Software Engineering

SEN-220

Progress Report for

Development Project

Backend Team:

DEPARTMENT OF COMPUTER SCIENCE

BAHRIA UNIVERSITY, KARACHI, PAKISTAN

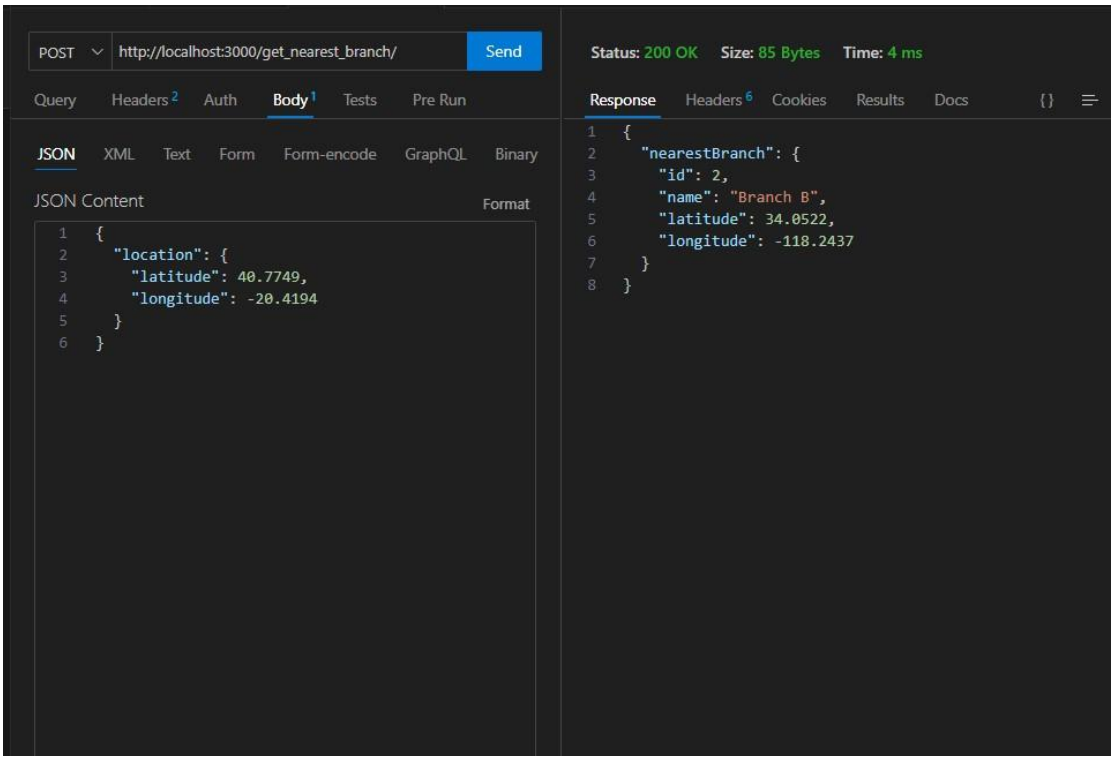
1. **Furqan Ahmed: API for GPS and Order Tracking (completed)**

DB->Connectivity:

```
PS C:\Users\ahmed\Desktop\New folder\VOEC_Backend> node "c:\Users\ahmed\Desktop\New folder\VOEC_Backend\server.js"
Server is running...
Connected to the database.
Database selected. 'se_project'
```

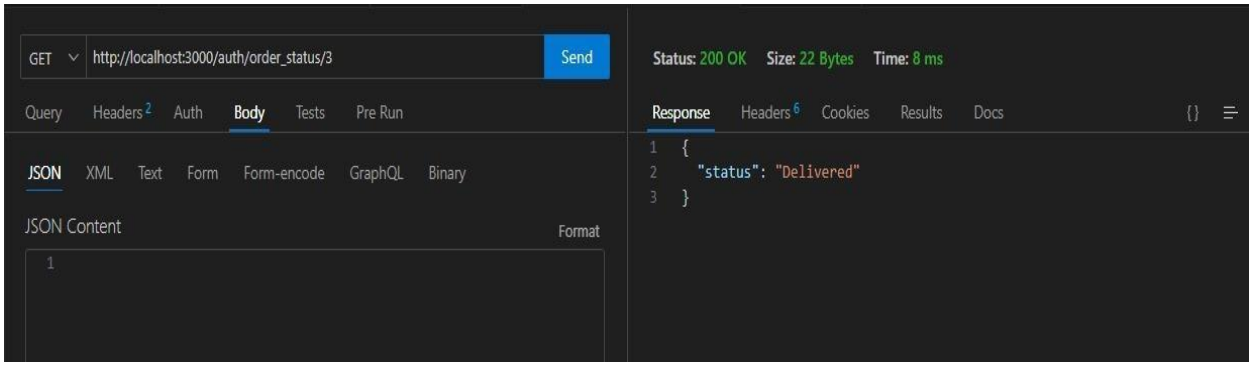
Get Nearest Branch:

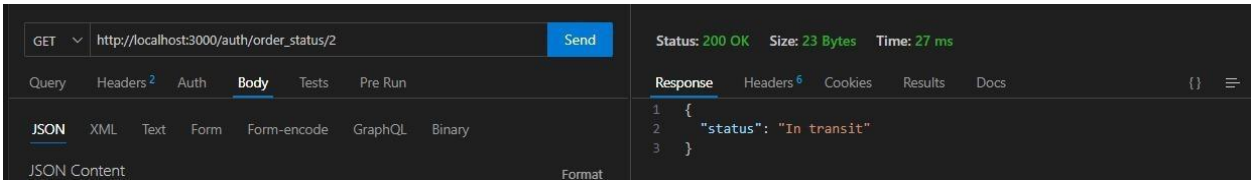
```
app.post('/get_nearest_branch', (req, res) => {
    try {

        const { latitude, longitude } = req.body.location;


        const nearestBranch = findNearestBranch(latitude, longitude);

        res.json({ nearestBranch });
    } catch (error) {
        res.status(400).json({ error: error.message });
    }
});

function findNearestBranch(customerLat, customerLng) {
    let nearestBranch = null;
    let minDistance = Infinity;

    for (const branch of branches) {
        const branchLat = branch.latitude;
        const branchLng = branch.longitude;


        const distance = Math.sqrt((customerLat - branchLat) ** 2 + (customerLng - branchLng) ** 2);


        if (distance < minDistance) {
            minDistance = distance;
            nearestBranch = branch;
        }
    }

    return nearestBranch;
```


POST
http://localhost:3000/get_nearest_branch/
Send
Query
Headers 2
Auth
Body 1
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
JSON Content
Format
{
"location": {
"latitude": 40.7749,
"longitude": -20.4194
}
}
Status: 200 OK
Size: 85 Bytes
Time: 4 ms
Response
Headers 6
Cookies
Results
Docs
{
"nearestBranch": {
"id": 2,
"name": "Branch B",
"latitude": 34.0522,
"longitude": -118.2437
}
}

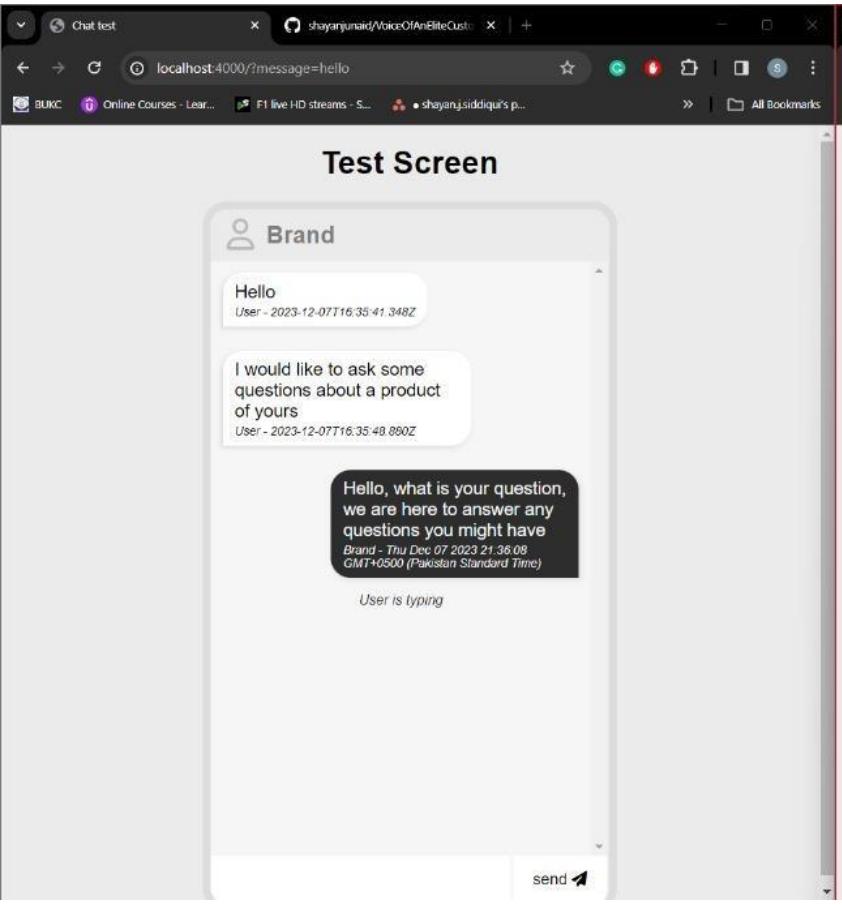
Track Order Status:

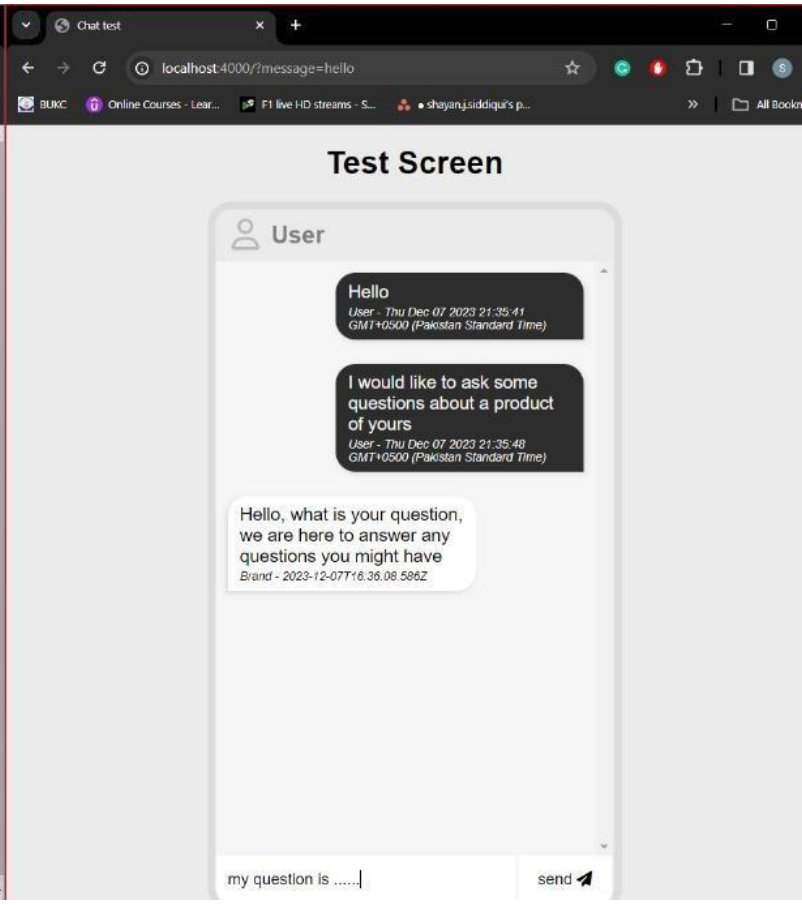
```
module.exports.order_status = (req, res) => {

    db.query('SELECT * FROM order_test WHERE o_id = ?', [req.params.id], (err, orders) => {
        if (err) throw err;

        if (orders.length === 0) {
            return res.status(404).send('Order not found');
        }

        const order = orders[0];


        db.query('SELECT latitude, longitude FROM customer WHERE CustId = ?', [order.CustomerId]
            if (err) throw err;

            const customerLocation = results[0];

            if (!customerLocation) {
                return res.status(404).send('Location not found');
            }

            console.log(customerLocation);

            const storeLocation = { latitude: 24.8067, longitude: 67.0284 };

            const percentageRemaining = calculatePercentageRemaining(customerLocation.latitude,
            currentDistance = (`${percentageRemaining.toFixed(2)}% distance remaining.`);
            console.log(currentDistance);


            let status;
            if (currentDistance >= 100) {
                status = 'Delivered';
            } else if (currentDistance >= 80) {
                status = 'Almost there';
            } else if (currentDistance >= 50) {
                status = 'In transit';
            } else if (currentDistance >= 10) {
                status = 'Dispatched';
            } else {
                status = 'Processing';
            }

            res.send({ status });
```


GET
http://localhost:3000/auth/order_status/3
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
JSON Content
Format
1
Status: 200 OK
Size: 22 Bytes
Time: 8 ms
Response
Headers 6
Cookies
Results
Docs
{}
1 {
2 "status": "Delivered"
3 }

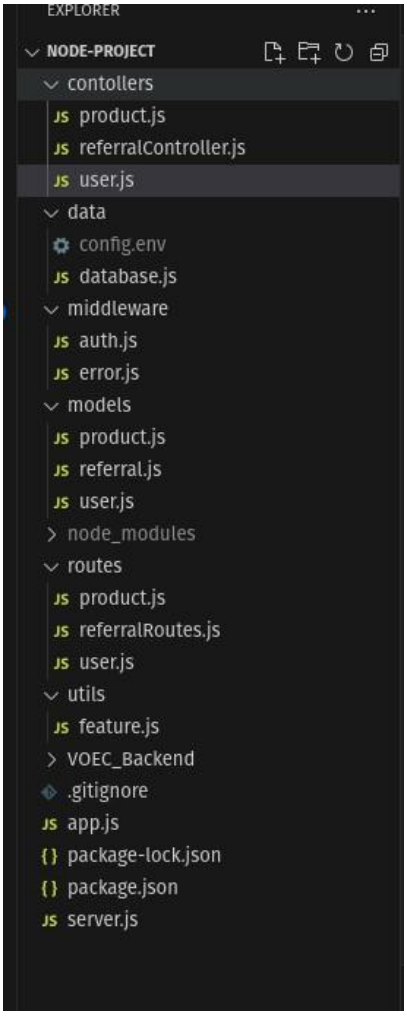

GET
http://localhost:3000/auth/order_status/2
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
JSON Content
Format
Status: 200 OK
Size: 23 Bytes
Time: 27 ms
Response
Headers 6
Cookies
Results
Docs
{}
1 {
2 "status": "In transit"
3 }

2. Shayan Junaid: Chat Box (Completed)

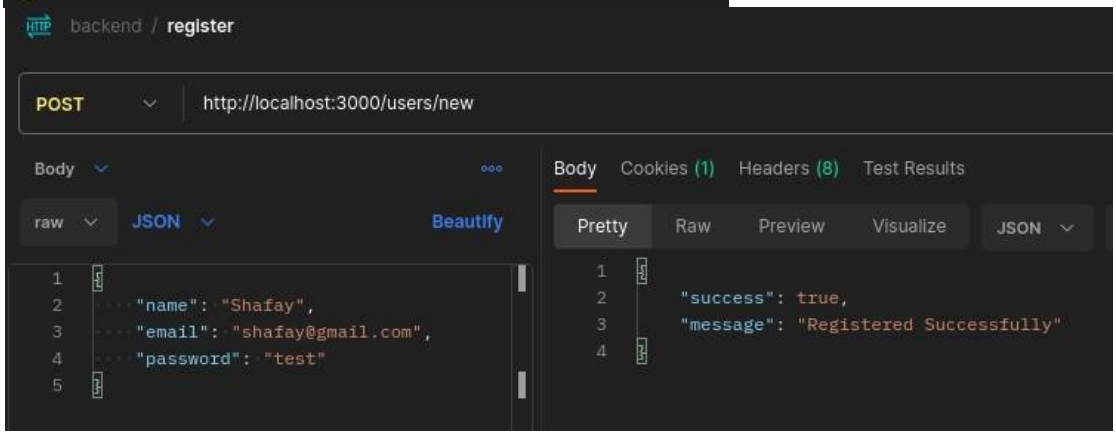


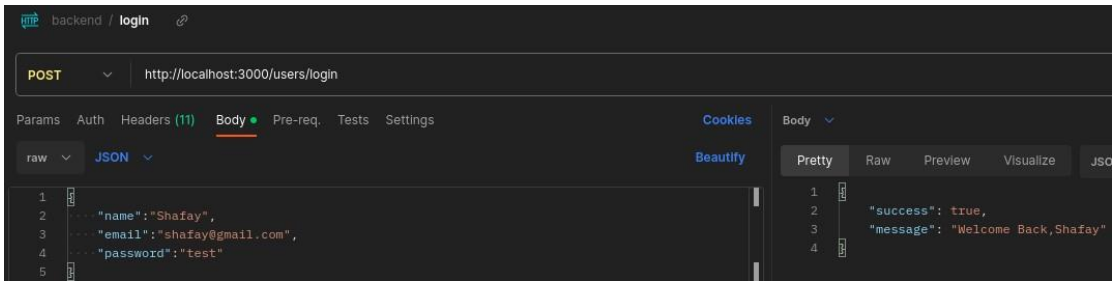


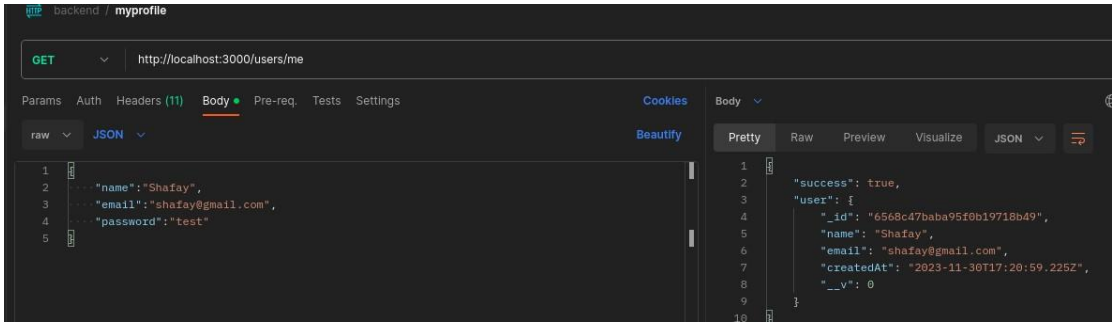
3. Shafy: Login/Signup, Referral System, Product Review (Completed)

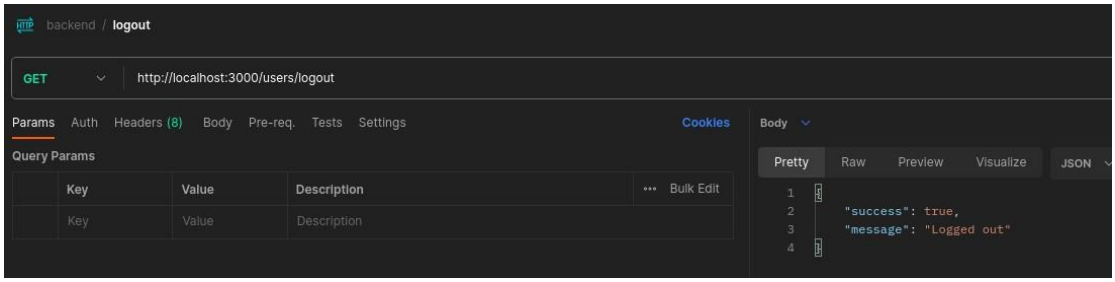


```
Codeium: Refactor | Explain | Generate JSDoc
export const getMyProfile = (req, res, next) => {

    res.status(200).json({
        success: true,
        user: req.user,
    })

}

Codeium: Refactor | Explain | Generate JSDoc
export const logout = (req, res, next) => {

    res.status(200).cookie("token", "", {
        expires: new Date(Date.now())

    }).json({
        success: true,
        message: "Logged out"
    })
}
```

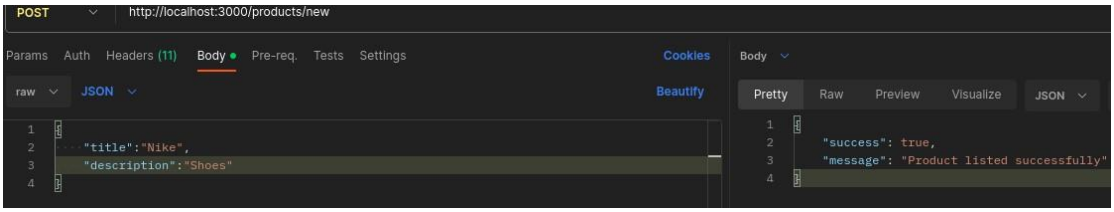


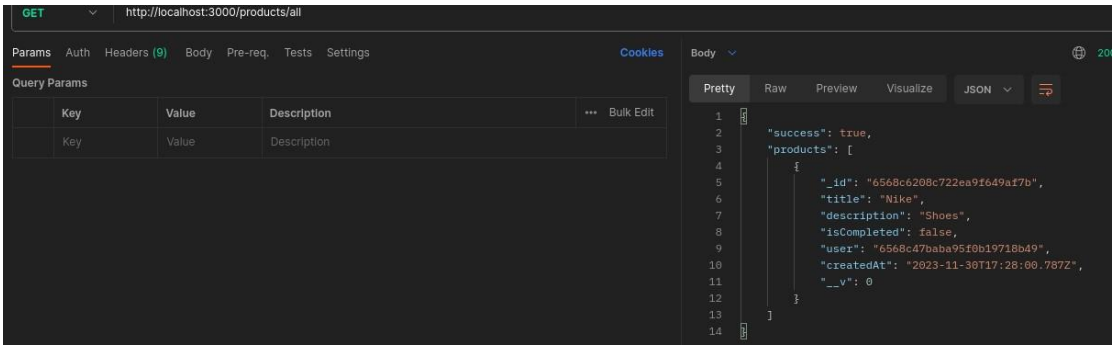


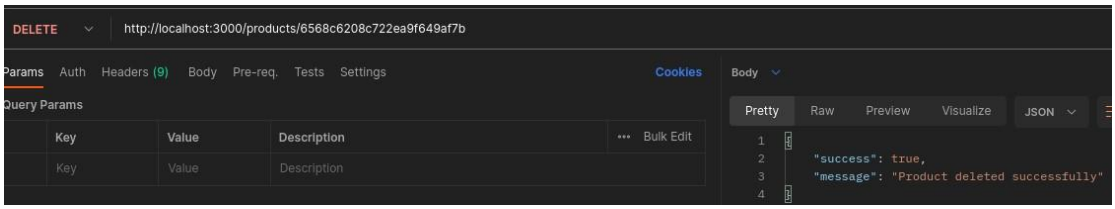


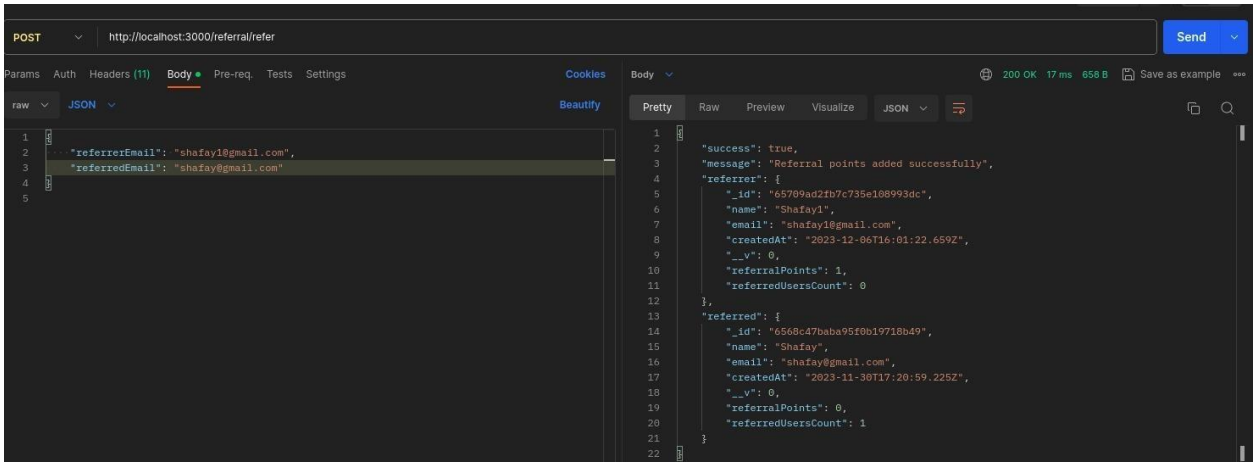



```
export const newProduct = async (req, res, next) => {
    try {
        const { title, description } = req.body;

        await Product.create({
            title,
            description,
            user: req.user,
        });

        res.status(201).json({
            success: true,
            message: "Product listed successfully"
        });
    } catch (error) {
        next(error);
    }
}

Codeium: Refactor | Explain | Generate JSDoc
export const getAllProduct = async (req, res, next) => {

    try {
        const userid = req.user._id;

    const products = await Product.find({user: userid});

    res.status(200).json({
        success: true,
        products
    })
    } catch (error) {
        next(error);
    }

}
```

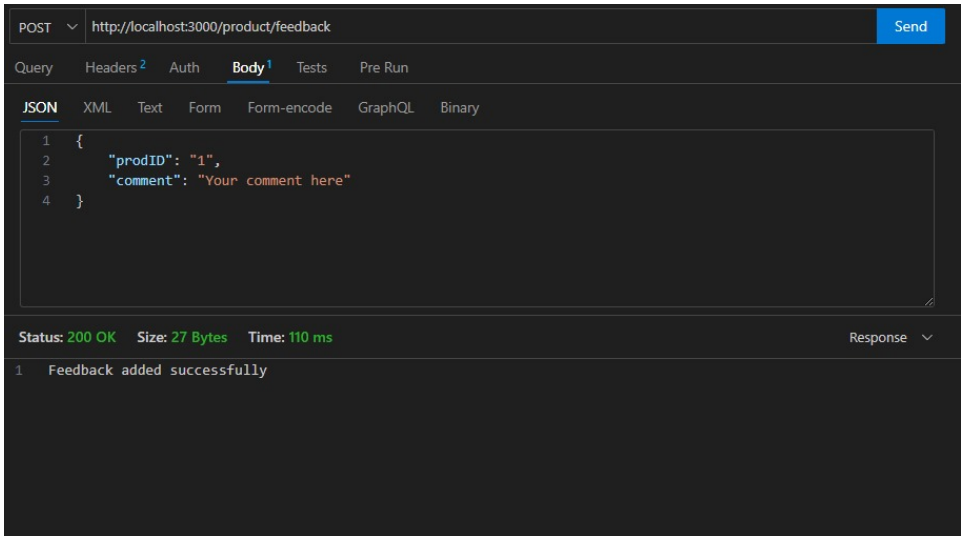

POST
http://localhost:3000/products/new
Params Auth Headers (11) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
{
"title":"Nike",
"description":"Shoes"
}
Body
Pretty Raw Preview Visualize JSON
{
"success": true,
"message": "Product listed successfully"
}


GET
http://localhost:3000/products/all
Params Auth Headers (9) Body Pre-req. Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Key Value Description
Body
Pretty Raw Preview Visualize JSON
{
"success": true,
"products": [
{
"_id": "6568c6208c722ea9f649af7b",
"title": "Nike",
"description": "Shoes",
"isCompleted": false,
"user": "6568c47baba95f0b19718b49",
"createdAt": "2023-11-30T17:28:00.787Z",
"__v": 0
}
]
}


DELETE
http://localhost:3000/products/6568c6208c722ea9f649af7b
Params Auth Headers (9) Body Pre-req. Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Key Value Description
Body
Pretty Raw Preview Visualize JSON
{
"success": true,
"message": "Product deleted successfully"
}


POST
http://localhost:3000/referral/refer
Send
Params Auth Headers (11) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
{
"referrerEmail": "shafay1@gmail.com",
"referredEmail": "shafay@gmail.com"
}
Body
200 OK 17 ms 658 B
Save as example
Pretty Raw Preview Visualize JSON
{
"success": true,
"message": "Referral points added successfully",
"referrer": {
"_id": "65709ad2fb7c735e108993dc",
"name": "Shafay1",
"email": "shafay1@gmail.com",
"createdAt": "2023-12-06T16:01:22.659Z",
"__v": 0,
"referralPoints": 1,
"referredUsersCount": 0
},
"referred": {
"_id": "6568c47baba95f0b19718b49",
"name": "Shafay",
"email": "shafay@gmail.com",
"createdAt": "2023-11-30T17:20:59.225Z",
"__v": 0,
"referralPoints": 0,
"referredUsersCount": 1
}
}

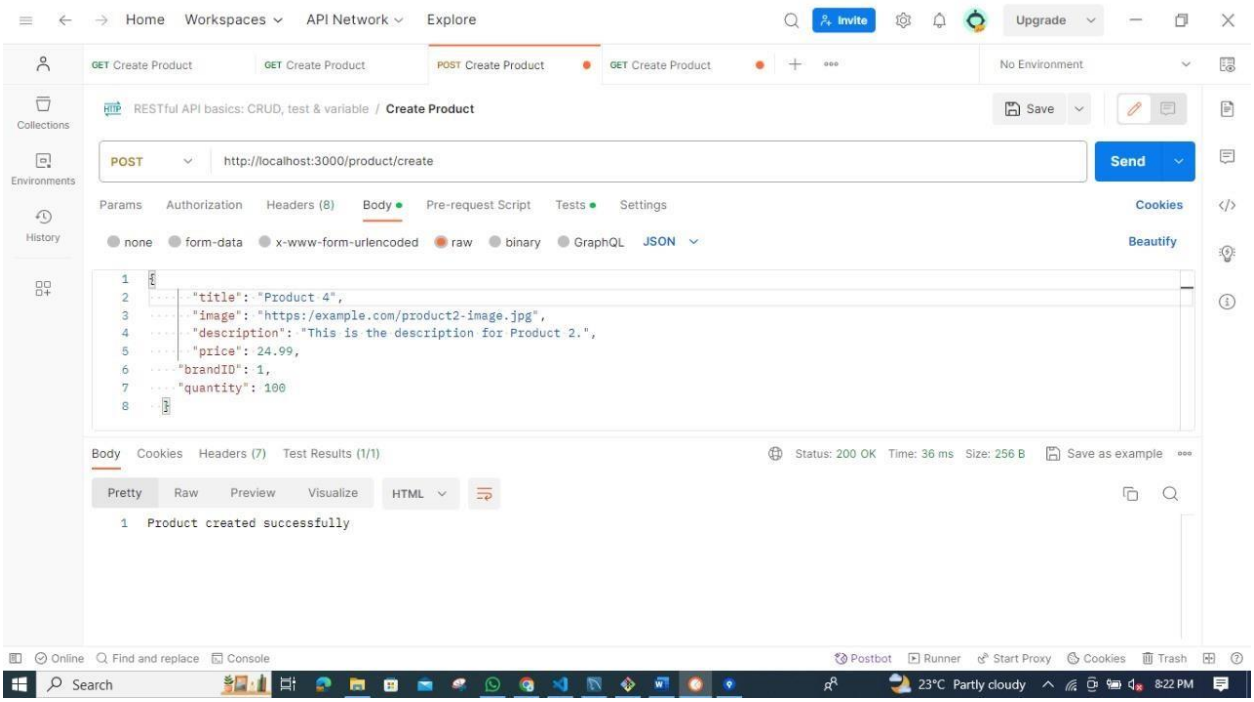

Codeium: Refactor | Explain | Generate JSDoc | ×
export const addFeedback = async (req, res, next) => {
try {
const { feedback } = req.body;
const userId = req.user._id;
const user = await User.findById(userId);
if (!user) {
return res.status(404).json({
success: false,
message: "User not found"
});
}
user.feedback = feedback;
await user.save();
res.status(200).json({
success: true,
message: "Feedback added successfully"
});
} catch (error) {
next(error);
}
};

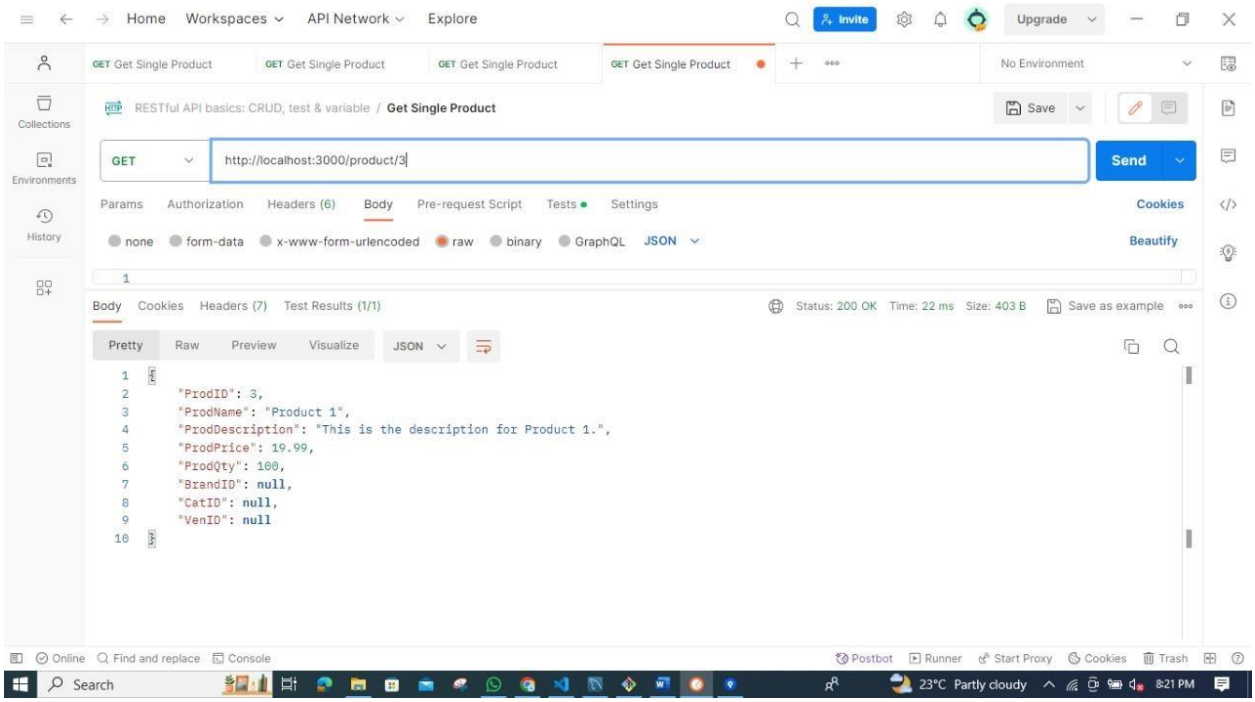


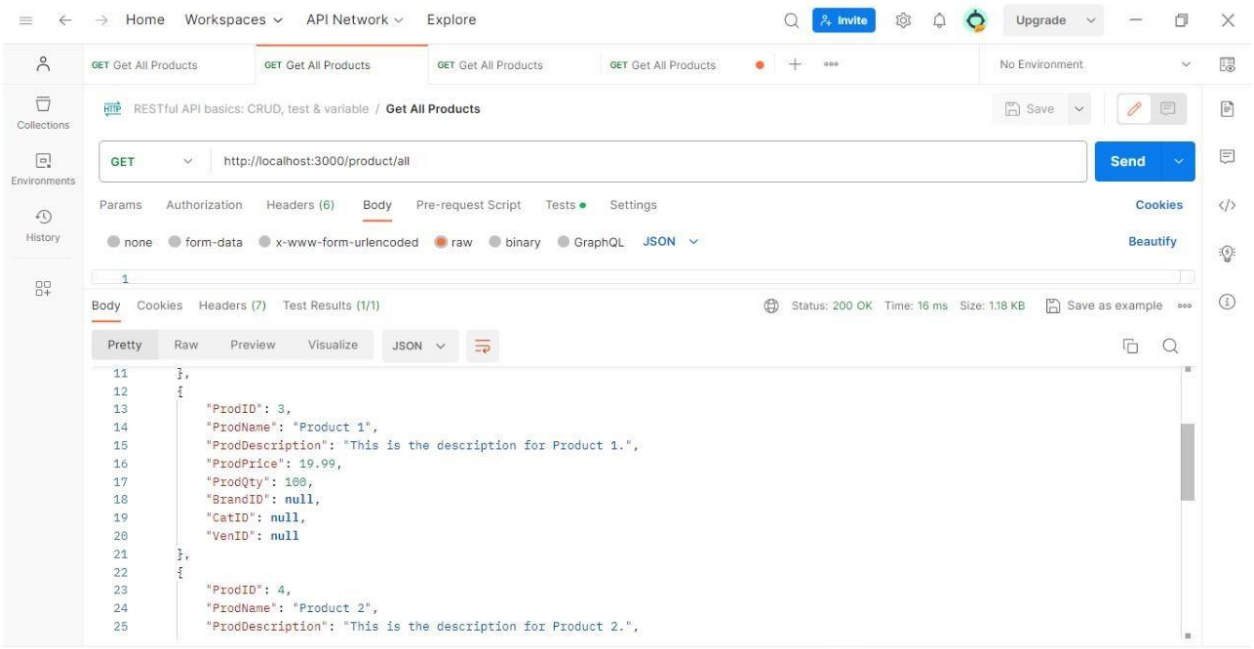

```
_id: ObjectId('6568c47baba95f0b19718b49')
name: "Shafay"
email: "shafay@gmail.com"
password: "$2b$10$rRFTnKVBblr12YDsBrEO2.ar7.9hUuV56bTvNGIDyqtPS4C30p3BC"
createdAt: 2023-11-30T17:20:59.225+00:00
__v: 0
referralPoints: 0
referredUsersCount: 1
feedback: "This is my feedback about the service."
```

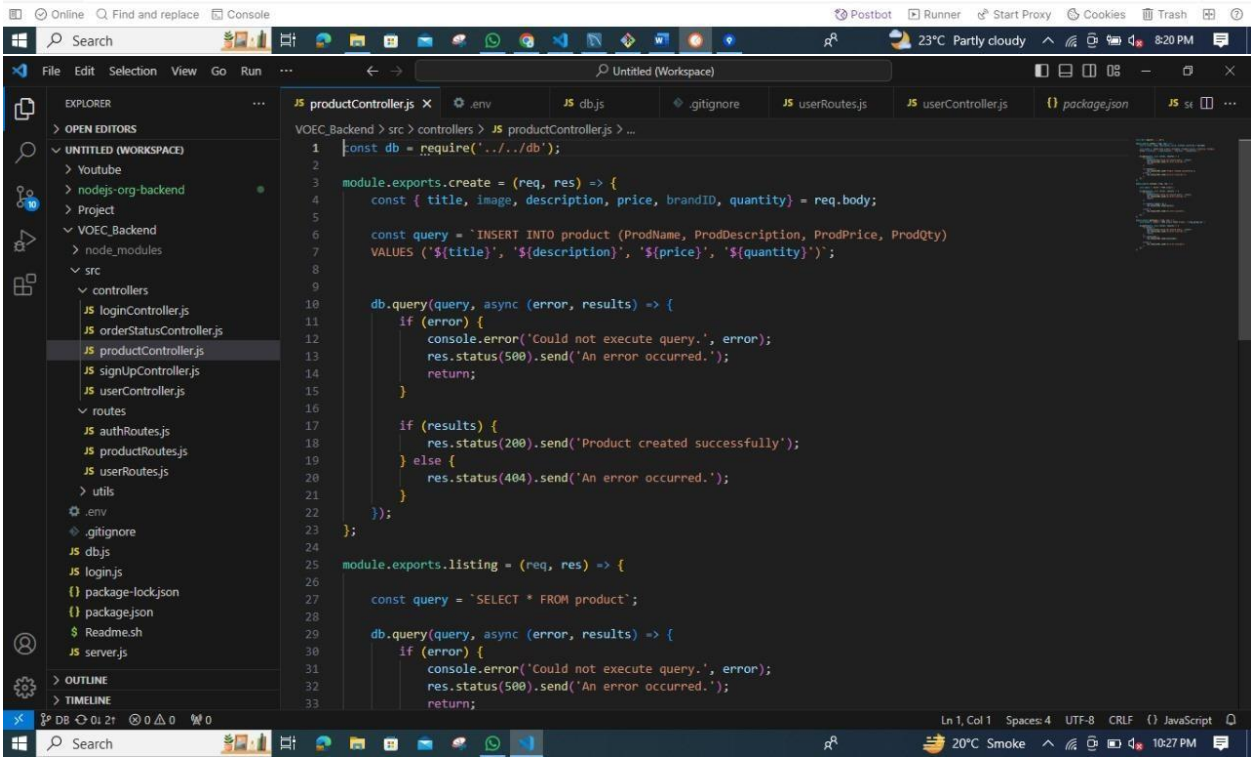
4. **Hina Shoukat: Product Listing, User profile, Brand profile and Product Comparison (Completed)**

- **Product Listing:**

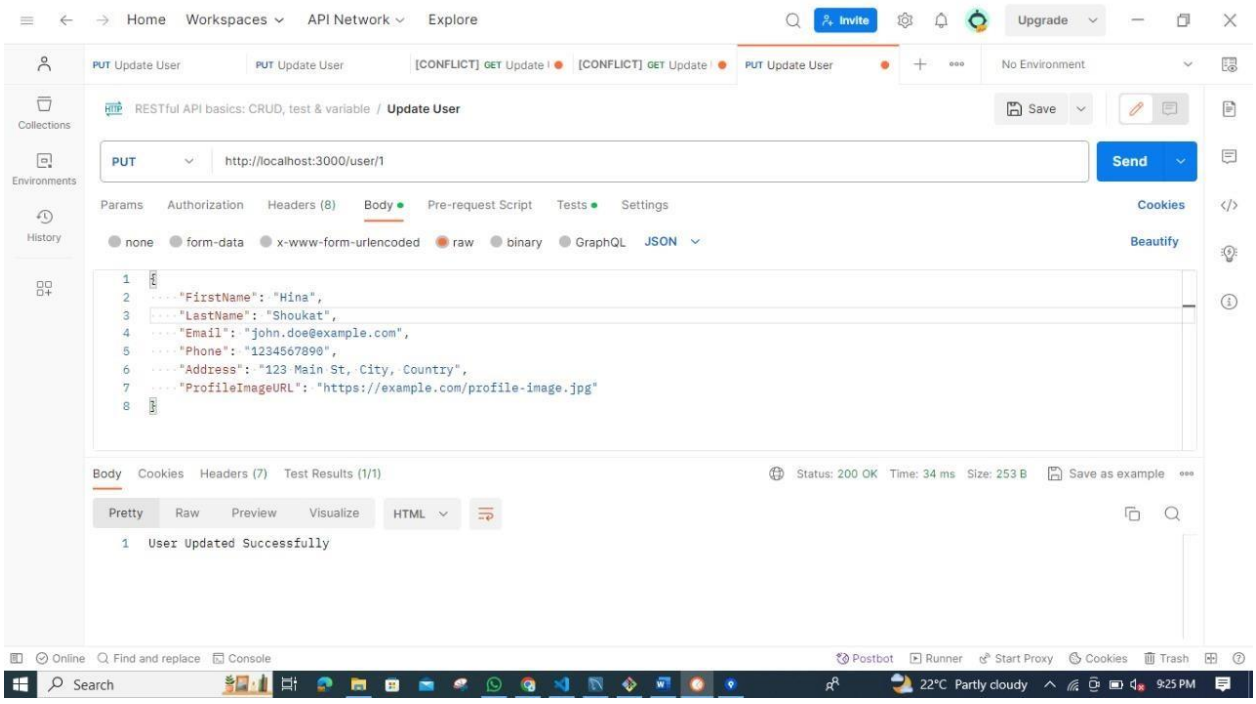


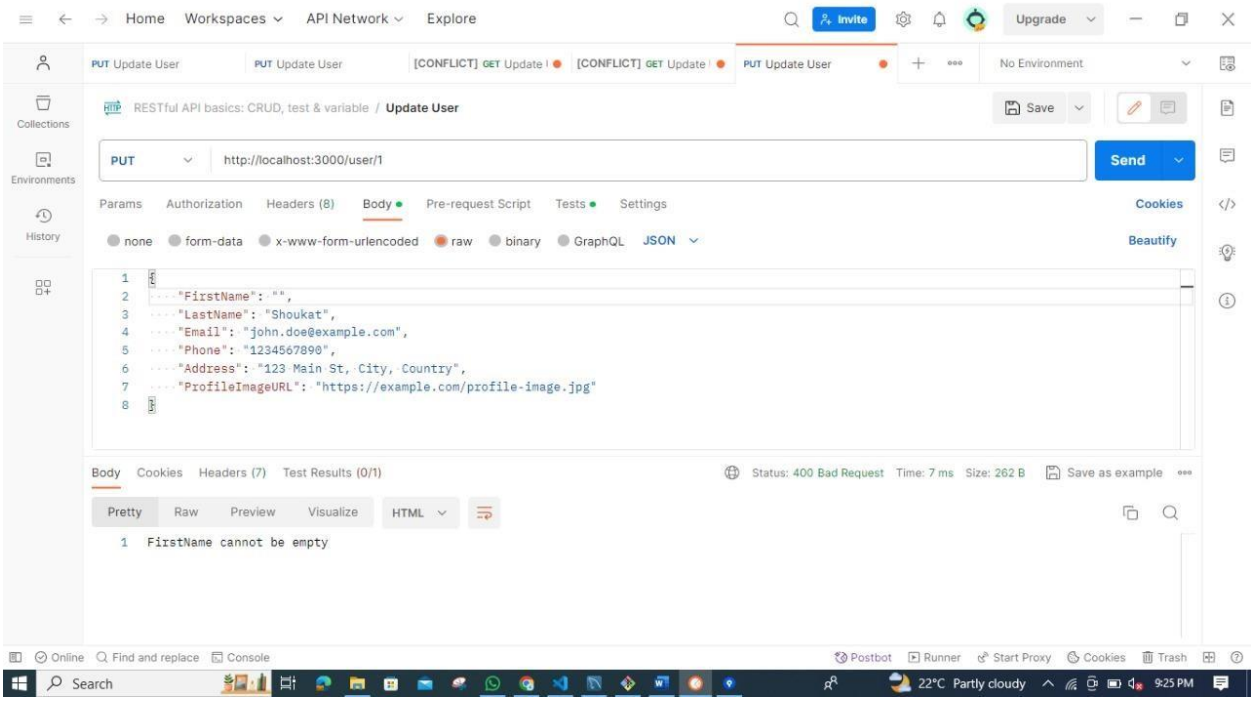


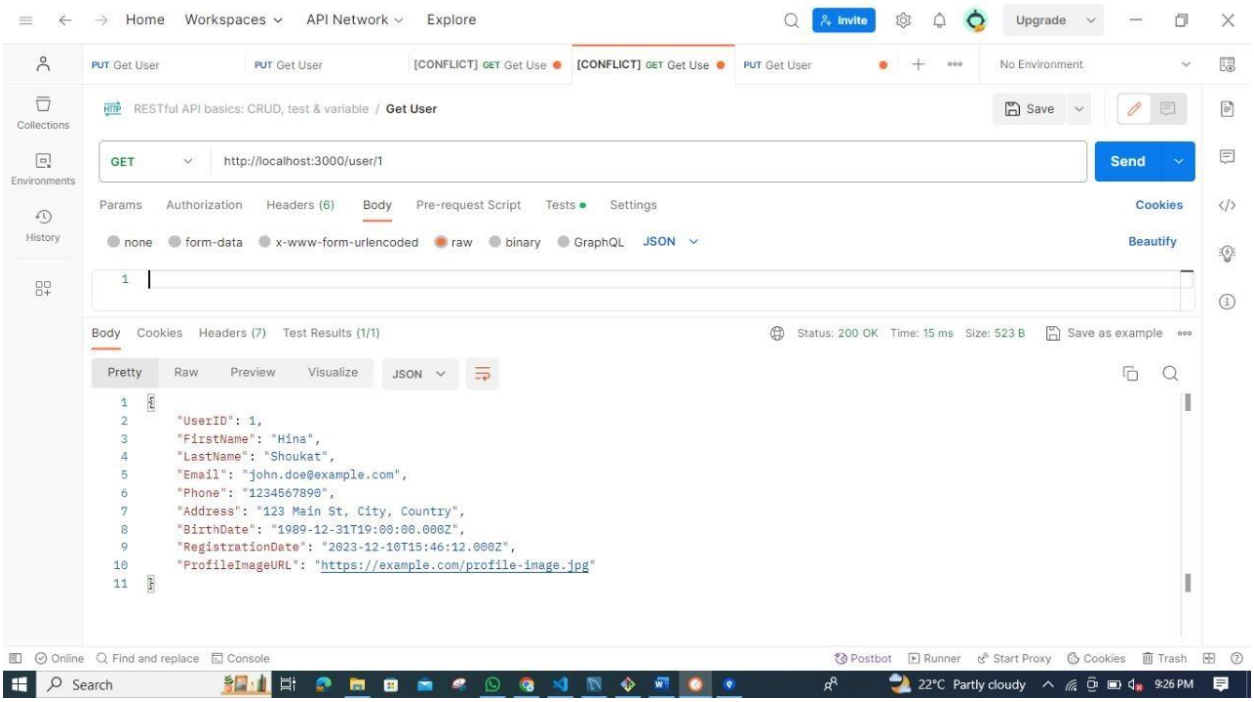




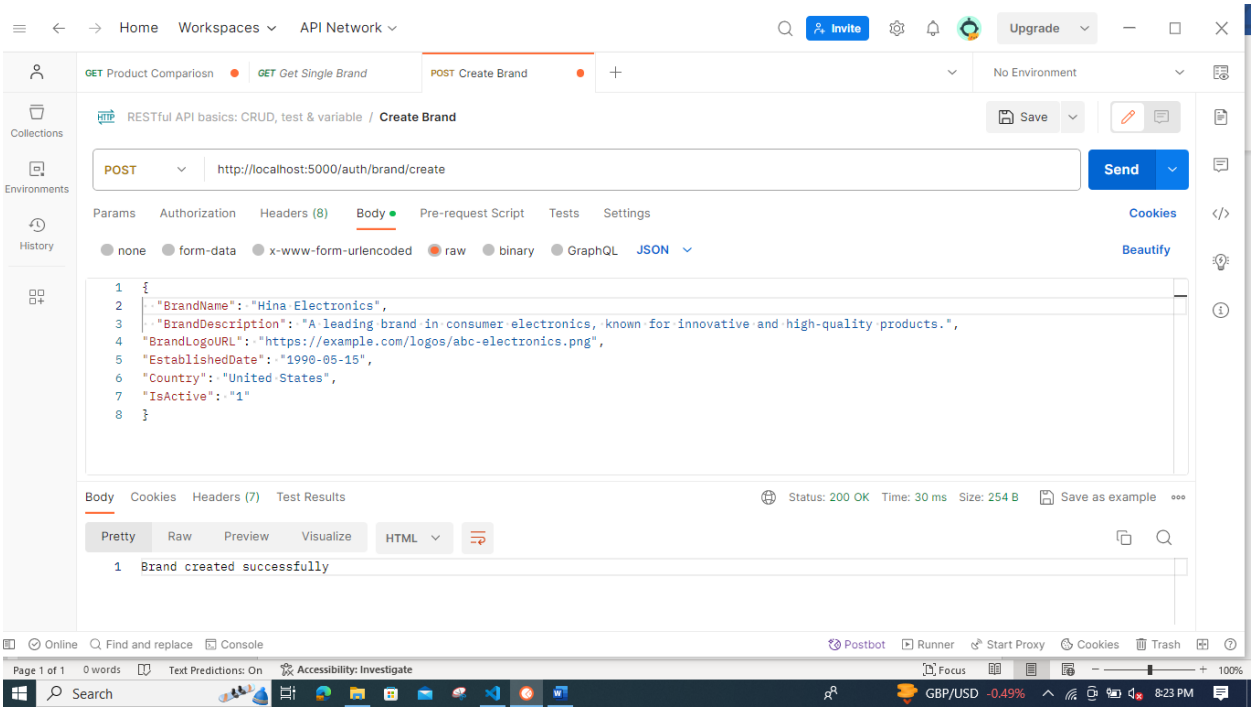
- **User profile:**

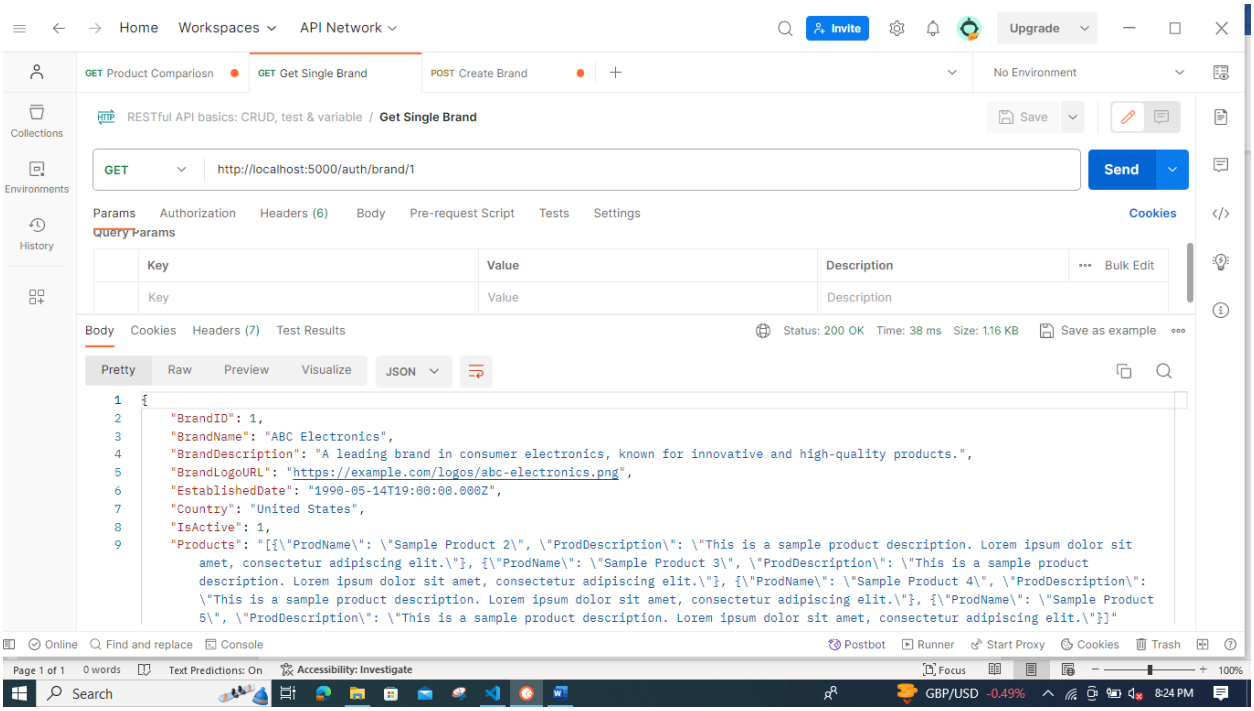

Home
Workspaces
API Network
Explore
Invite
Upgrade
PUT Update User
PUT Update User
[CONFLICT] GET Update
[CONFLICT] GET Update
PUT Update User
No Environment
Collections
Environments
History
RESTful API basics: CRUD, test & variable / Update User
Save
PUT
http://localhost:3000/user/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"FirstName": "Hina",
"LastName": "Shoukat",
"Email": "john.doe@example.com",
"Phone": "1234567890",
"Address": "123 Main St, City, Country",
"ProfileImageURL": "https://example.com/profile-image.jpg"
}
Body
Cookies
Headers (7)
Test Results (1/1)
Status: 200 OK
Time: 34 ms
Size: 253 B
Save as example
Pretty
Raw
Preview
Visualize
HTML
1 User Updated Successfully
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
22°C Partly cloudy
9:25 PM


Home
Workspaces
API Network
Explore
Invite
Upgrade
PUT Update User
PUT Update User
[CONFLICT] GET Update
[CONFLICT] GET Update
PUT Update User
No Environment
Collections
Environments
History
RESTful API basics: CRUD, test & variable / Update User
Save
PUT
http://localhost:3000/user/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"FirstName": "",
"LastName": "Shoukat",
"Email": "john.doe@example.com",
"Phone": "1234567890",
"Address": "123 Main St, City, Country",
"ProfileImageURL": "https://example.com/profile-image.jpg"
}
Body
Cookies
Headers (7)
Test Results (0/1)
Status: 400 Bad Request
Time: 7 ms
Size: 262 B
Save as example
Pretty
Raw
Preview
Visualize
HTML
1 FirstName cannot be empty
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
22°C Partly cloudy
9:25 PM

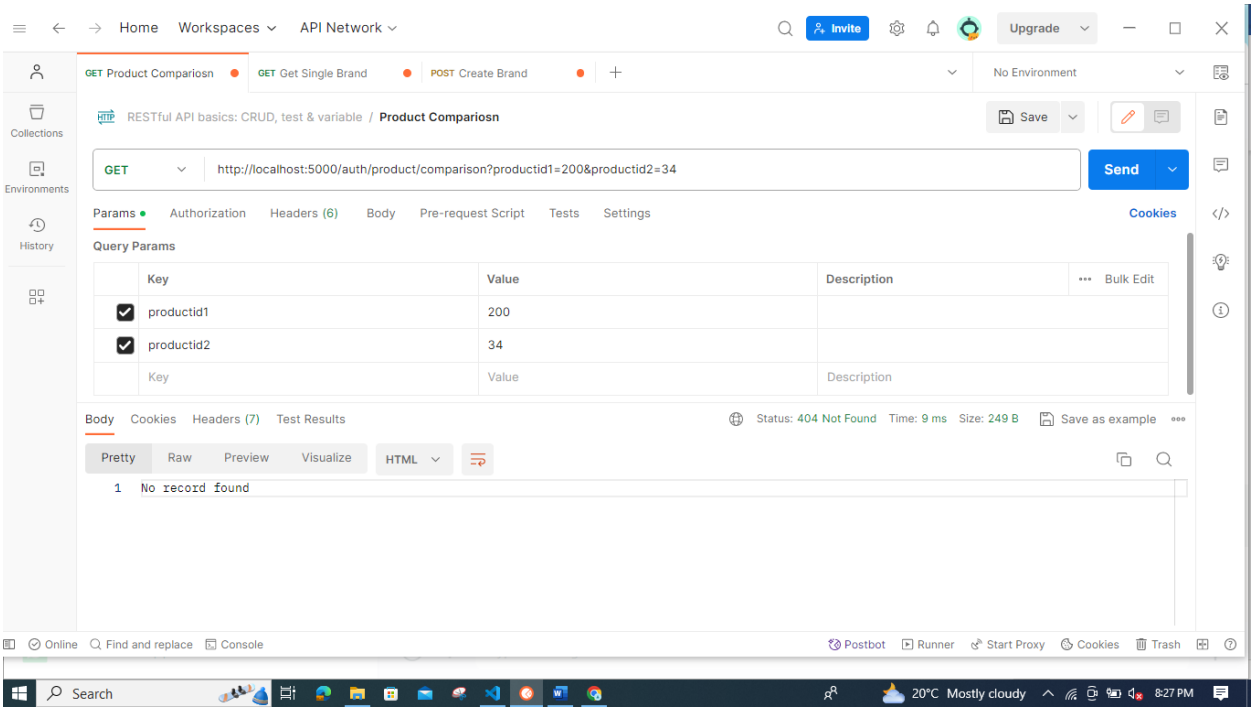


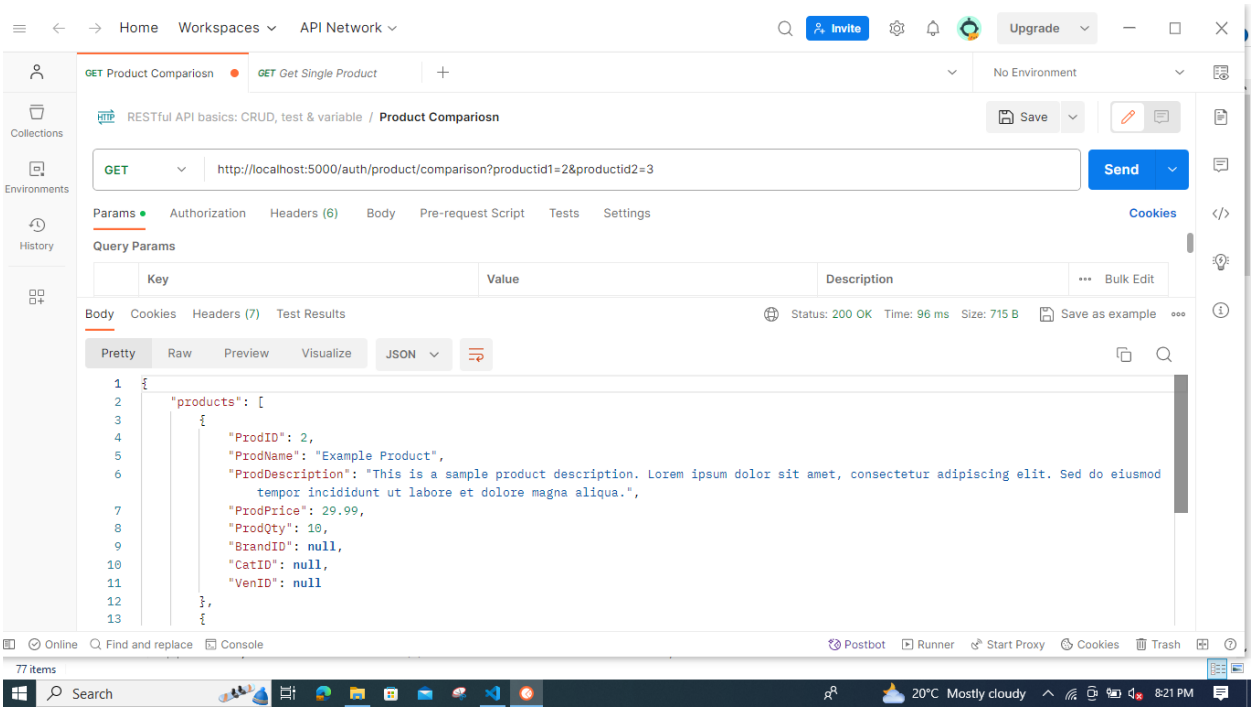
- **Brand profile:**





- **Product Comparison :**



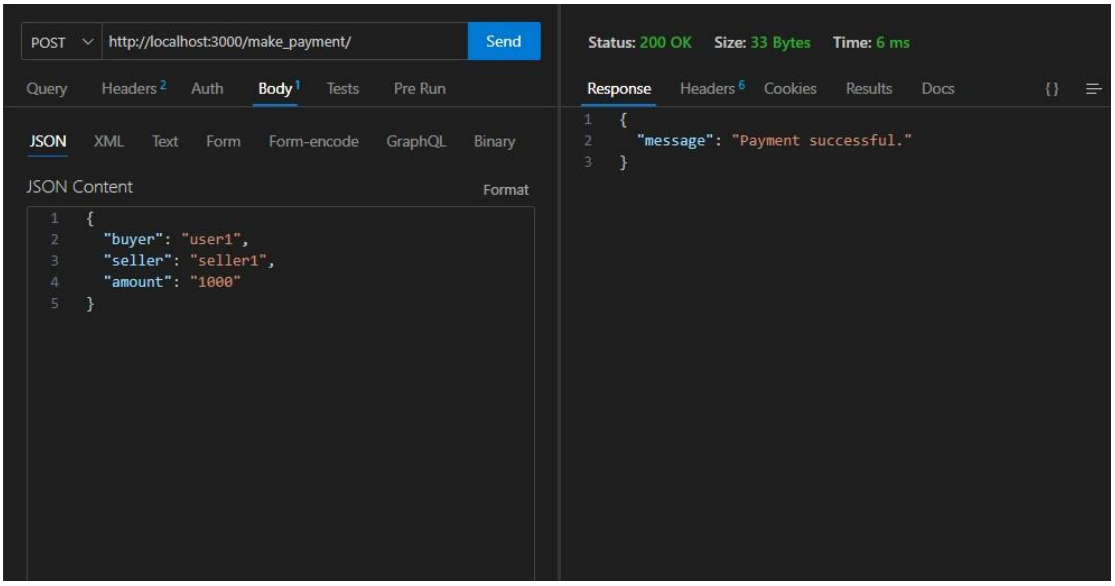

Home
Workspaces
API Network
Invite
Upgrade
GET Product Compariosn
GET Get Single Product
No Environment
Collections
Environments
History
RESTful API basics: CRUD, test & variable / Product Compariosn
Save
GET
http://localhost:5000/auth/product/comparison?productid1=2&productid2=3
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 96 ms
Size: 715 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"products": [
{
"ProdID": 2,
"ProdName": "Example Product",
"ProdDescription": "This is a sample product description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
"ProdPrice": 29.99,
"ProdQty": 10,
"BrandID": null,
"CatID": null,
"VenID": null
},
{
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
77 items
Search
20°C Mostly cloudy
8:21 PM

5. Muhammad Furqan: Payment API (Completed):

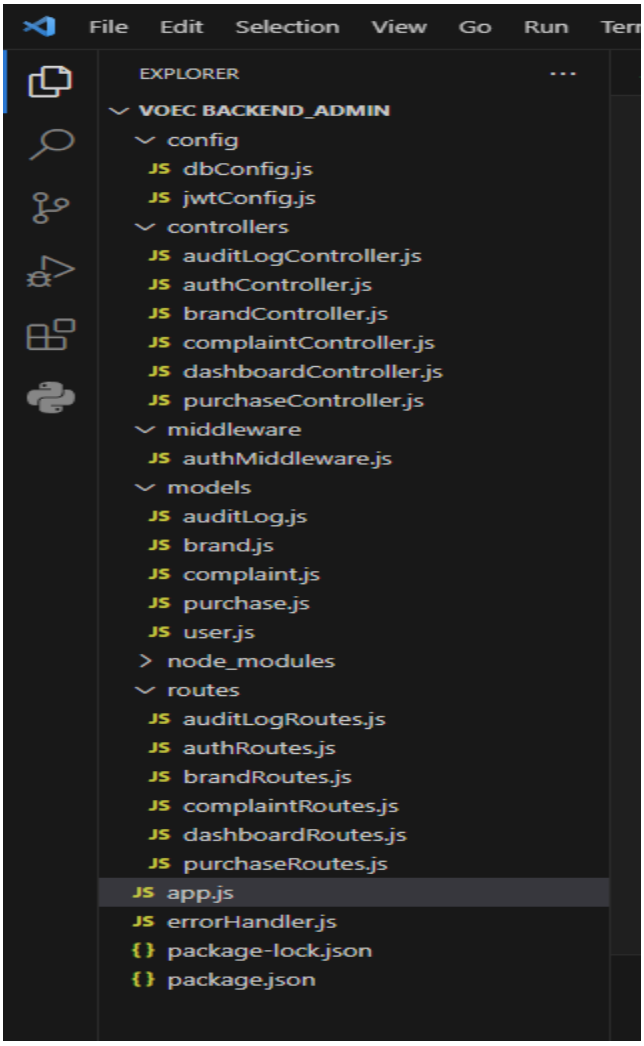
Added DB Transactions for secure payments.

```
app.post('/make_payment', (req, res) => {
    const { buyer, seller, amount } = req.body;


    const buyerExists = users.find(user => user.username === buyer);
    const sellerExists = sellers.find(user => user.username === seller);

    if (!buyerExists || !sellerExists) {
        return res.status(400).json({ error: 'Invalid buyer or seller.' });
    }


    if (buyerExists.bankDetails.balance < amount) {
        return res.status(400).json({ error: 'Insufficient funds.' });
    }


    buyerExists.bankDetails.balance -= amount;
    sellerExists.bankDetails.balance += amount;


    res.json({ message: 'Payment successful.' });
});
```

```
const db = require ('../../db');

module.exports.make_payment = async (req, res) => {
    const { user, seller, amount } = req.body;

    if (!user || !seller || !amount || amount <= 0) {
        return res.status(400).json({ error: 'Invalid request.' });
    }

    try {

        await db.beginTransaction();

        let userExists, sellerExists;
        try {
            userExists = await db.query(`SELECT balance FROM usertable WHERE UserName = '${
            sellerExists = await db.query(`SELECT balance FROM seller WHERE sellerName = '$
        }
        catch (error) {
            console.error(error);
            return res.status(500).json({ error: 'Database error' });
        }

            if (!userExists[0] || !sellerExists[0]) {
                return res.status(400).json({ error: 'User or seller does not exist' });
            }

        if (userExists.balance < amount) {
            throw new Error('Insufficient funds.');
        }

        await db.query('UPDATE usertable SET balance = balance - ? WHERE UserName = ?', [am
        await db.query('UPDATE usertable SET balance = balance + ? WHERE UserName = ?', [am

        await db.commit();

        res.json({ message: 'Payment successful.' });
    } catch (error) {

        await db.rollback();

        res.status(400).json({ error: error.message });
    }
};
```



6. Deedan Sajid:



Main admin:

app.js > ...

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const sequelize = require('./config/dbConfig');
const dashboardRoutes = require('./routes/dashboardRoutes');
const brandRoutes = require('./routes/brandRoutes');
const purchaseRoutes = require('./routes/purchaseRoutes');
const complaintRoutes = require('./routes/complaintRoutes');
const auditLogRoutes = require('./routes/auditLogRoutes');
const errorHandler = require('./utils/errorHandler');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// Routes
app.use('/api/dashboard', dashboardRoutes);
app.use('/api/brands', brandRoutes);
app.use('/api/purchases', purchaseRoutes);
app.use('/api/complaints', complaintRoutes);
app.use('/api/audit-logs', auditLogRoutes);

// Error handler middleware
app.use(errorHandler);

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

Auth controller:

controllers > authController.js > ...

```
// authController.js

const { Router } = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const User = require('../models/user');
const { secret } = require('../config/jwtConfig');

const router = Router();

router.post('/login', async (req, res) => {
  const { username, password } = req.body;

  const user = await User.findOne({ where: { username } });

  if (!user || !bcrypt.compareSync(password, user.password)) {
    return res.status(401).json({ message: 'Invalid credentials' });
  }

  const token = jwt.sign({ username }, secret, { expiresIn: '1h' });

  res.json({ token });
});

module.exports = router;
```

Auth Routes:

```
outes > JS authRoutes.js > ...
1   // authRoutes.js
2
3   const { Router } = require('express');
4   const authController = require('../controllers/authController');
5
6   const router = Router();
7
8   router.use('/auth', authController);
9
10  module.exports = router;
11
```

Brand Controller:

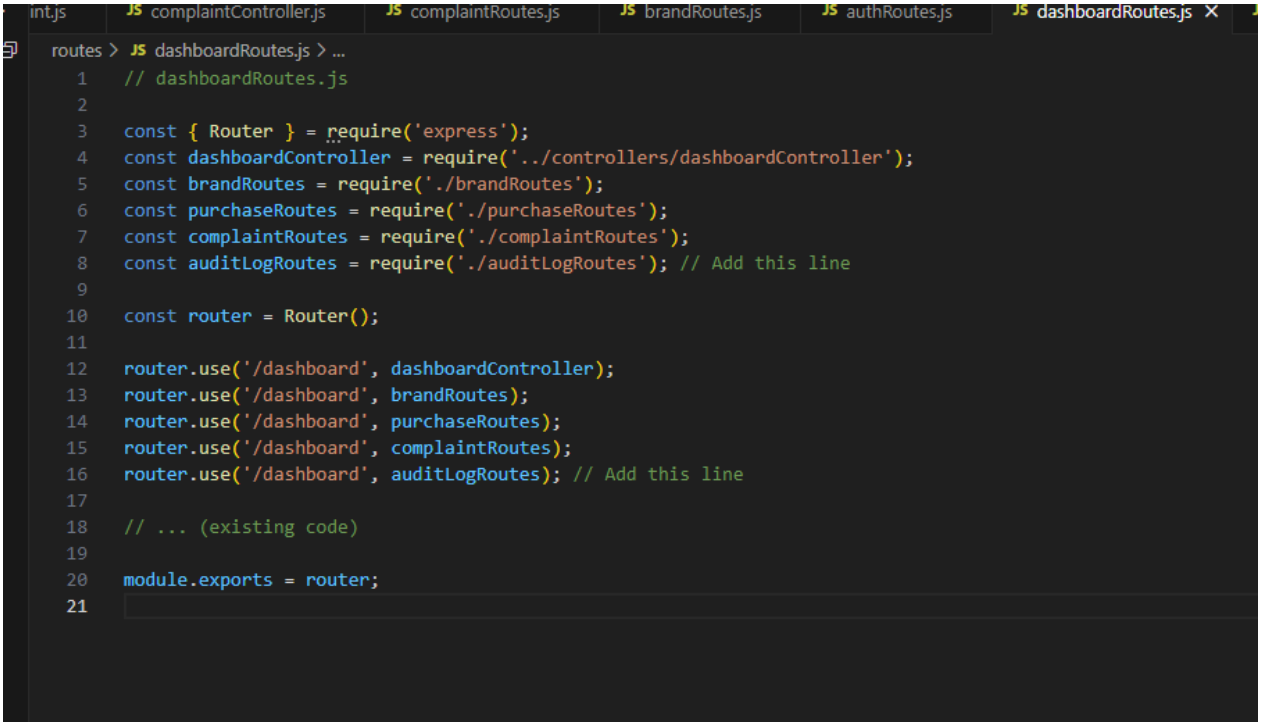
```
controllers > JS brandController.js > router.post('/create-brand') callback
1   // brandController.js
2
3   const { Router } = require('express');
4   const authenticateJWT = require('../middleware/authMiddleware');
5   const Brand = require('../models/brand');
6
7   const router = Router();
8
9   // Create brand
10  router.post('/create-brand', authenticateJWT, async (req, res) => {
11    const { name, description } = req.body;
12
13    try {
14      const brand = await Brand.create({ name, description });
15
16          // Log the brand creation action in the audit log
17          await AuditLog.create({
18              action: 'Brand Created',
19              brandId: brand.id,
20              details: `Brand ID: ${brand.id}`,
21          });
22
23      res.json({ brand });
24    } catch (error) {
25      console.error(error);
26      res.status(500).json({ message: 'Internal Server Error' });
27    }
28  });
29
30  // Update brand
31  router.put('/update-brand/:id', authenticateJWT, async (req, res) => {
32    const { id } = req.params;
33    const { name, description } = req.body;
```

Brand Route:

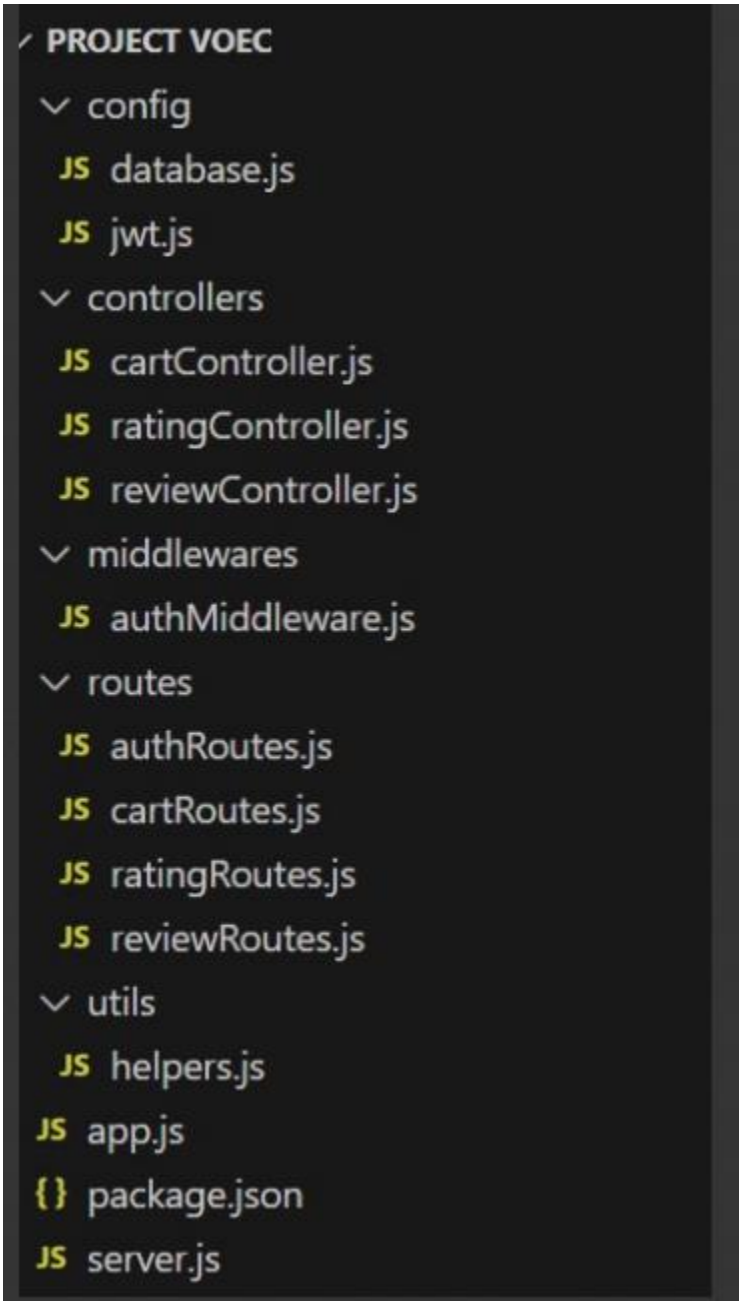
routes > JS brandRoutes.js > ...

```
// brandRoutes.js

const { Router } = require('express');
const brandController = require('../controllers/brandController');

const router = Router();

router.use('/brand', brandController);

module.exports = router;

```

Dashboard route:



7. **Ibtihaj:**



Cart Controller:

```
async function removeItemFromCart(userId, productId) {
  try {
    const query = 'DELETE FROM cart WHERE user_id = ? AND product_id = ?';
    await pool.query(query, [userId, productId]);
    return { message: 'Item removed from cart successfully' };
  } catch (error) {
    throw new Error(`Error removing item from cart: ${error}`);
  }
}

module.exports = {
  getCartItems,
  addItemToCart,
  removeItemFromCart,
};
```

```
const pool = require('../config/database');

async function getCartItems(userId) {
  try {
    const query = 'SELECT * FROM cart WHERE user_id = ?';
    const [rows] = await pool.query(query, [userId]);
    return rows;
  } catch (error) {
    throw new Error(`Error fetching cart items: ${error}`);
  }
}

async function addItemToCart(userId, productId, quantity) {
  try {
    const query = 'INSERT INTO cart (user_id, product_id, quantity) VALUES (?, ?, ?)';
    await pool.query(query, [userId, productId, quantity]);
    return { message: 'Item added to cart successfully' };
  } catch (error) {
    throw new Error(`Error adding item to cart: ${error}`);
  }
}
```

Cart Controller:

```
const pool = require('../config/database');

async function addProductRating(userId, productId, rating) {
  try {
      const existingRatingQuery = 'SELECT * FROM product_ratings WHERE user_id = ? AND product_id = ?';
    const [existingRows] = await pool.query(existingRatingQuery, [userId, productId]);

    if (existingRows && existingRows.length > 0) {
      const updateRatingQuery = 'UPDATE product_ratings SET rating = ? WHERE user_id = ? AND product_id =
      await pool.query(updateRatingQuery, [rating, userId, productId]);
      return { message: 'Rating updated successfully' };
    } else {
      const addRatingQuery = 'INSERT INTO product_ratings (user_id, product_id, rating) VALUES (?, ?, ?)';
      await pool.query(addRatingQuery, [userId, productId, rating]);
      return { message: 'Rating added successfully' };
    }
  } catch (error) {
    throw new Error(`Error adding product rating: ${error}`);
  }
}
```

```
async function getProductAverageRating(productId) {
  try {
    const query = 'SELECT AVG(rating) AS average_rating FROM product_ratings WHERE product_id = ?';
    const [rows] = await pool.query(query, [productId]);
    return rows[0].average_rating || 0;
  } catch (error) {
    throw new Error(`Error fetching product average rating: ${error}`);
  }
}

module.exports = {
  addProductRating,
  getProductAverageRating,
};
```

Review Controller:

```
const pool = require('../config/database');

async function addProductReview(userId, productId, reviewText) {
  try {
    const addReviewQuery = 'INSERT INTO product_reviews (user_id, product_id, review_text) VALUES (?, ?, ?
    await pool.query(addReviewQuery, [userId, productId, reviewText]);
    return { message: 'Review added successfully' };
  } catch (error) {
    throw new Error(`Error adding product review: ${error}`);
  }
}

async function getProductReviews(productId) {
  try {
    const query = 'SELECT * FROM product_reviews WHERE product_id = ?';
    const [rows] = await pool.query(query, [productId]);
    return rows;
  } catch (error) {
    throw new Error(`Error fetching product reviews: ${error}`);
  }
}
```

```
async function viewProductReview(reviewId) {
    try {
      const query = 'SELECT * FROM product_reviews WHERE id = ?';
      const [rows] = await pool.query(query, [reviewId]);
      if (rows.length === 0) {
        return { message: 'Review not found' };
      }
      return rows[0];
    } catch (error) {
      throw new Error(`Error fetching product review: ${error}`);
    }
  }

  module.exports = {
    addProductReview,
    getProductReviews,
    viewProductReview,
  };

```

Review controller:

```
const express = require('express');
const authenticateToken = require('../middlewares/authMiddleware');
const ratingController = require('../controllers/ratingController');

const router = express.Router();

// Route to add a product rating by a user
router.post('/products/:productId/rate', authenticateToken, async (req, res) => {
  const { productId } = req.params;
  const { rating } = req.body;
  const userId = req.user.id; // Assuming user ID is available in req.user

  try {
    const result = await ratingController.addProductRating(userId, productId, rating);
    res.json(result);
  } catch (error) {
    res.status(500).json({ message: `Internal Server Error: ${error.message}` });
  }
});

// Route to get average rating for a product
router.get('/products/:productId/average-rating', async (req, res) => {
  const { productId } = req.params;

  try {
    const averageRating = await ratingController.getProductAverageRating(productId);
    res.json({ averageRating });
```

Link to all the workdone: [furqanx11/VOEC_Backend (github.com)]