Github Link: https://github.com/Aaenoor/B204-App-Web-Development

```
const express=require("express");
const path = require("path");
const ejsMate = require("ejs-mate");
const methodOverride = require("method-override");
const connectToMongoDB = require('./config/db');
const multer = require('multer');
const Product = require('./models/product')
const Order = require('./models/order')
const TotalBill = require('./models/bill')
const OrderHistory = require('./models/orderHistory')
const nodemailer = require('nodemailer');
let {createPayment, executePayment, generateTotalBill } =
require('./public/js/paypal')
let {sendSuccessEmail, sendFailureEmail} = require('./public/js/emailNotifier')
require('dotenv').config();

const app = express();
const PORT = process.env.PORT || 3001;
connectToMongoDB();

// Setting for node mailer to send email
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
      user: process.env.EMAIL_NOTIFY,
      pass: process.env.EMAIL_PASSWORD
    }
});

app.set("view engine","ejs");
app.set("views",path.join(__dirname,"views"));
app.engine("ejs",ejsMate);

app.use(express.static(path.join(__dirname,"/public")));
app.use(express.static(path.join(__dirname,"/uploads")));
app.use(express.json());
app.use(express.urlencoded({extended: true}));

app.use(methodOverride("_method"));

// Multer storage for handling image uploads
const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, 'uploads/');
    },
    filename: function (req, file, cb) {
        cb(null, Date.now() + '-' + file.originalname);
    }
});

// multer upload middleware
const upload = multer({ storage: storage });

app.get('/',(req,res)=>{
    res.redirect("/ecoMarket")
})

app.get('/ecoMarket',async(req,res)=>{
    try {
```

```
        const products = await Product.find().limit(3);
        if(!products){
            res.redirect('/ecoMarket');
        }
        res.render('home', { productsData: products });
    } catch (error) {
        res.redirect('/ecoMarket');
    }
})


// Routes for Products
// GET Route
app.get('/ecoMarket/products', async (req, res) => {
    try {
        const products = await Product.find();
        if(!products){
            res.redirect('/ecoMarket');
        }
        res.render('products', { productsData: products });
    } catch (error) {
        res.redirect('/ecoMarket');
    }
});

app.get('/ecoMarket/product/:id', async (req, res) => {
    try {
        const id = req.params.id;
        await Product.findByIdAndDelete(id);
        res.redirect('/ecoMarket/products');
    } catch (error) {
        console.error('Error deleting product:', error);
        res.redirect('/ecoMarket');
    }
});

// POST Route for uploading a product with image
app.post('/ecoMarket/product', upload.single('productImage'), async (req, res) => {
    console.log("req received")
    console.log(req.body);
    console.log(req.file)
    try {
        const newProduct = new Product({
            name: req.body.name,
            price: req.body.price,
            description: req.body.description,
            category: req.body.category,
            delivery: req.body.delivery,
            warranty: req.body.warranty,
            rating: req.body.rating,
            available: req.body.available === 'true',
            image: req.file.filename,
        });
        const savedProduct = await newProduct.save();
        console.log("Product dave to database")

        res.redirect('/ecoMarket/products');

    } catch (error) {
```

```
        res.status(500).json({ error: error.message });
    }
});

// DELETE Route for deleting a product by filename
app.post('/ecoMarket/product/:name', async (req, res) => {
    try {
        const deletedProduct = await Product.findOneAndDelete({ name:
req.params.name });
        if (!deletedProduct) {
            return res.status(404).json({ error: 'Product not found' });
        }

        fs.unlinkSync(`uploads/${req.params.filename}`);

        res.json({ message: 'Product deleted successfully' });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.get('/ecoMarket/orders', async (req, res) => {
    try {
        const orders = await Order.find();
        res.json(orders);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.post('/ecoMarket/orders', async (req, res) => {
    try {
        await Order.deleteMany({});
        const newOrder = new Order();
        newOrder.items = req.body;
        const savedOrder = await newOrder.save();
        res.status(201).json(savedOrder);
    } catch (error) {
        res.status(400).json({ error: error.message });
    }
});

app.get('/ecoMarket/productsDetailedData',async(req,res)=>{
    try {
        const products = await Product.find();
        if(!products){
            res.redirect('/ecoMarket');
        }
        res.send({productsData: products})
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
})

// GET route to fetch product details based on ID
app.get('/ecoMarket/productDetails', async (req, res) => {
    const productId = req.query.id;
    const product = await Product.findById(productId);
    console.log("Proemfene fdfdfdf", product)
```

```
    res.render('productDetails', { product: product });
});


app.get('/ecoMarket/addProduct',(req,res)=>{
    res.render("addProduct")
})


app.get('/ecoMarket/ordersHistory', async (req, res) => {
    try {
        const orderHistoryData = await OrderHistory.find();
        res.render('ordersHistory', { orderHistoryData });

    } catch (error) {
        console.error('Error rendering order history page:', error);
        res.redirect('/ecoMarket');
    }
});

app.get('/ecoMarket/checkout',(req,res)=>{
    res.render("checkout")
})

app.get('/ecoMarket/contactUs',(req,res)=>{
    res.render("contactUs")
})

app.listen(PORT,()=>{
    console.log(`Server is running on http://localhost:${PORT}`);
})

// ----------------------------
// Paypal endpoints
// ----------------------------

app.get('/ecoMarket/success', async (req, res) => {
    const payerId = req.query.PayerID;
    const paymentId = req.query.paymentId;
    try {
        const paymentExecution = await executePayment(paymentId, payerId);
        let customerName = paymentExecution.payer.payer_info.first_name + ' ' +
paymentExecution.payer.payer_info.last_name
        let email = paymentExecution.payer.payer_info.email;
        let address = paymentExecution.payer.payer_info.shipping_address;
        let shippingAddress = address.line1 + ', ' + address.city + ', ' +
address.state + ', ' + address.country_code;
        const amount = paymentExecution.transactions[0].amount.total;

        const newOrderHistory = new OrderHistory({ customerName, email, amount,
shippingAddress });
        const savedOrderHistory = await newOrderHistory.save();

        await sendSuccessEmail(process.env.EMAIL_NOTIFY, transporter);

        res.render('success');

    } catch (error) {
```

```
        console.error('Error executing PayPal payment:', error);
        res.render('cancel');
    }
});

// Route to handle payment cancellation or failure
app.get('/ecoMarket/cancel', (req, res) => {
    sendFailureEmail(process.env.EMAIL_NOTIFY, transporter);
    res.render('cancel');
});

// POST route to initiate PayPal payment
app.post('/pay', async (req, res) => {
    try {
        let totalBillDocument = await TotalBill.findOne();
        let totalBill = totalBillDocument.totalBill
        console.log("Total Bill is : ",totalBill)

        const payment = await createPayment(totalBill);
        const approvalUrl = payment.links.find(link => link.rel ===
'approval_url').href;
        res.redirect(approvalUrl);
    } catch (error) {
        console.error('Error creating PayPal payment:', error);
        res.status(500).send('Failed to initiate PayPal payment');
    }
});

// POST route to calculate total bill based on orders
app.post('/ecoMarket/ordersTotalBill', async (req, res) => {
    try {
        const { bill } = req.body;

        let totalBillDocument = await TotalBill.findOne();

        if (!totalBillDocument) {
            totalBillDocument = new TotalBill({ totalBill: bill });
        } else {
            totalBillDocument.totalBill = bill;
        }
        await totalBillDocument.save();

        res.json({ totalBill: bill });
    } catch (error) {
        console.error('Error calculating total bill:', error);
        res.status(500).json({ error: 'Failed to calculate total bill' });
    }
});
```