

PRAKTIKUM SISTEM BASIS DATA

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

1. Implementasi Regex pada emailnya dan juga hash passwordnya
 - User Register

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

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

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

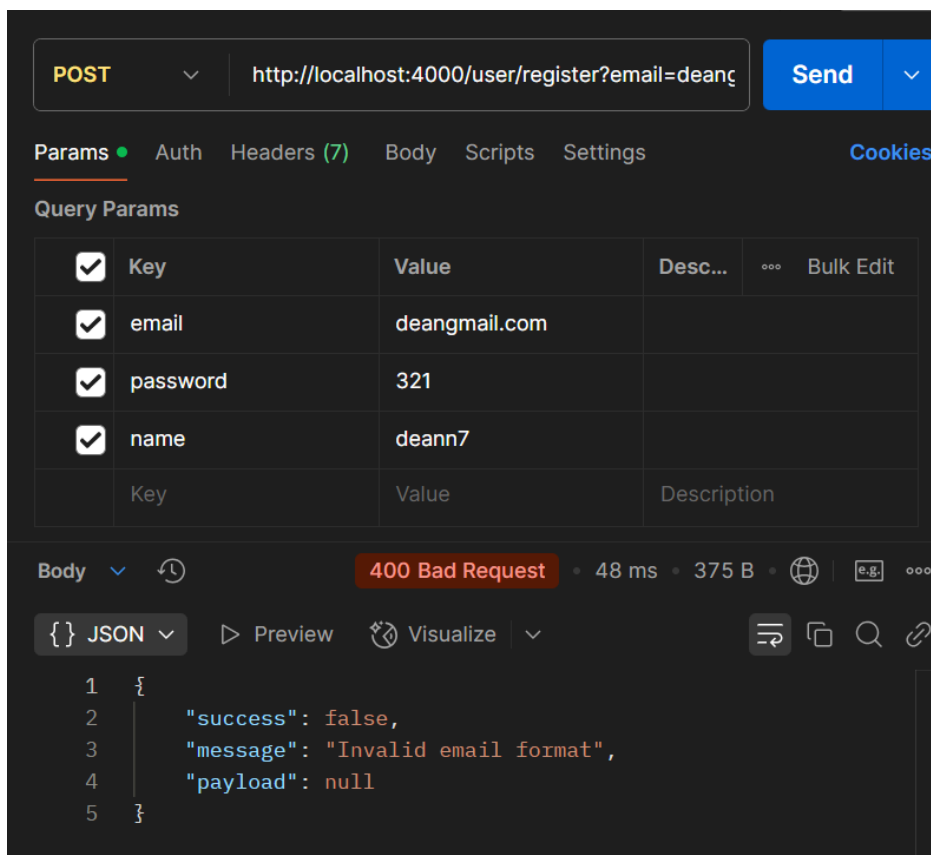
  if (!emailRegex.test(email)) {
    return baseResponse(res, false, 400, "Invalid email format", null);
  }

  if (!name || !email || !password) {
    return baseResponse(res, false, 400, "Missing user information", null);
  }
  try {
    const hashPassword = await bcrypt.hash(password, 10);
    const 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);
  }
};
```

- User Update

```
exports.updateUser = async (req, res) => {
  const { id, name, email, password, balance } = req.body;
  if (!id){
    return baseResponse(res, false, 400, "Missing user ID", null);
  }
  if (!emailRegex.test(email)) {
    return baseResponse(res, false, 400, "Invalid email format", null);
  }
  if (!name || !email || !password || balance === undefined) {
    return baseResponse(res, false, 400, "Missing user information", null);
  }
  try {
    const hashPassword = await bcrypt.hash(password, 10);
    const user = await userRepository.updateUser(id, { name, email, password: hashPassword, balance });
    if (!user) {
      return baseResponse(res, false, 404, "User not found", null);
    }
    baseResponse(res, true, 200, "User updated successfully", user);
  } catch (error) {
    baseResponse(res, false, 500, "An error occurred while updating user", null);
  }
};
```

Hasil Screenshot apabila tidak memenuhi emailRegex:



The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:4000/user/register?email=deang
- Send Button:** Send
- Params:** Auth, Headers (7), Body, Scripts, Settings, Cookies
- Query Params:** A table with 5 columns: Key, Value, Desc..., Bulk Edit, and an additional column. The data is as follows:

Key	Value	Desc...	Bulk Edit
email	deangmail.com		
password	321		
name	deann7		

The response status is **400 Bad Request** with a duration of 48 ms and a size of 375 B. The response body is in JSON format:

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

Hasil Hash Di database

email varchar(255)	password varchar(255)	balance
dean@gmail.com	\$2b\$10\$csxJ3oLLEbDG/OkNqVg1f.6c7j8L...	0
abcd@gmail.con	123	0

2. Compare password hash di database dengan yang di request pada endpoint

```

exports.userLogin = async (req, res) => {
  const { email, password } = req.query;

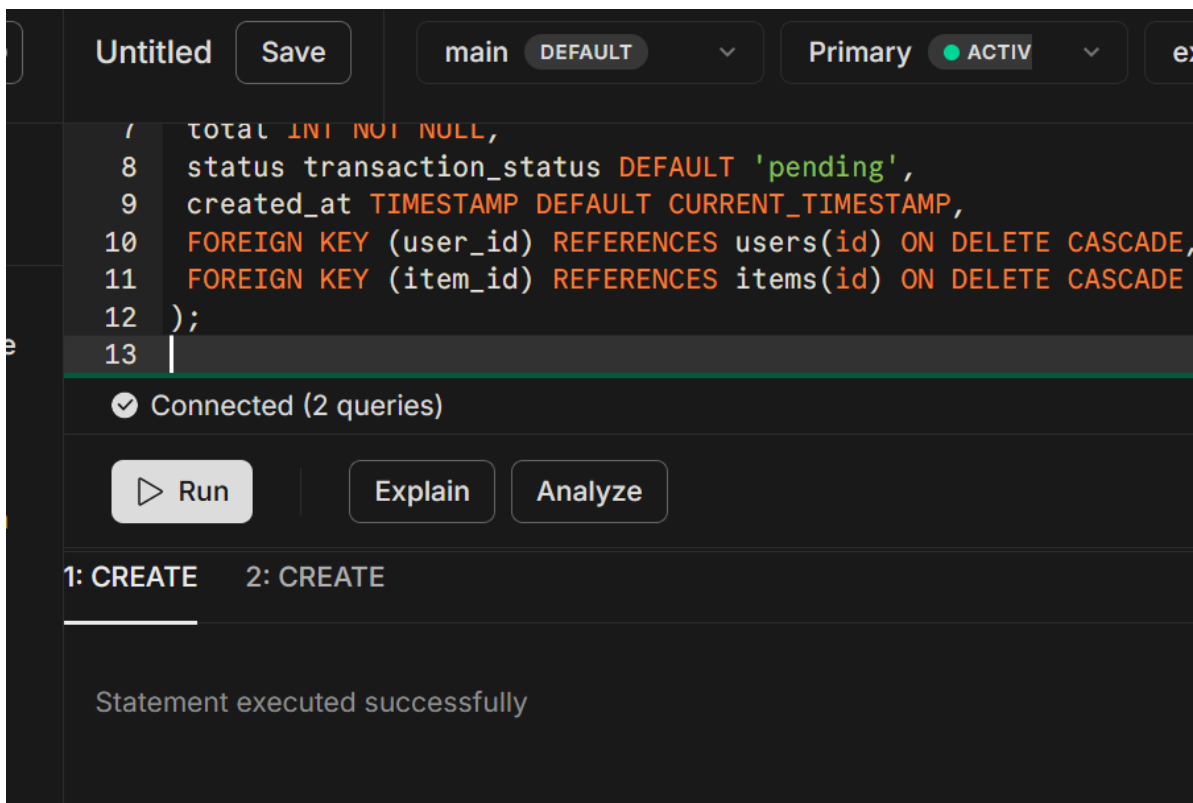
  if (!email || !password) {
    return baseResponse(res, false, 400, "Missing email or password", null);
  }
  try {
    const user = await userRepository.userLogin(email, password);
    if (!user) {
      return baseResponse(res, false, 404, "User not found", null);
    }
    const hashPasswordMatched = await bcrypt.compare(password, user.password);
    if (!hashPasswordMatched) {
      return baseResponse(res, false, 401, "Invalid password", null);
    }
    baseResponse(res, true, 200, "Login success", user);
  } catch (error) {
    baseResponse(res, false, 500, "An error occurred while logging in", null);
  }
};

```

3. Implementation Cors

```
const corsOptions = {
  origin: function (origin, callback) {
    if (!origin) return callback(null, true);
    try {
      const parsedOrigin = new URL(origin);
      if (parsedOrigin.hostname === 'os.netlabdte.com') {
        callback(null, true);
      } else {
        callback(new Error('Blocked by CORS: Domain tidak diizinkan'));
      }
    } catch (err) {
      callback(new Error('Invalid origin'));
    }
  },
  methods: ['GET', 'POST', 'PUT', 'DELETE']
};
app.use(cors(corsOptions));
```

4. Menjalankan Query

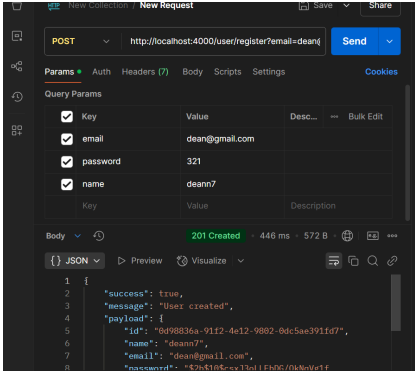
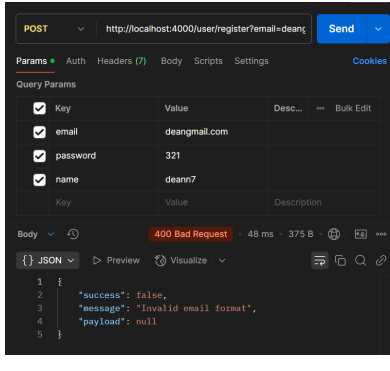
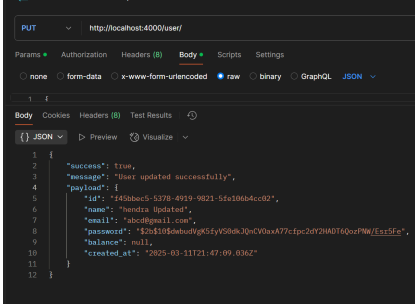
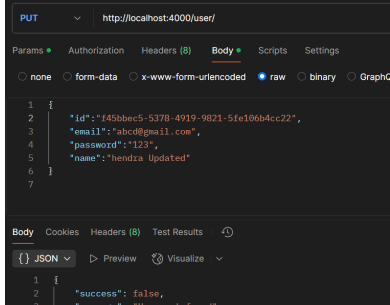
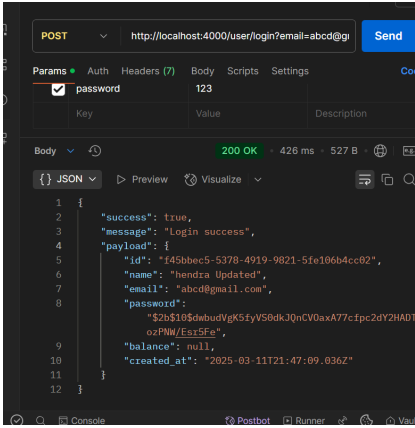
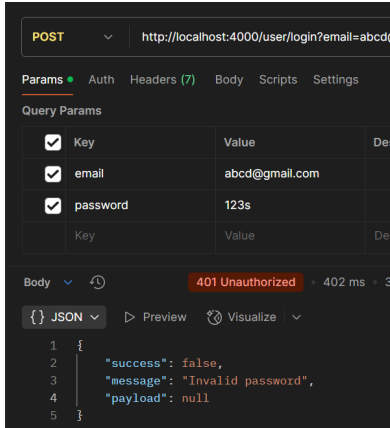


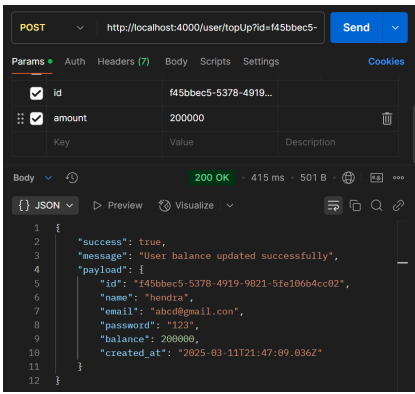
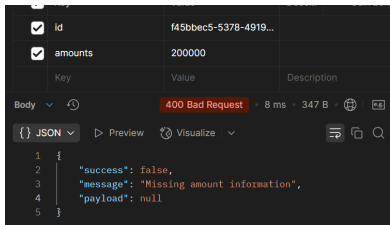
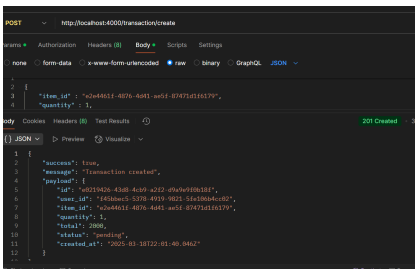
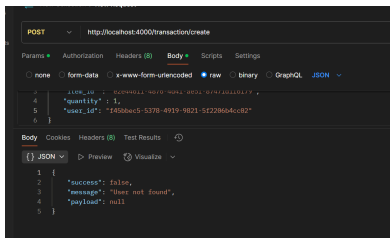

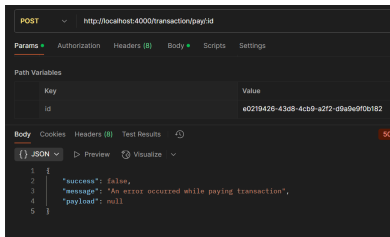
The screenshot shows a database query editor interface. At the top, there are tabs for 'Untitled', 'Save', 'main', 'DEFAULT', 'Primary', and 'ACTIV'. The main area displays a SQL query:

```
7 total INT NOT NULL,  
8 status transaction_status DEFAULT 'pending',  
9 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
10 FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,  
11 FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE  
12 );  
13
```

Below the query, there is a status bar indicating 'Connected (2 queries)'. There are buttons for 'Run', 'Explain', and 'Analyze'. The 'Run' button is highlighted, and the status bar shows '1: CREATE 2: CREATE'. The bottom of the interface displays the message 'Statement executed successfully'.

5. Screenshot Response

method	endpoint	success	false
POST	/user/register		
PUT	/user		
POST	/user/login		

POST	/user/topUp		
POST	/transaction/create		
POST	/transaction/pay		
DELETE	/transaction/:id	