

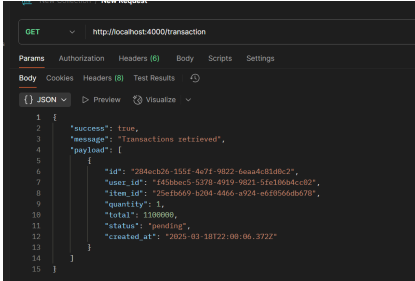
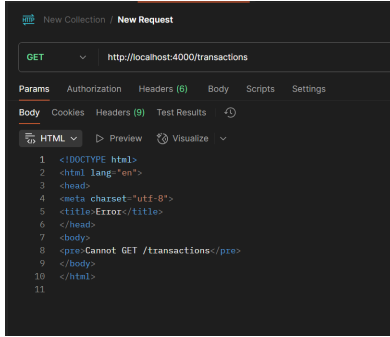
PRAKTIKUM SISTEM BASIS DATA

Nama	Deandro Najwan Ahmad Syahbanna	No. Modul	6
NPM	2306213174	Tipe	TUTAM

- Menambahkan Bagian Yang Kurang, Yaitu untuk mendapatkan data dari semua transaksi yang ada (getAllTransactions)

File	Screenshot (Penambahan)
transaction.controller.js	<pre> exports.getAllTransactions = async (req, res) => { try { const transactions = await transactionRepository.getAllTransactions(); if(transactions.length === 0) { return baseResponse(res, false, 404, "Transactions not found", null); } baseResponse(res, true, 200, "Transactions retrieved", transactions); } catch (error) { baseResponse(res, false, 500, "An error occurred while retrieving transactions", null); } } </pre>
transaction.repository.js	<pre> exports.getAllTransactions = async () => { try { const res = await db.query(`SELECT * FROM transactions`); return res.rows; } catch (error) { console.error("Error Executing query", error); } } </pre>
transaction.route.js	<pre> router.get('/', transactionController.getAllTransactions); </pre>

2. Response Format

method	endpoint	success	false
GET	/transaction		

3. Implementasi Peningkatan Kode terhadap Scalability, Security dan Error Handling

Tipe	Implementasi	Deskripsi
Error Handling	<pre> exports.userRegister = async (req, res) => { const { name, email, password } = req.query; if (!emailRegex.test(email)) { return baseResponse(res, false, 400, "Invalid email format", null); } if (!name !email !password) { return baseResponse(res, false, 400, "Missing user information", null); } try { const hashPassword = await bcrypt.hash(password, 10); const userExist = await userRepository.getUserByEmail(email); if (userExist) { return baseResponse(res, false, 400, "Email already registered", null); } const user = await userRepository.userRegister({ name, email, password: hashPassword, balance: 0 }); baseResponse(res, true, 201, "User created", user); } catch (error) { baseResponse(res, false, 500, "An error occurred while registering user", null); } }; </pre>	Menambah checking pada info user jika apakah sudah ada akun yang menggunakan email sama, maka dia tidak bisa membuat akun dengan email yang sama tersebut
Security	<pre> exports.userTopUp = async (req, res) => { const { id, amount } = req.query; if (!id){ return baseResponse(res, false, 400, "Missing user ID", null); } if (amount === undefined) { return baseResponse(res, false, 400, "Missing amount information", null); } if (amount <= 0){ return baseResponse(res, false, 400, "You Cant top up negative or zero amount", null); } try { const user = await userRepository.userTopUp(id, amount); if (!user) { return baseResponse(res, false, 404, "User not found", null); } baseResponse(res, true, 200, "User balance updated successfully", user); } catch (error) { baseResponse(res, false, 500, "An error occurred while updating user balance", null); } } </pre>	Menambah checking pada amount yang dimaksudkan untuk mencegah dimana user melakukan top up dengan sembarang semisal dia memasukkan value yang kurang dari 0 (negatif)

<p>Error Handling</p>	<pre> exports.createTransaction = async (req, res) => { const {item_id, user_id, quantity} = req.body; if (!item_id !user_id !quantity) { return baseResponse(res, false, 400, "Missing transaction information", null); } if(quantity <= 0) { return baseResponse(res, false, 400, "You need more than one quantity", null); } const matchedItem = await itemRepository.getItemById(item_id); if (!matchedItem) { return baseResponse(res, false, 404, "Item not found", null); } const matchedUser = await userRepository.getUserById(user_id); if (!matchedUser) { return baseResponse(res, false, 404, "User not found", null); } try { const transaction = await transactionRepository.createTransaction({ item_id, user_id, quantity, status: "pending" }); baseResponse(res, true, 201, "Transaction created", transaction); } catch (error) { baseResponse(res, false, 500, "An error occurred while creating transaction", null); } } </pre>	<p>Memastikan bahwa quantity yang dimaksudkan dalam transaction lebih dari 0, karena kalau misal 0 atau kurang dari 0 maka tidak akan masuk akal untuk membuat transaction di awal.</p>
<p>Security</p>	<pre> exports.createTransaction = async (req, res) => { const {item_id, user_id, quantity} = req.body; if (!item_id !user_id !quantity) { return baseResponse(res, false, 400, "Missing transaction information", null); } if(quantity <= 0) { return baseResponse(res, false, 400, "You need more than one quantity", null); } const matchedItem = await itemRepository.getItemById(item_id); if (!matchedItem) { return baseResponse(res, false, 404, "Item not found", null); } const matchedUser = await userRepository.getUserById(user_id); if (!matchedUser) { return baseResponse(res, false, 404, "User not found", null); } try { const transaction = await transactionRepository.createTransaction({ item_id, user_id, quantity, status: "pending" }); baseResponse(res, true, 201, "Transaction created", transaction); } catch (error) { baseResponse(res, false, 500, "An error occurred while creating transaction", null); } } </pre>	<p>Menambahkan function yang mengecek apakah item atau user yang dimaksud benaran ada dan jika tidak maka database akan memberikan response penolakan</p>

Scalability

```
exports.createStore = async (req, res) => {
  const { name, address } = req.body;
  if (!name || !address) {
    return res.status(400).json({
      success: false,
      message: "Missing store name or address",
      payload: null
    });
  }
  try {
    const store = await storeRepository.createStore(req.body);
    const ifStoreNameExist = await storeRepository.getStoreByName(name);
    if (ifStoreNameExist) {
      return res.status(400).json({
        success: false,
        message: "Store name already exist, please use another name",
        payload: null
      });
    }
    baseResponse(res, true, 201, "Store created successfully", store);
  } catch (error) {
    console.error("Error executing query:", error.message);
    res.status(500).json({
      success: false,
      message: "Internal Server Error",
      payload: null
    });
  }
};
```

Memastikan bahwa tidak ada toko dengan nama yang sama terbuat lebih dari sekali, agar mencegah adanya toko dengan nama yang sama dan menyebabkan kebingungan di dalam pasar

Error Handling

```
exports.payTransaction = async (req, res) => {
  const { id } = req.params;
  if (!id) {
    return baseResponse(res, false, 400, "Missing transaction ID", null);
  }

  try {
    const transactionData = await transactionRepository.getTransactionWithId(id);
    if (!transactionData) {
      return baseResponse(res, false, 404, "Transaction not found", null);
    }

    if (transactionData.status === 'paid') {
      return baseResponse(res, false, 400, "Transaction already paid", null);
    }

    if (transactionData.quantity > transactionData.item_stock) {
      return baseResponse(res, false, 400, "Not enough stock", null);
    }

    if (transactionData.total > transactionData.user_balance) {
      return baseResponse(res, false, 400, "Not enough balance", null);
    }

    // Update item stock
    const updatedItemStock = transactionData.item_stock - transactionData.quantity;
    const updatedItem = await itemRepository.updateItem(transactionData.item_id, { stock: updatedItemStock });

    if (!updatedItem) {
      return baseResponse(res, false, 500, "Failed to update item stock", null);
    }

    const updatedUserBalance = transactionData.user_balance - transactionData.total;
    const updatedUser = await userRepository.updateUser(transactionData.user_id, { balance: updatedUserBalance });

    if (!updatedUser) {
      return baseResponse(res, false, 500, "Failed to update user balance", null);
    }
  } catch (error) {
    console.error("Error executing query:", error.message);
    return baseResponse(res, false, 500, "Internal Server Error", null);
  }
};
```

Memastikan bahwa jumlah item dan juga jumlah balance yang dimiliki oleh user sudah memenuhi ketentuan yang ada sehingga bisa melakukan transaksi ini.

Security

```
exports.payTransaction = async (req, res) => {
  const { id } = req.params;
  if (!id) {
    return baseResponse(res, false, 400, "Missing transaction ID", null);
  }

  try {
    const transactionData = await transactionRepository.getTransactionWithId(id);
    if (!transactionData) {
      return baseResponse(res, false, 404, "Transaction not found", null);
    }

    if (transactionData.stat === "PAID") {
      return baseResponse(res, false, 400, "Transaction already paid", null);
    }

    if (transactionData.quantity > transactionData.item_stock) {
      return baseResponse(res, false, 400, "Not enough stock", null);
    }

    if (transactionData.total > transactionData.user_balance) {
      return baseResponse(res, false, 400, "Not enough balance", null);
    }

    // Update item stock
    const updatedItemStock = transactionData.item_stock - transactionData.quantity;
    const updatedItem = await itemRepository.updateItem(transactionData.item_id, { stock: updatedItemStock });

    if (!updatedItem) {
      return baseResponse(res, false, 500, "Failed to update item stock", null);
    }

    const updatedUserBalance = transactionData.user_balance - transactionData.total;
    const updatedUser = await userRepository.updateUser(transactionData.user_id, { balance: updatedUserBalance });

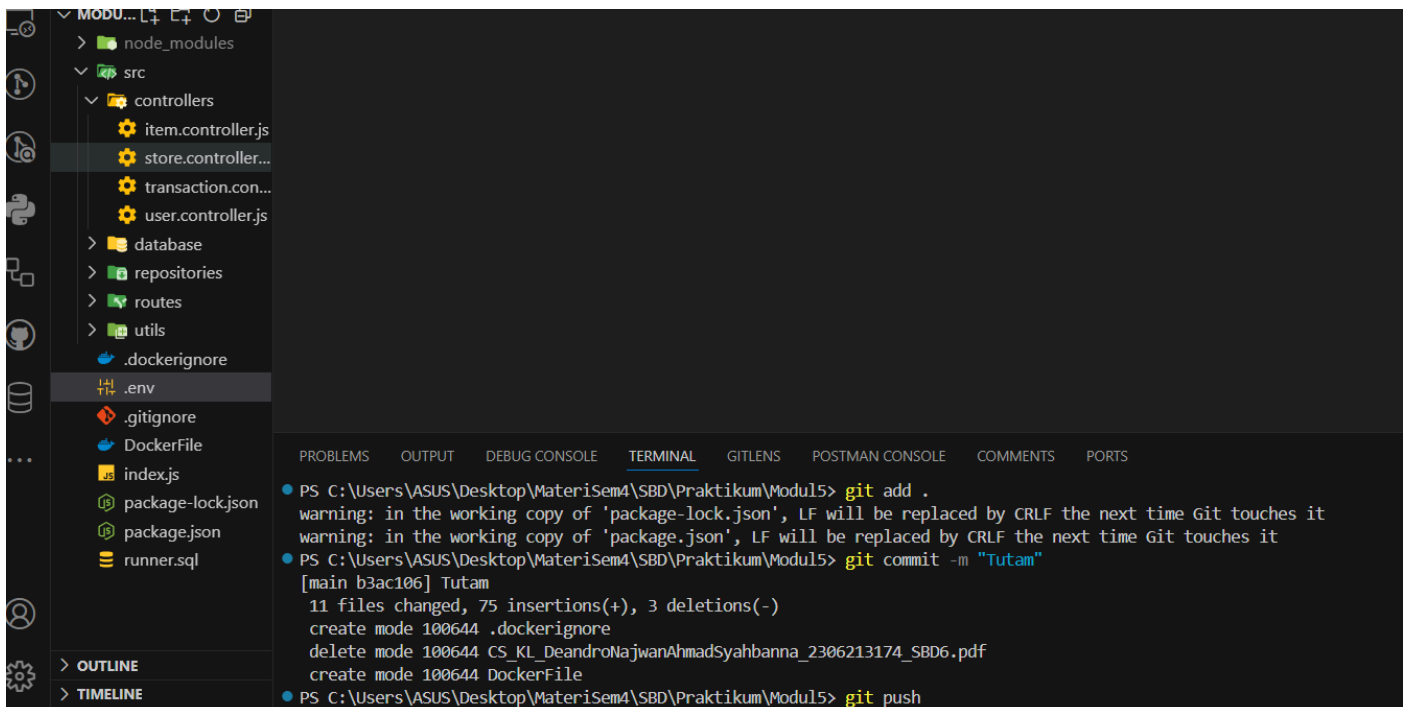
    if (!updatedUser) {
      return baseResponse(res, false, 500, "Failed to update user balance", null);
    }
  } catch (error) {
    return baseResponse(res, false, 500, "Internal server error", null);
  }
}
```

Memastikan bahwa terdapat pengecekan pada transaction id, apakah memang benar ada atau hanya sekedar id bohongan saja

Stress Test

1. Deploy Backend Alibaba

- Tambah Docker File Lalu push github



```
PS C:\Users\ASUS\Desktop\MateriSem4\SBD\Praktikum\Modul5> git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\ASUS\Desktop\MateriSem4\SBD\Praktikum\Modul5> git commit -m "Tutam"
[main b3ac106] Tutam
11 files changed, 75 insertions(+), 3 deletions(-)
create mode 100644 .dockerignore
delete mode 100644 CS_KL_DeandroNajwanAhmadSyahbanna_2306213174_SBD6.pdf
create mode 100644 DockerFile
PS C:\Users\ASUS\Desktop\MateriSem4\SBD\Praktikum\Modul5> git push
```

```
=> => sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e 7.67kB / 7.67kB
=> => sha256:929b04d7c782f04f615cf785488fed452b6569f87c73ff666ad553a7554f0006 1.72kB / 1.72kB
=> => sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549 446B / 446B
=> => extracting sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870
=> => extracting sha256:dd71dde834b5c203d162902e6b8994cb2309ae049a0eabc4efea161b2b5a3d0e
=> => extracting sha256:1e5a4c89cee5c0826c540ab06d4b6b491c96eda01837f430bd47f0d26702d6e3
=> => extracting sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549
=> [internal] load build context
=> => transferring context: 115.68kB
=> [2/5] WORKDIR /app
=> [3/5] COPY package*.json ./
=> [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:73f254cc1d18042e803690017b23c02bbcf1bd93758ecc1d32891db7eb3fd8c5
```

- Tambah Token Github

Github token

.....

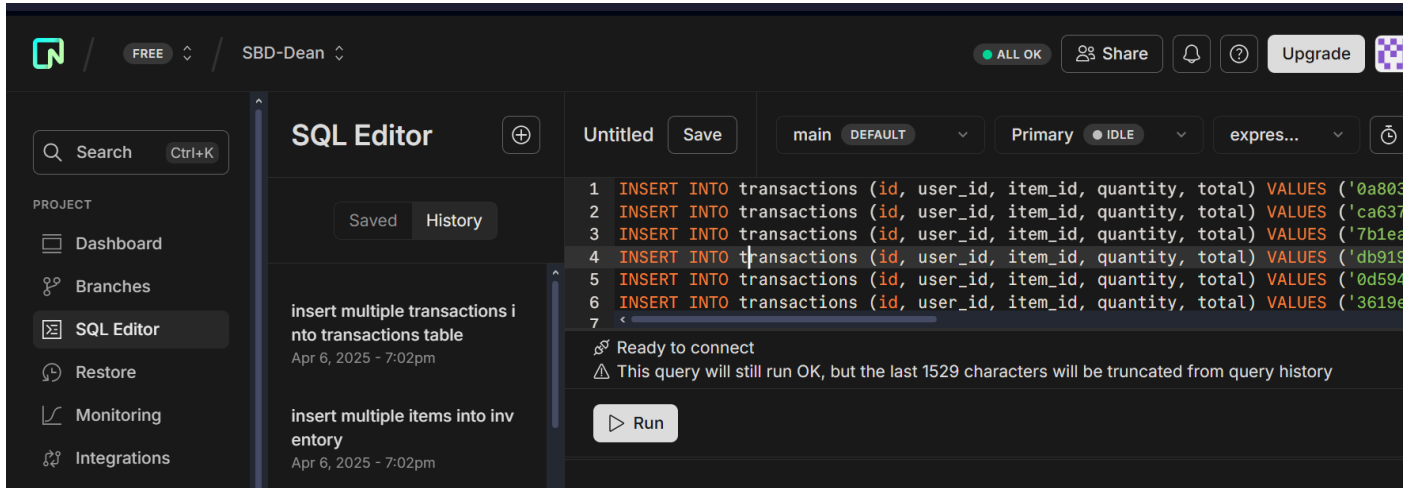
You can create a personal access token [here](#). In the scopes section you should ch

Save

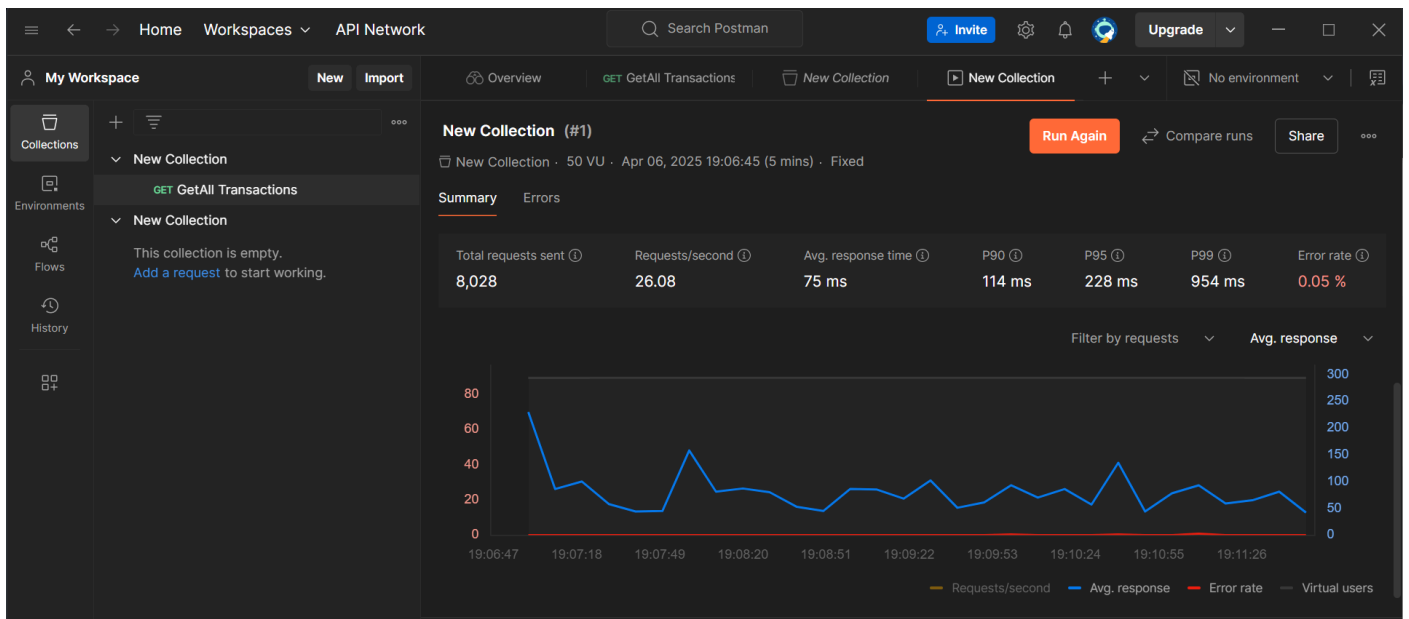
- **Buat Service App dan Set konfigurasi**

A screenshot of the SBD (Service Based Design) interface. On the left, a sidebar shows a tree view with 'sbd' expanded, revealing a 'SERVICES' section containing 'express_deandro'. The main area displays the details for 'sbd / express_deandro', which is marked as an 'APP'. Below this, a 'Source' section is visible, showing a code editor with a snippet of Java code.

2. Masukkan Data Dummy



3. Lakukan Stress test dengan error rate < 10%



4. Analisis Stress Test

Keberhasilan Stress Test diakibatkan oleh beberapa hal dimana diantaranya, keberhasilan dalam mengkonfigurasi easy panel yaitu dengan memasukkan docker file ke dalam repo github praktikum saat ini yang mempermudah proses test ini dapat berjalan dengan lancar. Selain itu, ada faktor lain juga dimana terdapat penambahan data yang tepat pada database dan juga query request GET transaction yang sesuai juga membuat Stress Test ini dapat berjalan dengan lancar dan memiliki tingkat error rate hanya sebesar 0,05%.