Praktikum 8 - Matakuliah Pilihan 1 (Web)
Program Studi: Teknik Informatika

Lakukan praktikum dibawah ini, dan buat screenshot untuk pembuktian mengerjakan setiap poin dengan mengisi tabel dibawah, kemudian tunjukan hasil akhir dari men-share repository github yang telah dibuat.

A. Membuat Server API dengen Expess.js

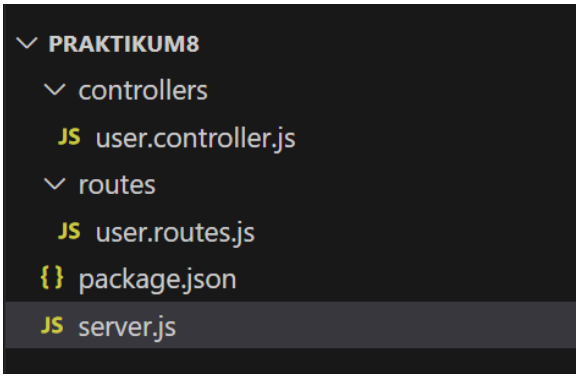
1. Buat sebuah folder proyek API dengan nama **APIproject8**
2. Lakukan seperti pada praktikum 3
   Ketik: npm init -y , setelah itu npm install express
3. Buat file server.js

```
JS server.js > ...
1   const express = require('express');
2   const app = express();
3   const PORT = 8001;
4
5   app.use(express.json());
6
7   app.get('/', (req, res) => {
8     res.send('Hello, World');
9   });
10
11  app.listen(PORT, () => {
12    console.log(`Server berjalan di http://localhost:${PORT}`);
13  });
14
```

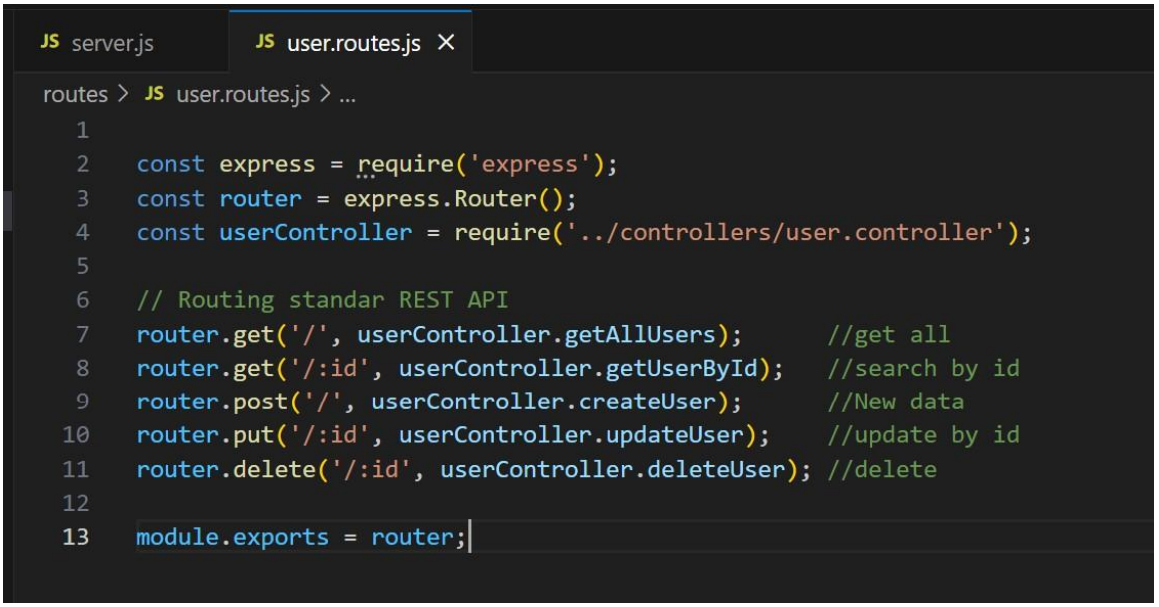
4. Jalankan server.js dengan mengetik
   Ketik: node server.js

B. Membuat Struktur MVC (Routes-Controller)

1. Buat folder **routes, controllers** dan **models**
2. Kemudian didalam folder routes buat sebuah file dengan nama user.routes.js



3. Tulis kode program di file user.routes.js seperti pada gambar dibawah ini



4. Buat file di dalam folder controllers dengan nama user.controller.js
5. Tulis kode program di dalam file user.controller.js seperti pada gambar dibawah ini

Karena pada controller user tersebut require model bernama User, maka kita siapkan Model user, yang berkaitan dengan database.

6. Update file server.js dengan menambahkan kode berikut

Kode diatas pada file server.js untuk memberitahu ada routes bernama userRoutes dengan lokasi file di routes/user.routes (tidak perlu ditulis .js)

C. Membuat koneksi Database dengan Models

1. Nyalakan mysql service dan buatlah sebuah database dengan nama dbpraktikum8

```
CREATE DATABASE IF NOT EXISTS dbpraktikum8;
CREATE TABLE IF NOT EXISTS users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255) DEFAULT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP);
```

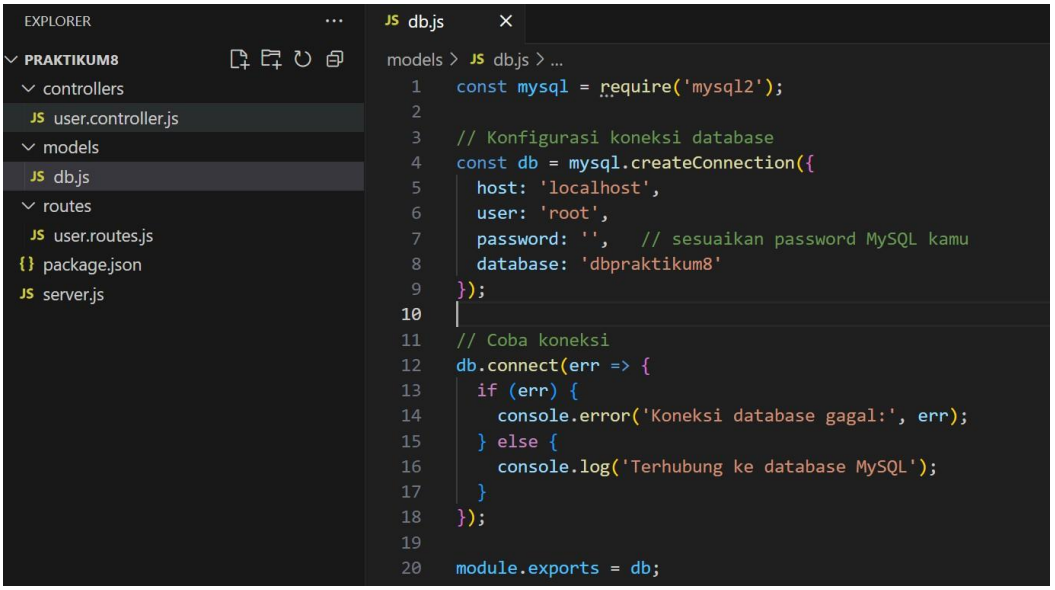
2. Lalu masukan data dummy ke dalamnya

```
INSERT INTO users (name, email, password) VALUES
('Riska Safitri', 'riska@mail.com', '123456'),
('Josephine', 'josep@mail.com', 'abcdef'),
('Moh. Ilham', 'ilham@mail.com', 'qwerty');
```

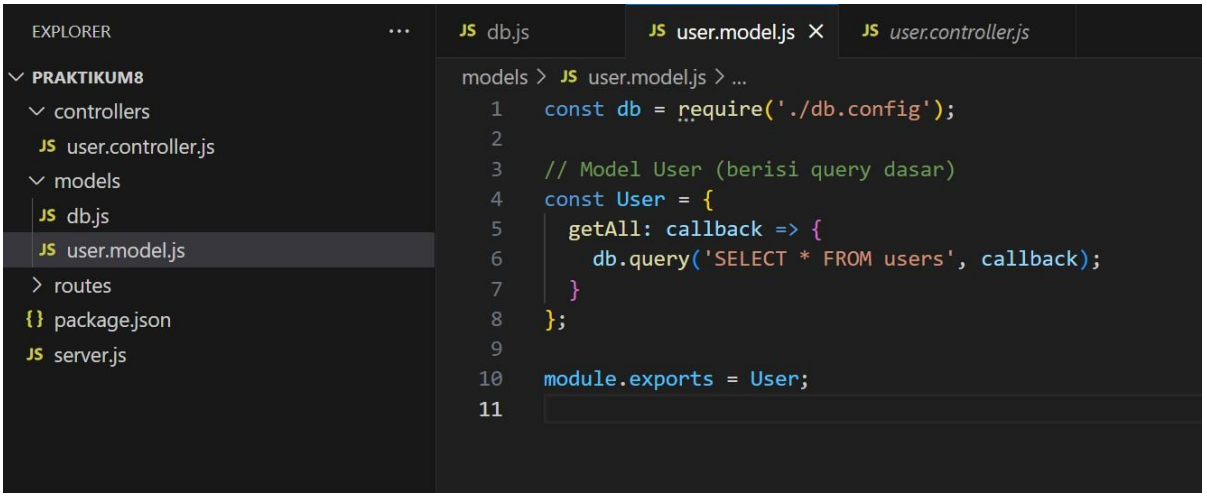
3. Jika database sudah terisi data di tabel users, lalu kita persiapkan kembali di express.js

4. Install Module mysql2 dengan menggunakan node. Masih di folder project ketik perintah berikut: npm install express mysql2

5. Kemudian buat sebuah file di dalam folder models, dengan nama db.config.js dan ketikan seperti berikut



6. File db.config.js adalah sebagai class connector antara express dan database
7. Buat file lagi untuk model user, di dalam folder models. Dengan nama user.model.js



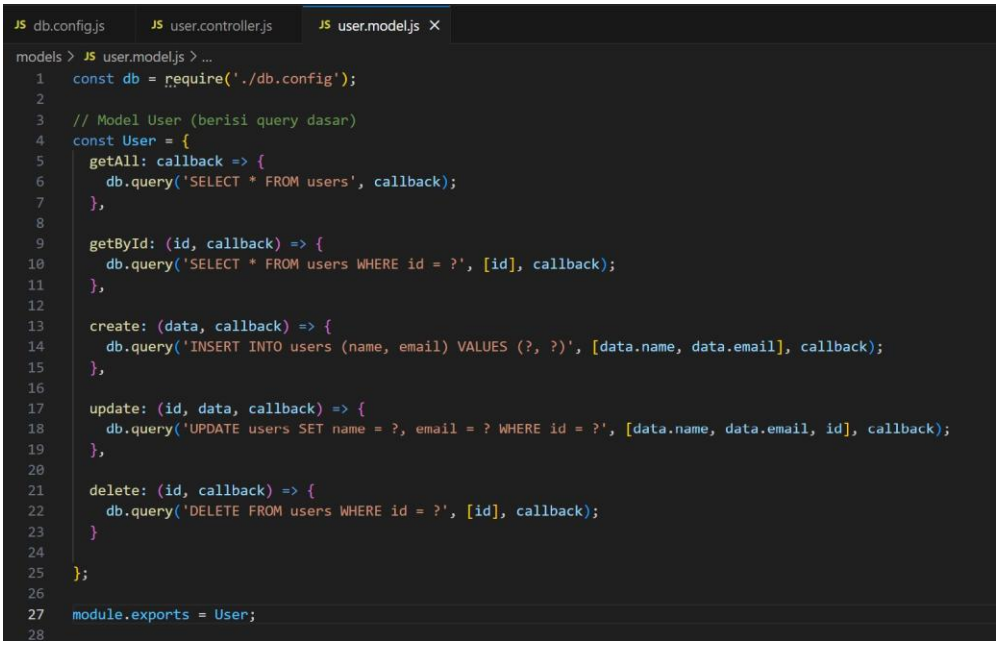
8. Jalankan atau restart ulang node server.js
(Pastikan mysql sudah running, user password mysql sudah benar)

C. Melakukan Test API

Gunakan browser/postman untuk mendapatkan data getAll users dengan mengunjungi endpoints /api/users/

D. Lengkapi Controllers dan Model

1. Tambahkan class untuk model baru, agar terhubung dengan controller. Ubah pada file user.model.js



2. Tambahkan class baru untuk routes yang sudah dipersiapkan lainnya, bisa dilihat pada kode program dibawah ini

File: user.controller.js

```
// GET user by ID
exports.getUserById = (req, res) => {
  const { id } = req.params;
  User.getById(id, (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    if (results.length === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json(results[0]);
  });
};

// POST user baru
exports.createUser = (req, res) => {
  const data = req.body;
  User.create(data, (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    res.status(201).json({ id: result.insertId, ...data });
  });
};

// PUT update user
exports.updateUser = (req, res) => {
  const { id } = req.params;
  const data = req.body;
  User.update(id, data, (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    if (result.affectedRows === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json({ message: 'User berhasil diupdate' });
  });
};

// DELETE user
exports.deleteUser = (req, res) => {
  const { id } = req.params;
  User.delete(id, (err, result) => {
    if (err) return res.status(500).json({ error: err.message });
    if (result.affectedRows === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json({ message: 'User berhasil dihapus' });
  });
};
```

E. Melakukan Test API secara Lengkap

Dengan menggunakan POSTMAN, lakukan pengujian berikut:

1. Menguji endpoint /
2. Menguji endpoint /api/users (Method: GET)
3. Menguji endpoint /api/users/1 (Method: GET)
4. Menguji endpoint /api/users (Method: POST)
   Tambah body -> raw -> JSON
   {
   "name": "Budi Santoso",
   "email": "budi@example.com"
   }

5. Menguji /api/users/2 (Method: PUT)
   Masukan Body -> raw -> JSON
   ```
   {
     "name": "Joe Taslim",
     "email": "jojo@example.com"
   }
   ```

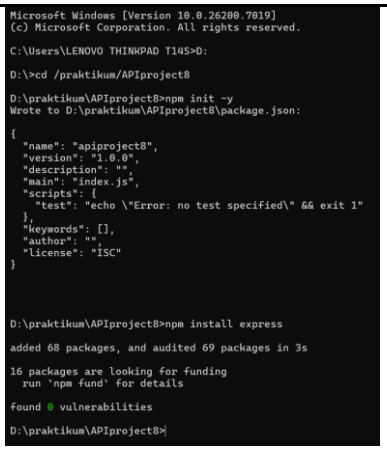
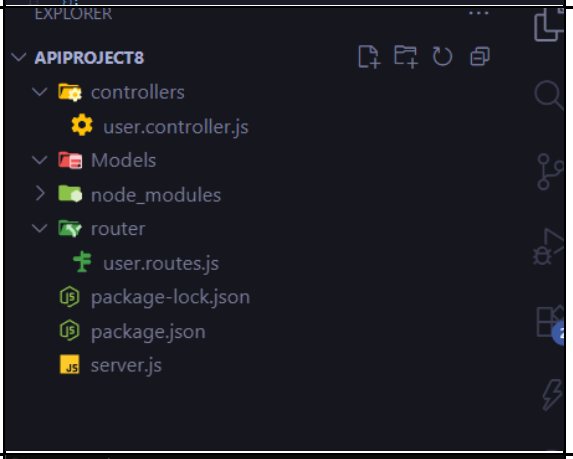
6. Menguji /api/users/3 (Method: DELETE)

F. Github + Visual Code

1. Buat proyek di Github dengan nama **Latihan8**

   ```
   git init
   git add .
   git commit -m "first commit"
   git branch -M main
   git remote add origin https://github.com/agunghakase/Latihan8.git
   git push -u origin main
   ```

Hasil Pengerjaan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Installasi dan Konfigurasi		
1.	Membuat Server API dengan Express.js	 	-
2.	Membuat struktur MVC		-
3.	Membuat database dengan models		-

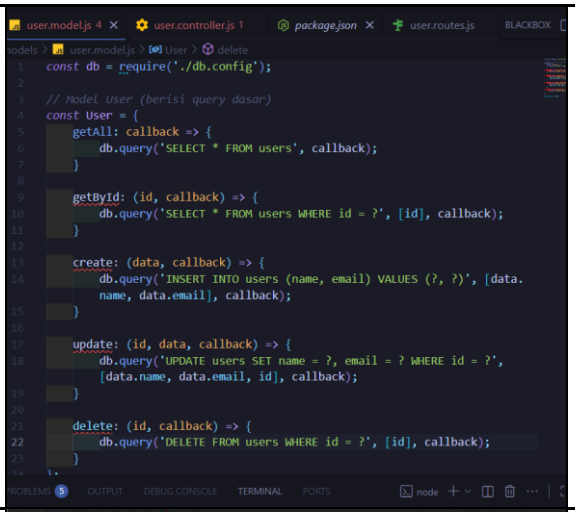
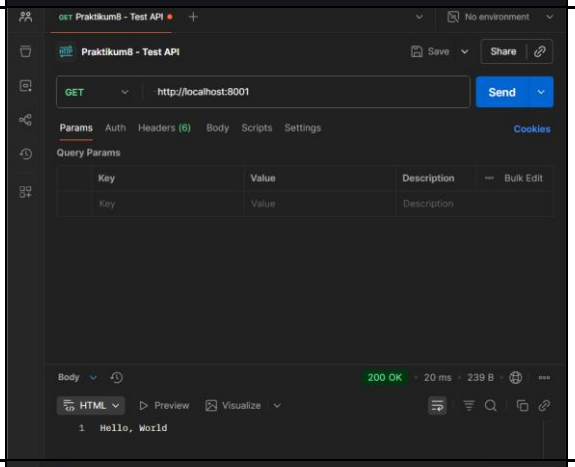
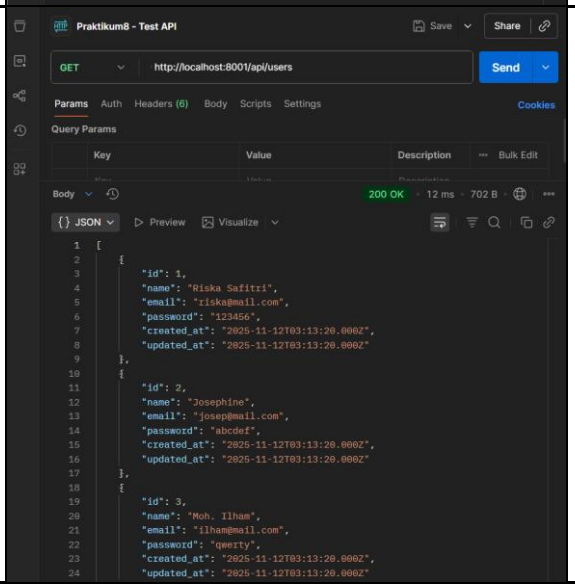
4.	Install module mysql2	powershell PS D:\praktikum\APIproject8> npm install express mysql2 added 12 packages, and audited 81 packages in 2s 17 packages are looking for funding run `npm fund` for details found 0 vulnerabilities	-

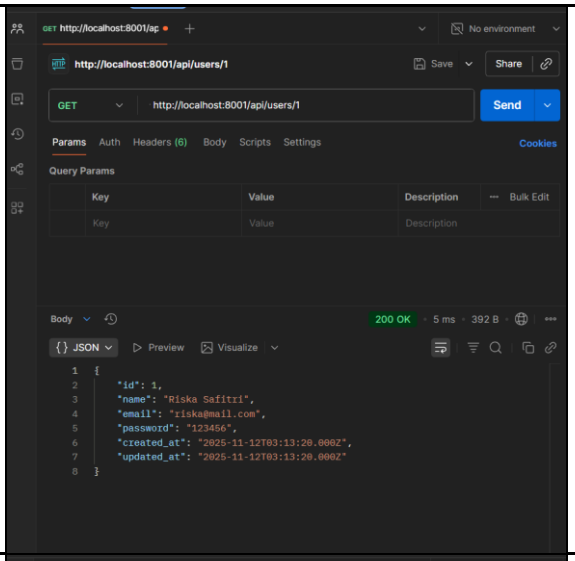
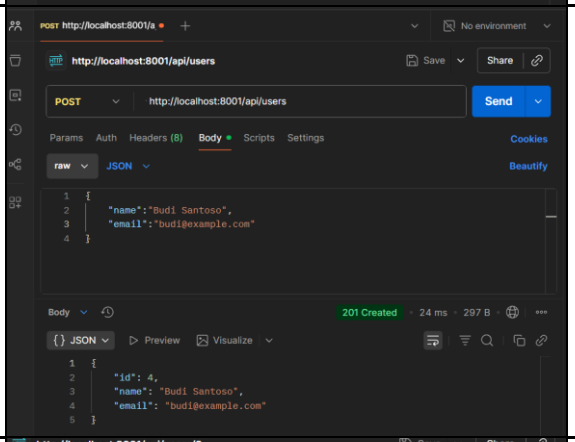
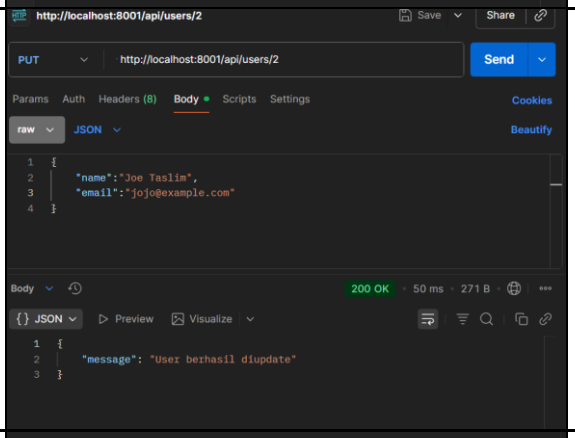
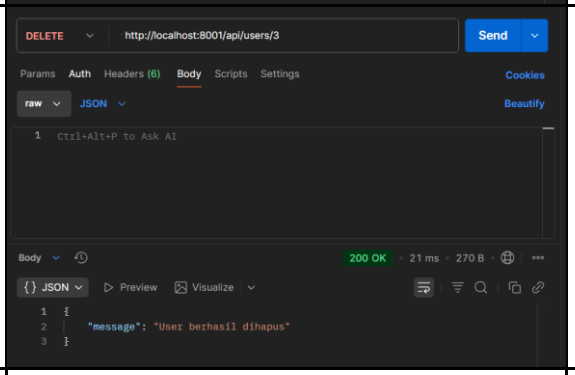
5.	Membuat db.config.js	server.js db.config.js user.controller.js
		EXPLORER APIPROJECT8
		controllers
		user.controller.js
		Models
		db.config.js
		node_modules
		router
		user.routes.js
		package-lock.json
		package.json
		server.js
		const mysql = require('mysql2');
		// Konfigurasi koneksi database
		const db = mysql.createConnection({
		host: 'localhost',
		user: 'root',
		password: 'root123',
		database: 'dbpraktikum8'
		});
		// Coba koneksi
		db.connect(err => {
		if (err) {
		console.error('Koneksi database gagal:',
		err);
		} else {
		console.log('Terhubung ke database MySQL');
		}
		});
		module.exports = db;
			-

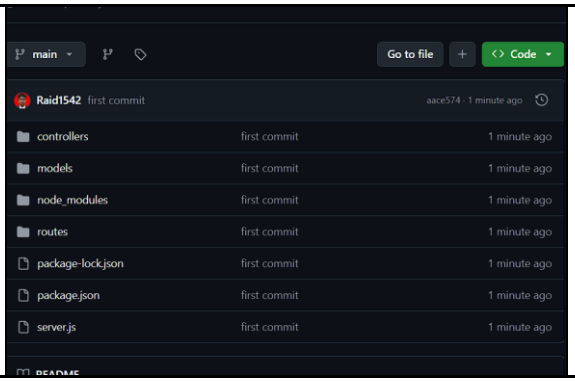
6.	Membuat user.model.js	server.js db.config.js user.model.js
		EXPLORER APIPROJECT8
		controllers
		user.controller.js
		Models
		db.config.js
		user.model.js
		node_modules
		router
		user.routes.js
		package-lock.json
		package.json
		server.js
		const db = reuqire('../db.config');
		// Model User (berisi query dasar)
		const User = {
		getAll: callback => {
		db.query('SELECT * FROM users', callback);
		}
		};
		modeule.exports = User;
			-

7.	Jalankan server.js	Node.js v22.19.0 PS D:\praktikum\APIproject8> node server.js D:\praktikum\APIproject8\node_modules\router\lib\route.js:228 throw new TypeError('argument handler must be a function') ^ TypeError: argument handler must be a function at Route.<computed> [as get] (D:\praktikum\APIproject8\node_modules\router\lib\route.js:228:15) at Router.<computed> [as get] (D:\praktikum\APIproject8\node_modul	-

8.	Melengkapi Controllers dan Model	// GET semua user
		exports.getAllUsers = (req, res) => {
		User.getAll((err, results) => { //ambil dari models
		if (err) return res.status(500).json({ error: err.message });
		res.json(results);
		});
		};
		
		// GET user by ID
		exports.getUserById = (req, res) => {
		const data = req.body;
		User.create(data, (err, results) => {
		if (err) return res.status(500).json({ error: err.message });
		res.status(201).json({ id: results.insertId, ...data });
		});
		};
		
		// POST user baru
		exports.createUser = (req, res) => {
		const data = req.body;
		User.create(data, (err, results) => {
		if (err) return res.status(500).json({ error: err.message });
			-

			
9.	Menguji endpoint /		-
10.	Menguji endpoint /api/users (Method: GET)		-

11.	Menguji endpoint /api/users/1 (Method: GET)		-
12.	Menguji endpoint /api/users (Method: POST)		-
13.	Menguji /api/users/2 (Method: PUT)		-
14.	Menguji /api/users/3 (Method: DELETE)		-
B.	Github dan Viscode		

1.	Upload di github		-