NETWORK LABORATORY
Electrical Engineering Department, 2nd floor
Universitas Indonesia
Depok. 16424

Nama : Gutri Rahmad Zuwa
NIM : 320200401009

PRAKTIKUM WEB DEVELOPMENT
Minggu ke-9, Sesi ke-1
Case Study

PRE CS

1. Resource
2. Npm i

```
PS C:\Users\gutri\Documents\KULIAH\Praktikum UI\Pemrograman Web\CS\Modul17_Gutri_Rahmad_Zuwa> npm i

added 148 packages, and audited 149 packages in 6s

10 packages are looking for funding
  run `npm fund` for details
```

3. Create table

```
modul_17=# CREATE TABLE unhan_modul_17(
modul_17(# id SERIAL primary key not null,
modul_17(# username text not null,
modul_17(# email text unique not null,
modul_17(# password text unique not null);
CREATE TABLE
```

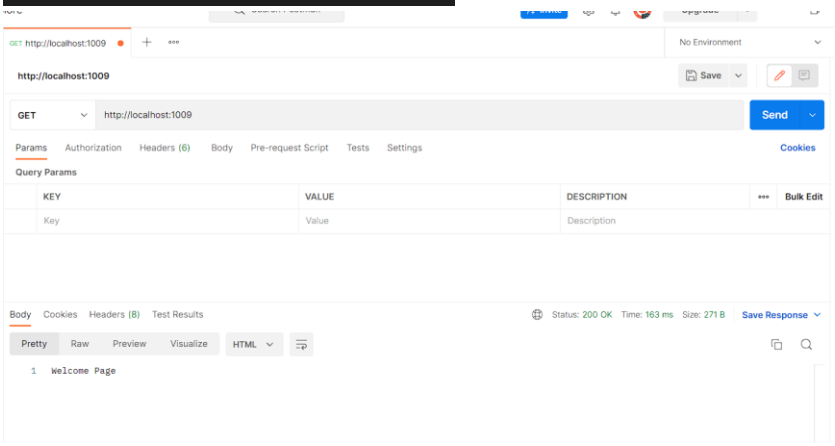
4. Db.config.js

```
nfig > JS db.config.js > [e] db
const { Client } = require('pg');

const db = new Client({
    user: '',
    host: '',
    database: 'modul_17',
    password: 'gutri',
    port: '5432',
});

module.exports = db;
```

5. .env

```
PORT=1009

DB_USER='postgres'
DB_HOST='localhost'
DB_NAME='modul_17'
DB_PASS='jnkmujiklop'
DB_PORT=5432
```

```
nfig > JS db.config.js > DB_PORT
const { Client } = require('pg');
require('dotenv').config();
const {
    DB_PORT,
    DB_HOST,
    DB_USER,
    DB_PASS,
    DB_NAME
} = process.env;

const db = new Client({
    user: DB_USER,
    host: DB_HOST,
    database: DB_NAME,
    password: DB_PASS,
    port: DB_PORT,
});
```



6. SECRET

```
SECRET=secret
```

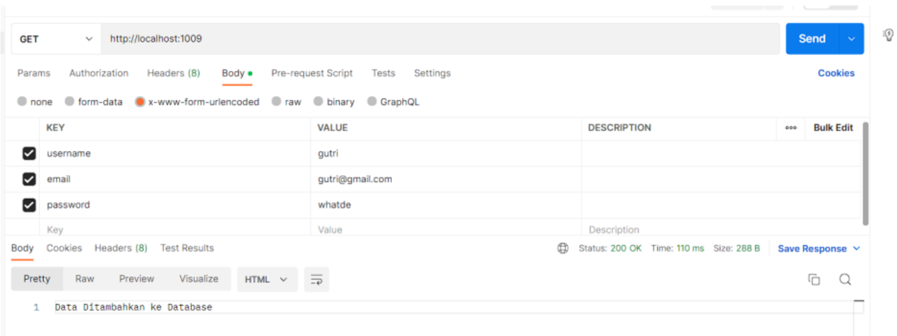
PART 1

7. Hash

```
const register = async(req, res, next) => {
    // * 7. silahkan ubah password yang telah
    diterima menjadi dalam bentuk hashing
    const {username, email, password} = req.body;
    try{
    const hash = await bcrypt.hash(password,10);
```

8. Register

```
database

    await db.query(`INSERT INTO unhan_modul_17
    VALUES(DEFAULT, $1, $2, $3)`, [username,
    email,hash])
    res.send("Data Ditambahkan ke Database")
} catch (err){
    res.send('Inputan Tidak Valid')
}
```




```
modul_17=# SELECT * from unhan_modul_17;
 id | username |       email       |                              password
----+----------+-------------------+------------------------------------------
--------------------
  1 | gutri    | gutri@gmail.com | $2b$10$Ljm0Dq58ukDM3vdd8inv1.vTrqaVTOb1hR
TmFUVZEQn1X64To6ep2
(1 row)
```

Part 2

9. Bycrypt compare

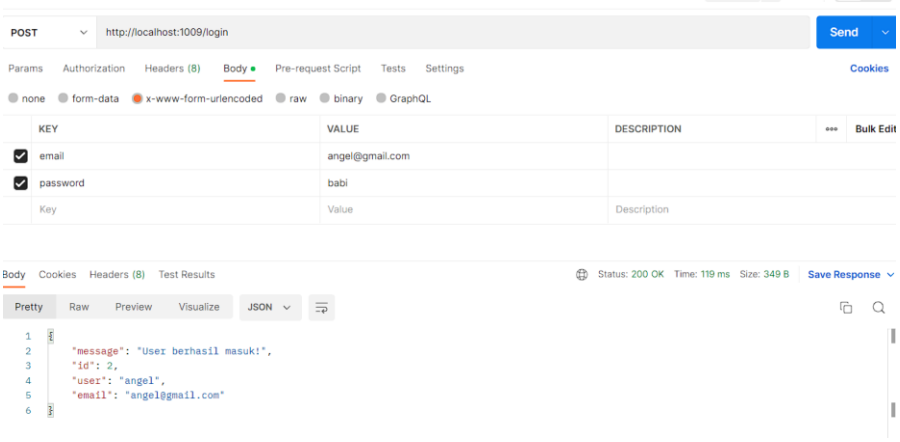
```
const login = async(req, res, next) => {

  // 9. komparasi antara password yang diinput oleh pengguna dan password yang ada didatabase
  const { email, password } = req.body;
  try {
  const data = await db.query(`SELECT * FROM unhan_modul_17 WHERE email= $1;`, [email]) //Verifying if the user exists in the database
  const user = data.rows;
  if (user.length === 0) {
  res.status(400).json({
  error: "User tidak ada, Silahkan Daftar dulu!",
  });
  }
  else {
      bcrypt.compare(password, user[0].password, (err, result) => { //Comparing the hashed password
      if (err) {
      res.status(500).json({
      error: "Server error",
      });
      } else if (result === true) { //Checking if credentials match
```

10. Jwt sign

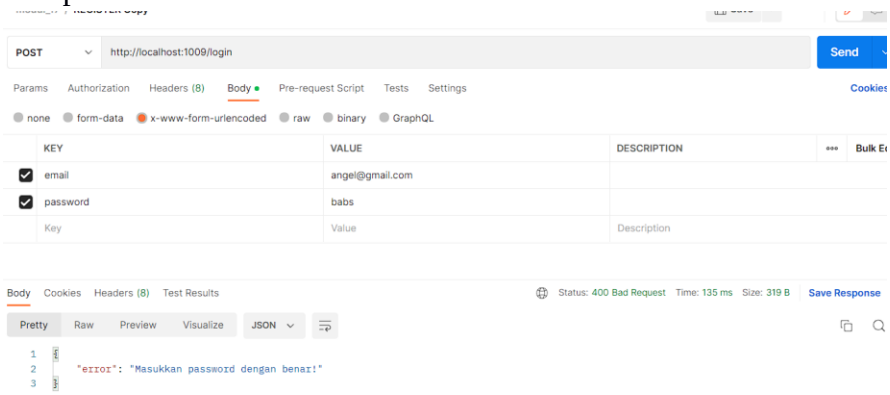
```
} else if (result === true) { //Checking if
// 10. Generate token menggunakan jwt sign
const token = jwt.sign(
{
id: user[0].id,
username: user[0].username,
email: user[0].email,
password:user[0].password,
},
process.env.SECRET
);
res.status(200).json({
message: "User berhasil masuk!",
token: token,
```

11. Menampilkan id, username, email

```
},
process.env.SECRET
);
res.cookie("JWT",token, {httpOnly: true,
sameSite:"strict",})
res.status(200).json({
message: "User berhasil masuk!",
//11. kembalikan nilai id, email, dan username
id: user[0].id,
user: user[0].username,
email: user[0].email,
});
}
else {
```



Jika password salah



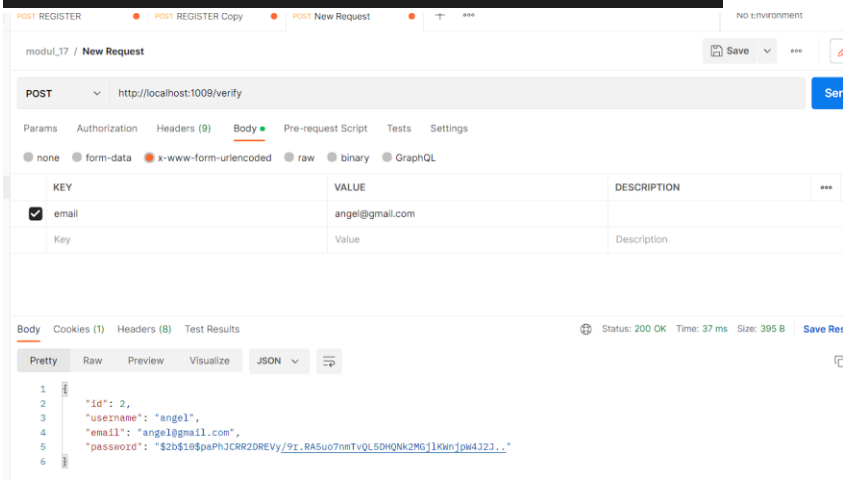
Part 3

12. Jwt verify

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


SECRET = process.env.SECRET
const Auth = {
    verifyToken(req, res, next){
        token = req.cookies["JWT"]
        if (token) {
            // 12. Lalukan jwt verify
            const verified = jwt.verify(token, SECRET)
            if(verified){
              console.log("Verifikasi Berhasil!");
              return next();
            } else {
              return res.status(401).send(error);
            }
        } else {
            res.status(403).send({message: 'Login terlebih
            dahulu!'})
              console.log('Anda tidak terautentikasi!');
        }
    }
}

module.exports = Auth;
```

13. Mebuat verify

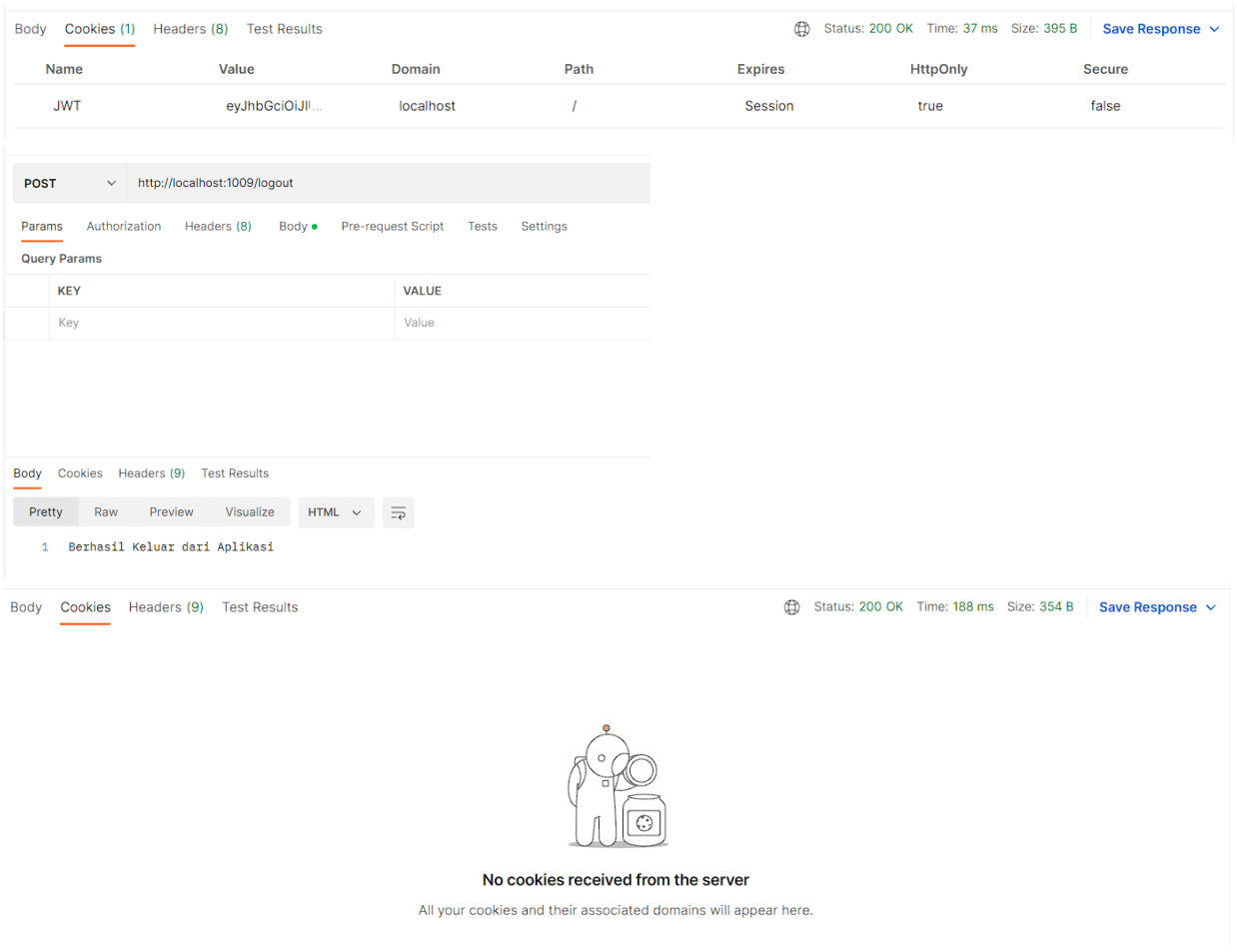
```
const verify = async(req, res, next) => {
    try {
        // 13. membuat verify
        const {email} = req.body;
        const user = await db.query(`SELECT * FROM
        unhan_modul_17 WHERE email=$1;`, [email])
        return res.status(200).json({
            id: user.rows[0].id,
            username: user.rows[0].username,
            email: user.rows[0].email,
            password : user.rows[0].password
        })
    } catch (err) {
        console.log(err.message);
        return res.status(500).send(err)
    }
}
```



Part 4

14. code untuk menghilangkan token dari cookies dan mengembalikan pesan "sudah keluar dari aplikasi"

```
const logout = async(req, res, next) => {

    try {
        // 14. code untuk menghilangkan token dari
        cookies dan mengembalikan pesan "sudah
        keluar dari aplikasi"
        return res.clearCookie("JWT").send("Berhasil
        Keluar dari Aplikasi")
    } catch (err) {
        console.log(err.message);
        return res.status(500).send(err)
    }

}
```



15. error handling

```
    res.send("Data Ditambahkan ke Database")
} catch (err){
    res.send('Inputan Tidak Valid')
}
```

```
Verifying if the user exists in the database
const user = data.rows;
if (user.length === 0) {
res.status(400).json({
error: "User tidak ada, Silahkan Daftar dulu!",
});
}
```

```
    console.log(err);
    res.status(500).json({
    error: "Database error occurred while
    signing in!", //Database connection error
    });
    };
```

```
    }
} else {
  res.status(403).send({message: 'Youre not
  authenticated, please login first'})
    console.log('Youre not authenticated');
}
```

GitHub:
https://github.com/gutri09/Modul17_Gutri-Rahmad-Zuwa