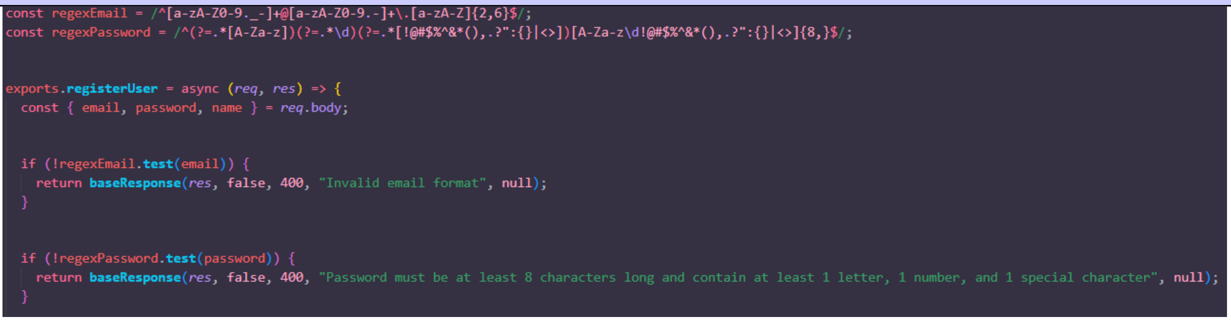
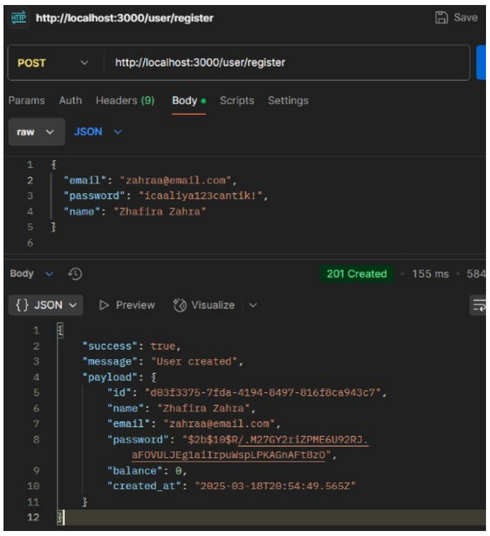
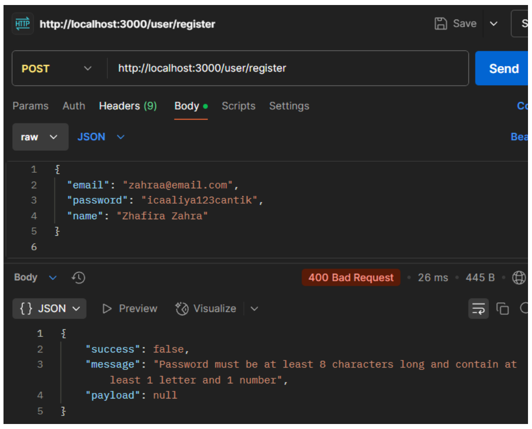
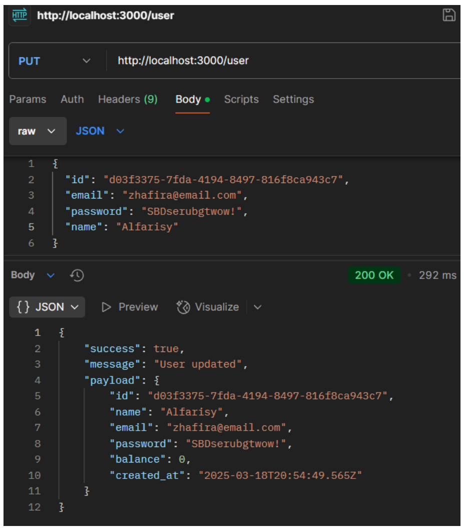
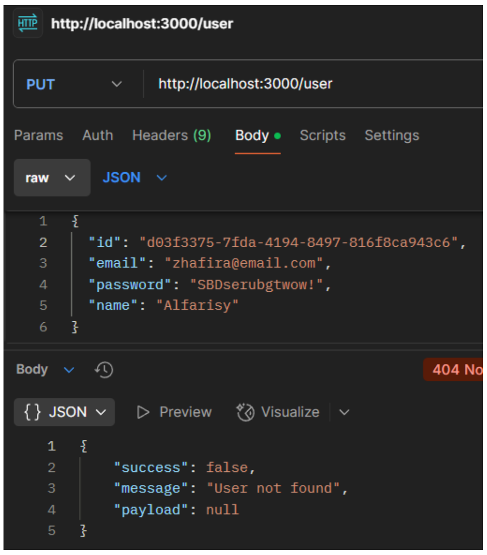
PRAKTIKUM
SISTEM BASIS DATA

Nama	Zhafira Zahra Alfarisy	**No. Modul**	6
NPM	2306250636	**Tipe**	Case Study

1. **Implementasikan regex di (POST) /user/register dan (PUT) /user dengan yang telah anda buat di TP, screenshot bagian kode yang mengimplementasikannya, serta body dan response yang gagal dikarenakan tidak memenuhi ketentuan regex.**

Kode
Screenshot


```
const regexEmail = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
const regexPassword = /^(?=.*[A-Za-z])(?=.*\d)(?=.*[!@#$%^&*(),.?":{}|<>])[A-Za-z\d!@#$%^&*(),.?":{}|<>]{8,}$/;


exports.registerUser = async (req, res) => {
  const { email, password, name } = req.body;


  if (!regexEmail.test(email)) {
    return baseResponse(res, false, 400, "Invalid email format", null);
  }


  if (!regexPassword.test(password)) {
    return baseResponse(res, false, 400, "Password must be at least 8 characters long and contain at least 1 letter, 1 number, and 1 special character", null);
  }
```

Body and Response		
Method	**Screenshot**	
	Success	**Failed**
POST		
	Success	**Failed**
PUT		

POST – Success:

```
http://localhost:3000/user/register
POST http://localhost:3000/user/register
Params  Auth  Headers (9)  Body  Scripts  Settings
raw  JSON
{
  "email": "zahraa@email.com",
  "password": "icaaliya123cantik!",
  "name": "Zhafira Zahra"
}

Body                                   201 Created · 155 ms · 584
{ } JSON   Preview   Visualize
{
    "success": true,
    "message": "User created",
    "payload": {
        "id": "d03f3375-7fda-4194-8497-816f8ca943c7",
        "name": "Zhafira Zahra",
        "email": "zahraa@email.com",
        "password": "$2b$10$R/.M27GY2riZPME6U92RJ.
            aFOVULJEg1aiIrpuWspLPKAGnAFt8zO",
        "balance": 0,
        "created_at": "2025-03-18T20:54:49.565Z"
    }
}
```

POST – Failed:

```
http://localhost:3000/user/register                    Save
POST http://localhost:3000/user/register               Send
Params  Auth  Headers (9)  Body  Scripts  Settings
raw  JSON
{
  "email": "zahraa@email.com",
  "password": "icaaliya123cantik",
  "name": "Zhafira Zahra"
}

Body                          400 Bad Request · 26 ms · 445 B
{ } JSON   Preview   Visualize
{
    "success": false,
    "message": "Password must be at least 8 characters long and contain at
        least 1 letter and 1 number",
    "payload": null
}
```

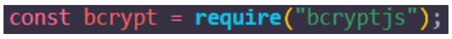
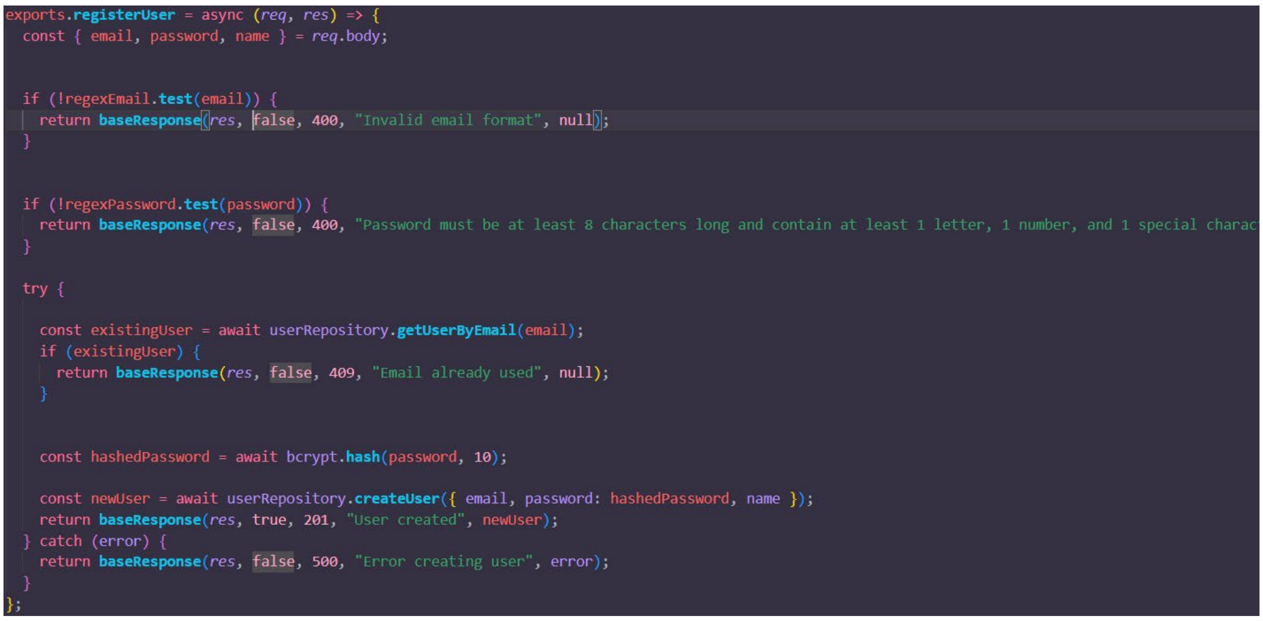
PUT – Success:

```
http://localhost:3000/user
PUT http://localhost:3000/user
Params  Auth  Headers (9)  Body  Scripts  Settings
raw  JSON
{
  "id": "d03f3375-7fda-4194-8497-816f8ca943c7",
  "email": "zhafira@email.com",
  "password": "SBDserubgtwow!",
  "name": "Alfarisy"
}
Body                                   200 OK · 292 ms
{ } JSON   Preview   Visualize
{
    "success": true,
    "message": "User updated",
    "payload": {
        "id": "d03f3375-7fda-4194-8497-816f8ca943c7",
        "name": "Alfarisy",
        "email": "zhafira@email.com",
        "password": "SBDserubgtwow!",
        "balance": 0,
        "created_at": "2025-03-18T20:54:49.565Z"
    }
}
```

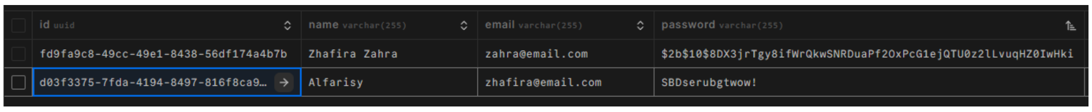
PUT – Failed:

```
http://localhost:3000/user
PUT http://localhost:3000/user
Params  Auth  Headers (9)  Body  Scripts  Settings
raw  JSON
{
  "id": "d03f3375-7fda-4194-8497-816f8ca943c6",
  "email": "zhafira@email.com",
  "password": "SBDserubgtwow!",
  "name": "Alfarisy"
}
Body                                   404 No
{ } JSON   Preview   Visualize
{
    "success": false,
    "message": "User not found",
    "payload": null
}
```

2. **Hash password yang masuk di (POST) /user/register dan (PUT) /user menggunakan bcrypt lalu simpan di database, screenshot bagian kode implementasinya serta hasil hash di database**

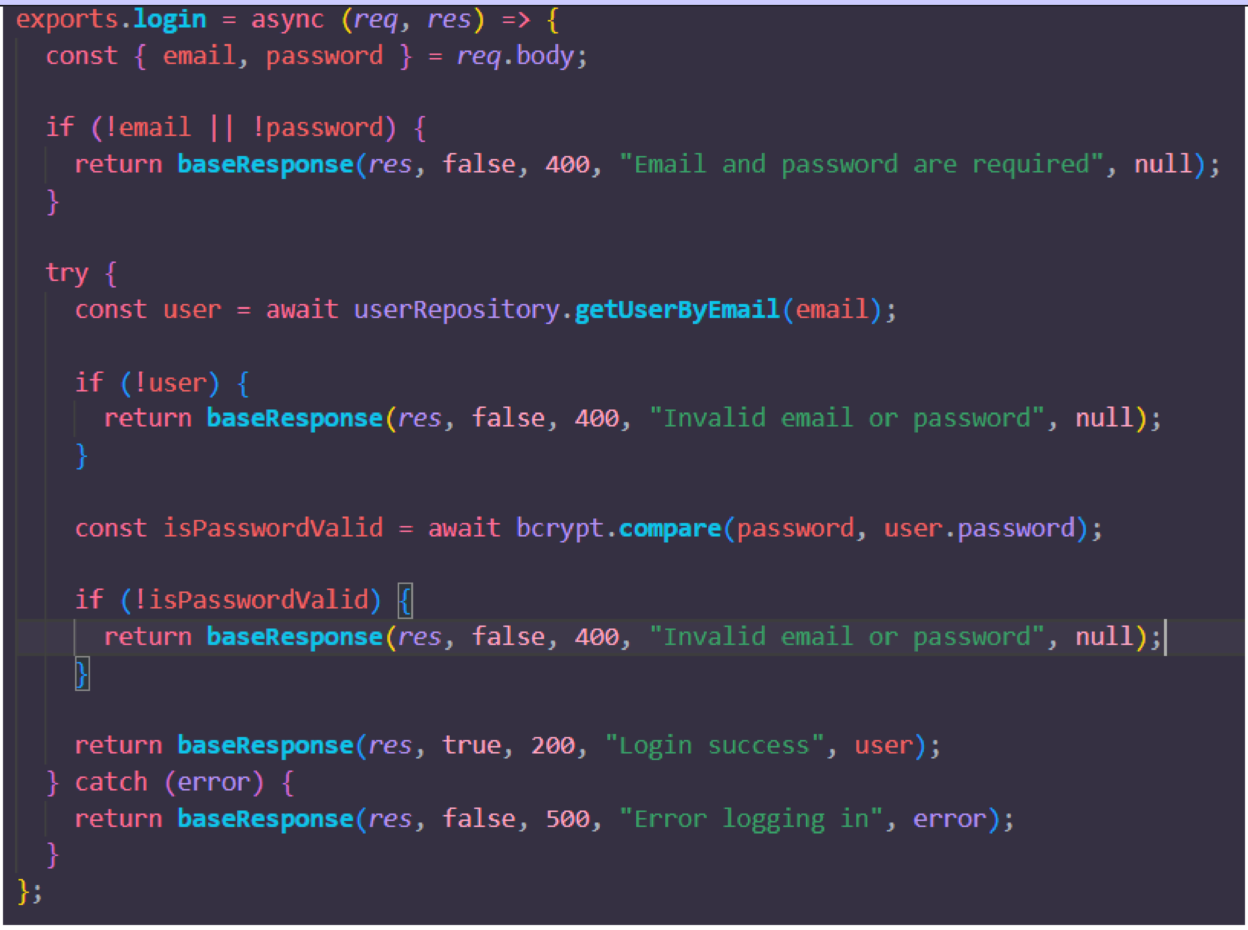
Kode
Screenshot
 `const bcrypt = require("bcryptjs");`  (code below)

```
exports.registerUser = async (req, res) => {
  const { email, password, name } = req.body;


  if (!regexEmail.test(email)) {
    return baseResponse(res, false, 400, "Invalid email format", null);
  }


  if (!regexPassword.test(password)) {
    return baseResponse(res, false, 400, "Password must be at least 8 characters long and contain at least 1 letter, 1 number, and 1 special charac
  }

  try {

    const existingUser = await userRepository.getUserByEmail(email);
    if (existingUser) {
      return baseResponse(res, false, 409, "Email already used", null);
    }


    const hashedPassword = await bcrypt.hash(password, 10);

    const newUser = await userRepository.createUser({ email, password: hashedPassword, name });
    return baseResponse(res, true, 201, "User created", newUser);
  } catch (error) {
    return baseResponse(res, false, 500, "Error creating user", error);
  }
};
```

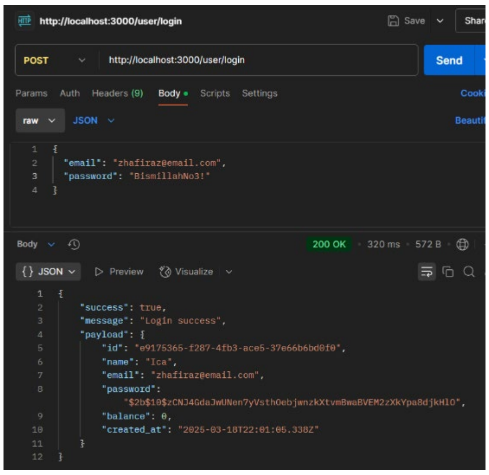
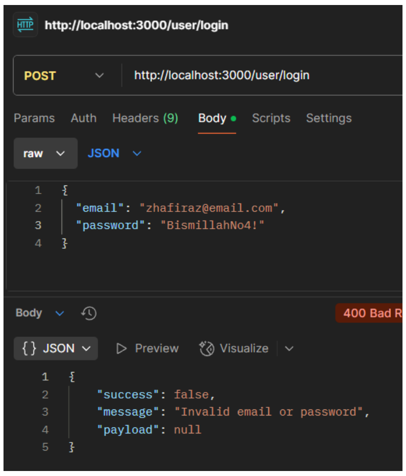
Hasil Hash di Database	
Method	Screenshot
Hasil Hash	

id uuid	name varchar(255)	email varchar(255)	password varchar(255)
fd9fa9c8-49cc-49e1-8438-56df174a4b7b	Zhafira Zahra	zahra@email.com	$2b$10$8DX3jrTgy8ifWrQkwSNRDuaPf2OxPcG1ejQTU0z2lLvuqHZ0IwHki
d03f3375-7fda-4194-8497-816f8ca9…	Alfarisy	zhafira@email.com	SBDserubgtwow!

3. **Compare password hash di database dengan yang di request pada endpoint (POST) /user/login, screenshot bagian kode yang mengimplementasikannya**

Kode
Screenshot


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

  if (!email || !password) {
    return baseResponse(res, false, 400, "Email and password are required", null);
  }

  try {
    const user = await userRepository.getUserByEmail(email);

    if (!user) {
      return baseResponse(res, false, 400, "Invalid email or password", null);
    }

    const isPasswordValid = await bcrypt.compare(password, user.password);

    if (!isPasswordValid) {
      return baseResponse(res, false, 400, "Invalid email or password", null);
    }

    return baseResponse(res, true, 200, "Login success", user);
  } catch (error) {
    return baseResponse(res, false, 500, "Error logging in", error);
  }
};
```

Body and Response		
Method	Screenshot	
POST	Success 	Failed 

Success:

POST http://localhost:3000/user/login — Body (raw, JSON)

```
{
  "email": "zhafiraz@email.com",
  "password": "BismillahNo3!"
}
```

200 OK · 320 ms · 572 B

```
{
    "success": true,
    "message": "Login success",
    "payload": {
        "id": "e9175365-f287-4fb3-ace5-37e66b6bd0f0",
        "name": "Ica",
        "email": "zhafiraz@email.com",
        "password":
            "$2b$10$zCNJ4GdaJwUNen7yVsthOebjwnzkXtvmBwaBVEM2zXkYpa8djkHlO",
        "balance": 0,
        "created_at": "2025-03-18T22:01:05.338Z"
    }
}
```

Failed:

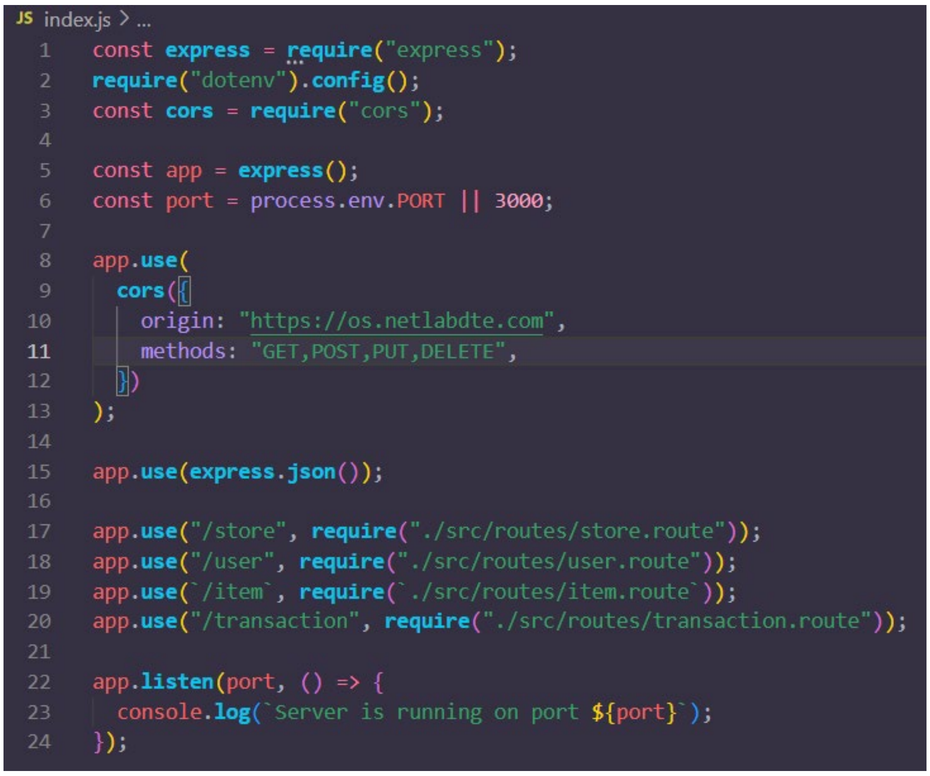
POST http://localhost:3000/user/login — Body (raw, JSON)

```
{
  "email": "zhafiraz@email.com",
  "password": "BismillahNo4!"
}
```

400 Bad R

```
{
    "success": false,
    "message": "Invalid email or password",
    "payload": null
}
```

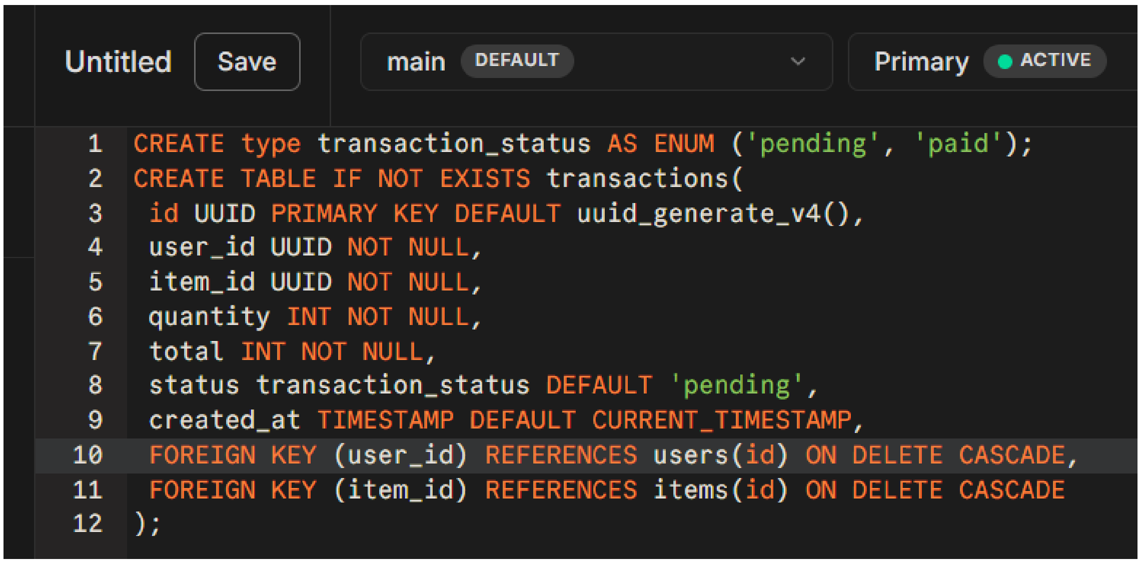
4. **Implementasikan middleware CORS seperti yang telah anda buat di TP.**

Screenshot


index.js > ...

```js
const express = require("express");
require("dotenv").config();
const cors = require("cors");

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

app.use(
  cors({
    origin: "https://os.netlabdte.com",
    methods: "GET,POST,PUT,DELETE",
  })
);

app.use(express.json());

app.use("/store", require("./src/routes/store.route"));
app.use("/user", require("./src/routes/user.route"));
app.use(`/item`, require(`./src/routes/item.route`));
app.use("/transaction", require("./src/routes/transaction.route"));

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

5. Jalankan Query pada Database

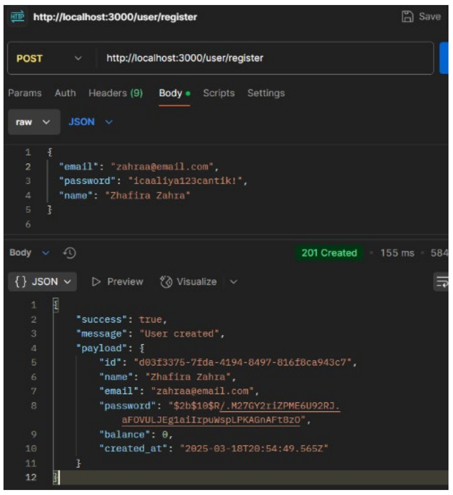
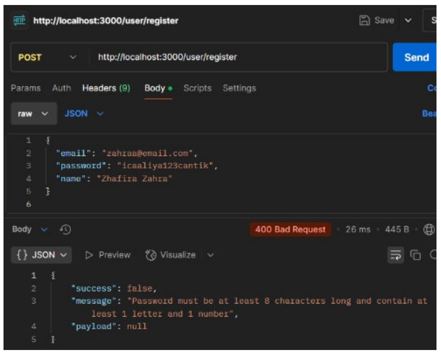
Screenshot


```
Untitled   Save      main DEFAULT      Primary ● ACTIVE
```

```sql
CREATE type transaction_status AS ENUM ('pending', 'paid');
CREATE TABLE IF NOT EXISTS transactions(
 id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
 user_id UUID NOT NULL,
 item_id UUID NOT NULL,
 quantity INT NOT NULL,
 total INT NOT NULL,
 status transaction_status DEFAULT 'pending',
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
 FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE
);
```

6. Lanjut Backend

7. Screenshot Response

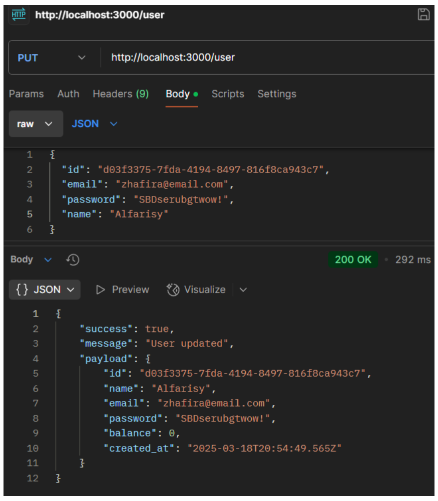
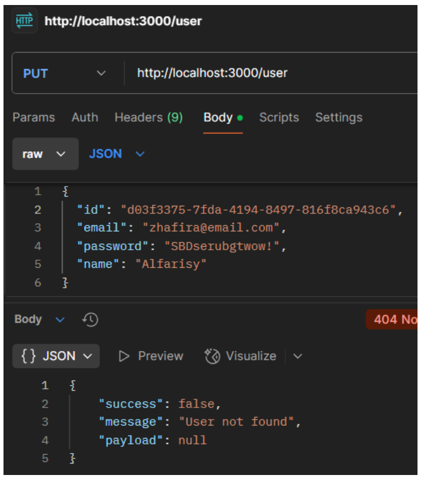
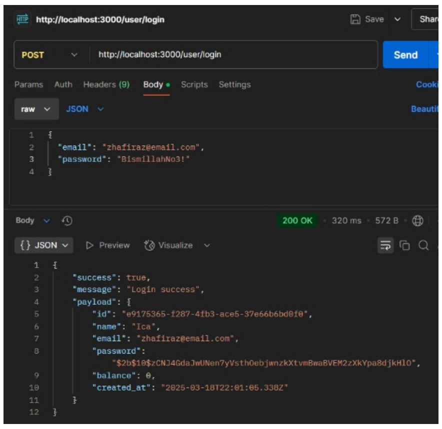
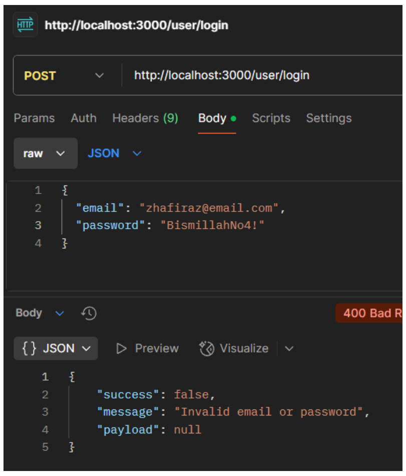
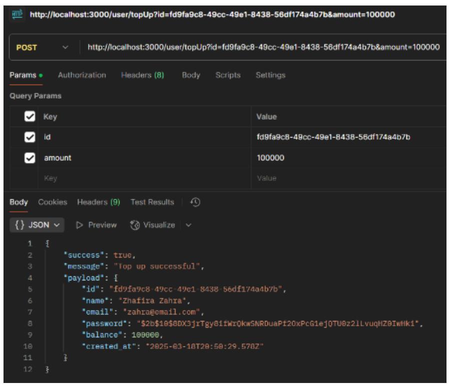
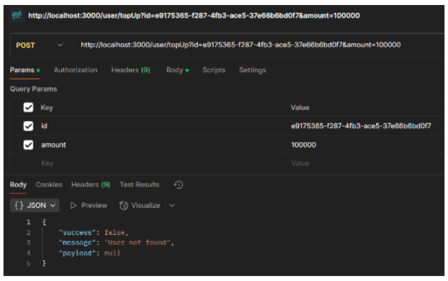
Method	Endpoint	Success	Failed
POST	/user/register		

Success:

```
http://localhost:3000/user/register
POST  http://localhost:3000/user/register
Params  Auth  Headers (9)  Body ●  Scripts  Settings
raw  JSON
{
  "email": "zahraa@email.com",
  "password": "icaaliya123cantik!",
  "name": "Zhafira Zahra"
}

Body   201 Created · 155 ms · 584
{} JSON   Preview   Visualize
{
    "success": true,
    "message": "User created",
    "payload": {
        "id": "d03f3375-7fda-4194-8497-816f8ca943c7",
        "name": "Zhafira Zahra",
        "email": "zahraa@email.com",
        "password": "$2b$10$R/.M27GY2riZPME6U92RJ.
            aFOVULJEg1aiIrpuWspLPKAGnAFt8zO",
        "balance": 0,
        "created_at": "2025-03-18T20:54:49.565Z"
    }
}
```

Failed:

```
http://localhost:3000/user/register
POST  http://localhost:3000/user/register   Send
Params  Auth  Headers (9)  Body ●  Scripts  Settings
raw  JSON
{
  "email": "zahraa@email.com",
  "password": "icaaliya123cantik",
  "name": "Zhafira Zahra"
}

Body   400 Bad Request · 26 ms · 445 B
{} JSON   Preview   Visualize
{
    "success": false,
    "message": "Password must be at least 8 characters long and contain at
        least 1 letter and 1 number",
    "payload": null
}
```

PUT	/user		
POST	/user/login		
POST	/user/topUp		

PUT http://localhost:3000/user — Params Auth Headers (9) Body Scripts Settings — raw JSON

```
{
  "id": "d03f3375-7fda-4194-8497-816f8ca943c7",
  "email": "zhafira@email.com",
  "password": "SBDserubgtwow!",
  "name": "Alfarisy"
}
```

Body 200 OK 292 ms

```
{
    "success": true,
    "message": "User updated",
    "payload": {
        "id": "d03f3375-7fda-4194-8497-816f8ca943c7",
        "name": "Alfarisy",
        "email": "zhafira@email.com",
        "password": "SBDserubgtwow!",
        "balance": 0,
        "created_at": "2025-03-18T20:54:49.565Z"
    }
}
```

PUT http://localhost:3000/user — Params Auth Headers (9) Body Scripts Settings — raw JSON

```
{
  "id": "d03f3375-7fda-4194-8497-816f8ca943c6",
  "email": "zhafira@email.com",
  "password": "SBDserubgtwow!",
  "name": "Alfarisy"
}
```

Body 404 No

```
{
    "success": false,
    "message": "User not found",
    "payload": null
}
```

POST http://localhost:3000/user/login — Save Share Send — Params Auth Headers (9) Body Scripts Settings Cooki — raw JSON Beauti

```
{
  "email": "zhafiraz@email.com",
  "password": "BismillahNo3!"
}
```

Body 200 OK 320 ms 572 B

```
{
    "success": true,
    "message": "Login success",
    "payload": {
        "id": "e9175365-f287-4fb3-ace5-37e66b6bd0f0",
        "name": "Ica",
        "email": "zhafiraz@email.com",
        "password":
            "$2b$10$zCNJ4GdaJwUNen7yVsthOebjwnzkXtvmBwaBVEM2zXkYpa8djkHlO",
        "balance": 0,
        "created_at": "2025-03-18T22:01:05.338Z"
    }
}
```

POST http://localhost:3000/user/login — Params Auth Headers (9) Body Scripts Settings — raw JSON

```
{
  "email": "zhafiraz@email.com",
  "password": "BismillahNo4!"
}
```

Body 400 Bad R

```
{
    "success": false,
    "message": "Invalid email or password",
    "payload": null
}
```

POST http://localhost:3000/user/topUp?id=fd9fa9c8-49cc-49e1-8438-56df174a4b7b&amount=100000 — Params Authorization Headers (8) Body Scripts Settings

Query Params

Key	Value
id	fd9fa9c8-49cc-49e1-8438-56df174a4b7b
amount	100000

Body Cookies Headers (9) Test Results — JSON

```
{
    "success": true,
    "message": "Top up successful",
    "payload": {
        "id": "fd9fa9c8-49cc-49e1-8438-56df174a4b7b",
        "name": "Zhafira Zahra",
        "email": "zahra@email.com",
        "password": "$2b$10$8DX3jrTgy8ifWrQkwSNRDuaPf2OxPcG1ejQTU0z2lLvuqHZ0IwHk1",
        "balance": 100000,
        "created_at": "2025-03-18T20:50:29.578Z"
    }
}
```

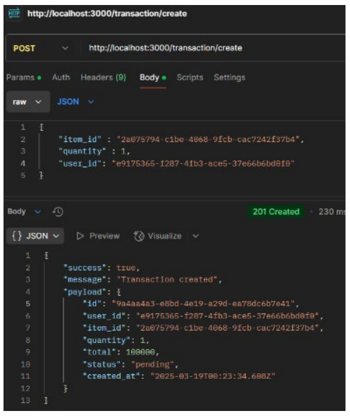
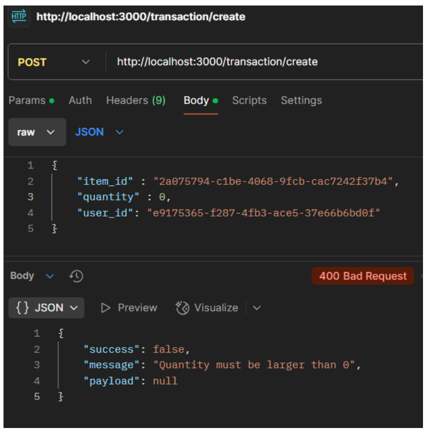
POST http://localhost:3000/user/topUp?id=e9175365-f287-4fb3-ace5-37e66b6bd0f7&amount=100000 — Params Authorization Headers (9) Body Scripts Settings

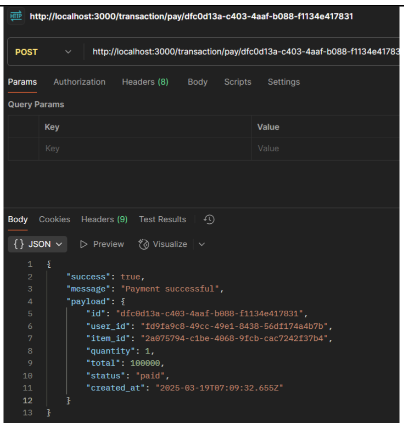
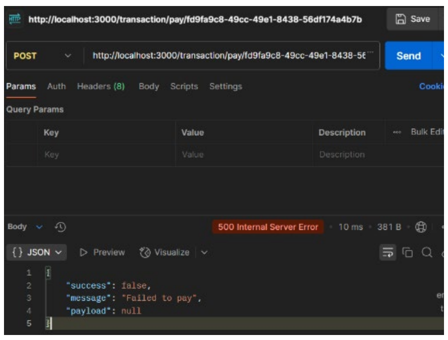
Query Params

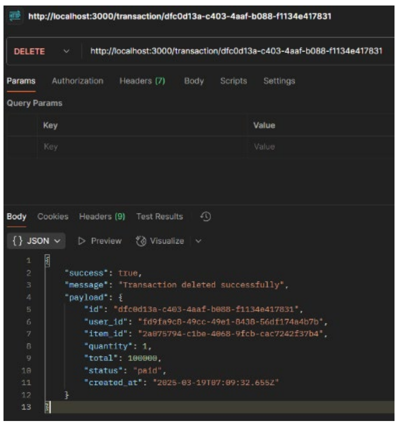
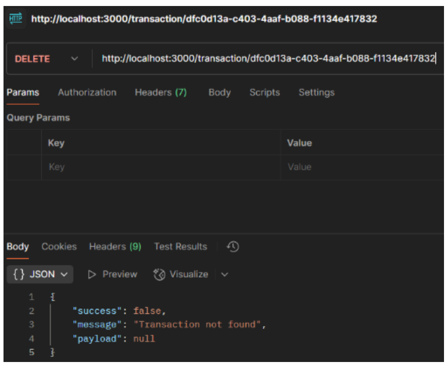
Key	Value
id	e9175365-f287-4fb3-ace5-37e66b6bd0f7
amount	100000

Body Cookies Headers (9) Test Results — JSON

```
{
    "success": false,
    "message": "User not found",
    "payload": null
}
```

POST	/transaction/create	 http://localhost:3000/transaction/create
		POST http://localhost:3000/transaction/create
		Params Auth Headers (9) Body Scripts Settings
		raw JSON
		`{`
		`"item_id" : "2a075794-c1be-4068-9fcb-cac7242f37b4",`
		`"quantity" : 1,`
		`"user_id": "e9175365-f287-4fb3-ace5-37e66b6bd0f0"`
		`}`
		Body 201 Created 230 ms
		`{`
		`"success": true,`
		`"message": "Transaction created",`
		`"payload": {`
		`"id": "9a4aa4a3-e8bd-4e19-a29d-ea78dc6b7e41",`
		`"user_id": "e9175365-f287-4fb3-ace5-37e66b6bd0f0",`
		`"item_id": "2a075794-c1be-4068-9fcb-cac7242f37b4",`
		`"quantity": 1,`
		`"total": 100000,`
		`"status": "pending",`
		`"created_at": "2025-03-19T00:23:34.608Z"`
		`}`
		`}`
			 http://localhost:3000/transaction/create
			POST http://localhost:3000/transaction/create
			Params Auth Headers (9) Body Scripts Settings
			raw JSON
			`{`
			`"item_id" : "2a075794-c1be-4068-9fcb-cac7242f37b4",`
			`"quantity" : 0,`
			`"user_id": "e9175365-f287-4fb3-ace5-37e66b6bd0f"`
			`}`
			Body 400 Bad Request
			`{`
			`"success": false,`
			`"message": "Quantity must be larger than 0",`
			`"payload": null`
			`}`

POST	/transaction/pay	 http://localhost:3000/transaction/pay/dfc0d13a-c403-4aaf-b088-f1134e417831
		POST http://localhost:3000/transaction/pay/dfc0d13a-c403-4aaf-b088-f1134e41783
		Params Authorization Headers (8) Body Scripts Settings
		Query Params
		Key Value
		Body Cookies Headers (9) Test Results
		`{`
		`"success": true,`
		`"message": "Payment successful",`
		`"payload": {`
		`"id": "dfc0d13a-c403-4aaf-b088-f1134e417831",`
		`"user_id": "fd9fa9c8-49cc-49e1-8438-56df174a4b7b",`
		`"item_id": "2a075794-c1be-4068-9fcb-cac7242f37b4",`
		`"quantity": 1,`
		`"total": 100000,`
		`"status": "paid",`
		`"created_at": "2025-03-19T07:09:32.655Z"`
		`}`
		`}`
			 http://localhost:3000/transaction/pay/fd9fa9c8-49cc-49e1-8438-56df174a4b7b
			POST http://localhost:3000/transaction/pay/fd9fa9c8-49cc-49e1-8438-5€
			Params Auth Headers (8) Body Scripts Settings
			Query Params
			Key Value Description
			Body 500 Internal Server Error 10 ms 381 B
			`{`
			`"success": false,`
			`"message": "Failed to pay",`
			`"payload": null`
			`}`

DELETE	/transaction/:id	 http://localhost:3000/transaction/dfc0d13a-c403-4aaf-b088-f1134e417831
		DELETE http://localhost:3000/transaction/dfc0d13a-c403-4aaf-b088-f1134e417831
		Params Authorization Headers (7) Body Scripts Settings
		Query Params
		Key Value
		Body Cookies Headers (9) Test Results
		`{`
		`"success": true,`
		`"message": "Transaction deleted successfully",`
		`"payload": {`
		`"id": "dfc0d13a-c403-4aaf-b088-f1134e417831",`
		`"user_id": "fd9fa9c8-49cc-49e1-8438-56df174a4b7b",`
		`"item_id": "2a075794-c1be-4068-9fcb-cac7242f37b4",`
		`"quantity": 1,`
		`"total": 100000,`
		`"status": "paid",`
		`"created_at": "2025-03-19T07:09:32.655Z"`
		`}`
		`}`
			 http://localhost:3000/transaction/dfc0d13a-c403-4aaf-b088-f1134e417832
			DELETE http://localhost:3000/transaction/dfc0d13a-c403-4aaf-b088-f1134e417832
			Params Authorization Headers (7) Body Scripts Settings
			Query Params
			Key Value
			Body Cookies Headers (9) Test Results
			`{`
			`"success": false,`
			`"message": "Transaction not found",`
			`"payload": null`
			`}`