```

```
11 module.exports = {
10     addProduct: async (req, res) => {
 9         try {
 8             const categories = req.body.categories.map(categoryId => new mongoose.Types.ObjectId(categoryId));
 7             const types = req.body.types.map(typeId => new mongoose.Types.ObjectId(typeId));
 6
 5
 4             let products = await Product.find({});
 3             let id;
 2             if(products.length>0){
 1                 let last_product_array = products.slice(-1);
15                 let last_product = last_product_array[0];
 1                 id = last_product.id+1;
 2             }
 3             else{
 4                 id = 1;
 5             }
 6
 7             const product = new Product({
 8                 id:id,
 9                 name:req.body.name,
10                 type:types,
11                 brand:req.body.brand,
12                 description:req.body.description,
13                 images:req.body.images,
14                 category:categories,
15                 new_price:req.body.new_price,
16                 old_price:req.body.old_price,
17                 stockQuantity:req.body.stockQuantity,
18             });
19             console.log(product);
20             await product.save();
21             console.log('saved');
22             res.json({
23                 success:true,
24                 name:req.body.name
25             });
26         } catch (error) {
27             console.error(error);
28             res.status(500).json({ error: 'Failed to create product' });
29         }
30     },
31
32     removeProduct: async (req, res) => {
```




```
 8
 7 const login = async (req, res) => {
 6     try {
 5         const { email, password } = req.body;
 4
 3         // Find the user by email
 2         const user = await User.findOne({email});
 1
129         console.log(user);
 1         // Check if the user exists and the password is correct
 2         if (user && (await user.comparePassword(password))) {
 3             // Generate a JWT access token with a short expiration time
 4             const accessToken = jwt.sign({ userId: user._id, email: user.email }, 'your-secret-key', { expiresIn: expireTime });
 5
 6             // Generate a refresh token with a longer expiration time
 7             const refreshToken = jwt.sign({ userId: user._id, email: user.email }, 'refresh-secret-key', { expiresIn: '1d' });
 8
 9             // Store the refresh token in the database
10             user.refreshToken = refreshToken;
11             await user.save();
12
13             res.status(200).json({ accessToken, refreshToken, role: user.role });
14         } else {
15             res.status(401).json({ error: 'Invalid credentials' });
16         }
17     } catch (error) {
18         console.error(error);
19         res.status(500).json({ error: 'Failed to authenticate user' });
20     }
21 };
22
H 23 const logout = (req, res) => {
24   try {
25     // You can perform additional tasks before or after logout if needed
26
27     res.status(200).json({ message: 'Logout successful' });
28   } catch (error) {
29     console.error(error);
30     res.status(500).json({ error: 'Failed to logout' });
31   }
32 };
33
34 module.exports = { sendOtp, register, login, refreshToken, logout };
35
```


1 2 4
Attraction
Tuesday 27 February 2024 11:01:03
50%
30%
24%
7%
PICT-WIFI
59%
'req' is declared but its value is never read.
getAllProducts: async (req, res) => {
try {
const products = await Product.aggregate([
{
$lookup: {
from: "categories",
let: { categoryIds: "$category" },
pipeline: [
{ $match: { $expr: { $in: ["$_id", "$$categoryIds"] } } }
],
as: "category"
}
},
{
$lookup: {
from: "types",
let: { typeIds: "$type" },
pipeline: [
{ $match: { $expr: { $in: ["$_id", "$$typeIds"] } } }
],
as: "type"
}
},
{
$addFields: {
category: {
$map: {
input: "$category",
as: "category",
in: "$$category.name"
}
},
type: {
$map: {
input: "$type",
as: "type",
in: "$$type.name"
}
}
}
}
]);
res.json(products);
~/Desktop/Webdev/CubeHub/backend/controllers/productController.js
105,1
47%