Task 1

```
const mongoose = require('mongoose');

// Define the schema for the Product collection
const productSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    price: {
        type: Number,
        required: true
    },
    description: {
        type: String,
        required: true
    },
    createdAt: {
        type: Date,
        default: Date.now
    }
});

// Create the Product model
const Product = mongoose.model('Product', productSchema);

// Export the Product model
module.exports = Product;
```

Task 2

```
const express = require('express');
const Product = require('./models/Product');

const app = express();

// Route handler for GET '/products'
app.get('/products', async(req, res) => {
    try {
        // Retrieve all products from the "Product" collection
        const products = await Product.find({}, 'name price');

        // Send the JSON response with the products
        res.json(products);
    } catch (error) {
        // Handle any errors that occur during the request
        res.status(500).json({ error: 'Internal server error' });
    }
});

// Start the server
app.listen(3000, () => {
    console.log('Server is listening on port 3000');
});
```

Task 3

```
const jwt = require('jsonwebtoken');

function generateToken(userId, secretKey) {
    // Define the payload for the JWT
    const payload = {
        userId: userId
    };

    // Generate the JWT token
    const token = jwt.sign(payload, secretKey);

    return token;
}
```

Task 4

```
const jwt = require('jsonwebtoken');

function authenticate(req, res, next) {
    // Get the JWT token from the request headers
    const token = req.headers.authorization;

    if (!token) {
        // Token is missing, return 401 Unauthorized error
        return res.status(401).json({ error: 'Unauthorized' });
    }

    // Verify the token
    jwt.verify(token, 'your-secret-key', (err, decoded) => {
        if (err) {
            // Token verification failed, return 401 Unauthorized
error
            return res.status(401).json({ error: 'Unauthorized' });
        }

        // Token is valid, store the decoded payload in the request
object
        req.user = decoded;

        // Call the next middleware function
        next();
    });
}
```