

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 dengan Express.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.

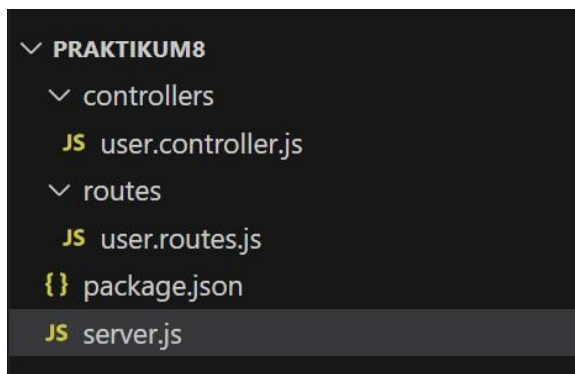
Buat file server.js

```
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

```
JS server.js JS user.routes.js X
routes > JS user.routes.js > ...
1
2 const express = require('express');
3 const router = express.Router();
4 const userController = require('../controllers/user.controller');
5
6 // Routing standar REST API
7 router.get('/', userController.getAllUsers); //get all
8 router.get('/:id', userController.getUserById); //search by id
9 router.post('/', userController.createUser); //New data
10 router.put('/:id', userController.updateUser); //update by id
11 router.delete('/:id', userController.deleteUser); //delete
12
13 module.exports = router;
```

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

```
const User = require('../models/user.model'); //memanggil 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);
  });
};
```

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

```
8 // Routes
9 const userRoutes = require('./routes/user.routes');
10 app.use('/api/users', userRoutes);
```

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

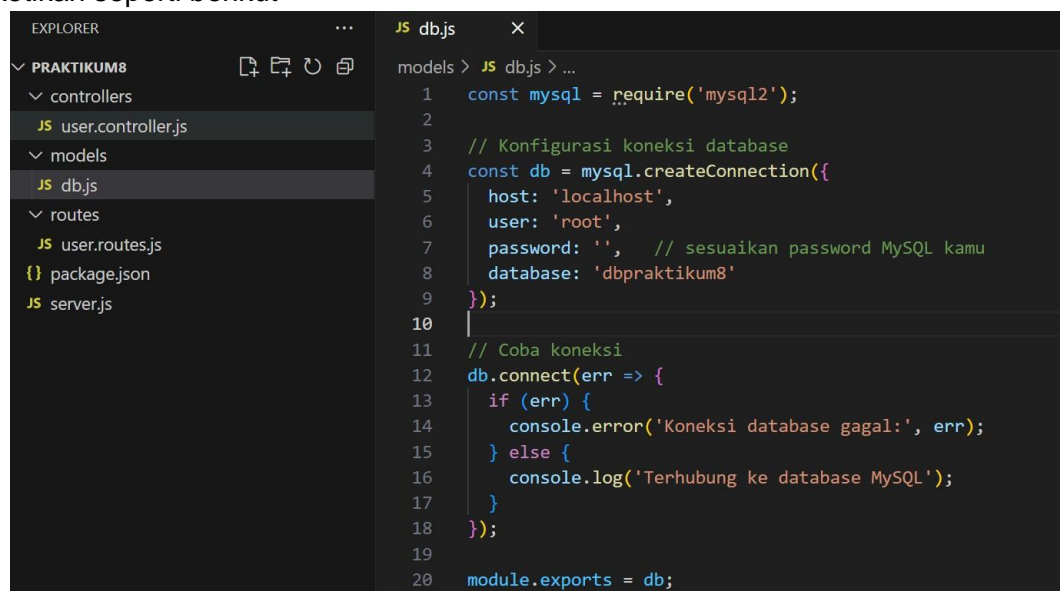
1. Nyalakan mysql service dan buatlah sebuah database dengan nama dbpraktikum8

```
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');
```

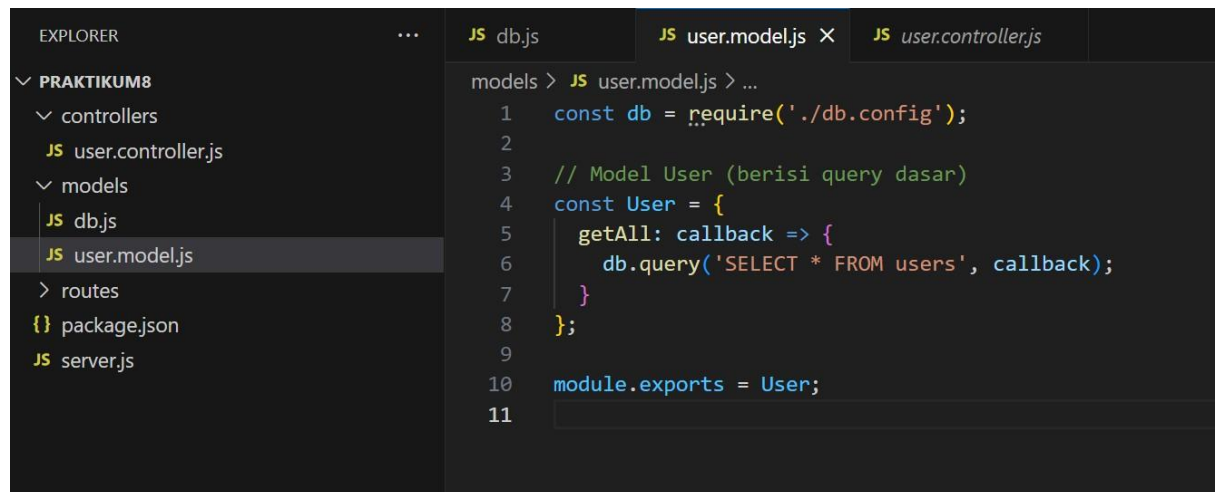
3. Jika database sudah terisi data di tabel users, lalu kita persiapkan kembali di [express.js](#)
4. Install Module mysql2 dengan menggunakan node. Masih di folder project ketik perintah berikut: `npm install express mysql2`
5. Kemudian buat sebuah file di dalam folder models, dengan nama [db.config.js](#) dan ketikan seperti berikut



The screenshot shows the VS Code interface. On the left, the 'EXPLORER' sidebar displays the project structure for 'PRAKTIKUM8', including folders 'controllers', 'models', and 'routes', and files 'user.controller.js', 'db.js', 'user.routes.js', 'package.json', and 'server.js'. The 'models' folder is expanded, and 'db.js' is selected. The main editor area shows the code for 'db.js'.

```
JS db.js
models > JS db.js > ...
1  const mysql = require('mysql2');
2
3  // Konfigurasi koneksi database
4  const db = mysql.createConnection({
5    host: 'localhost',
6    user: 'root',
7    password: '', // sesuaikan password MySQL kamu
8    database: 'dbpraktikum8'
9  });
10
11 // Coba koneksi
12 db.connect(err => {
13   if (err) {
14     console.error('Koneksi database gagal:', err);
15   } else {
16     console.log('Terhubung ke database MySQL');
17   }
18 });
19
20 module.exports = db;
```

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



The screenshot shows the VS Code interface. On the left, the 'EXPLORER' sidebar displays a project structure under 'PRAKTIKUM8' with folders 'controllers' and 'models'. The 'models' folder is expanded, showing 'db.js', 'user.model.js' (selected), and 'server.js'. The main editor area shows the code in 'user.model.js' with line numbers 1 through 11. The code defines a 'User' model with a 'getAll' method that queries a database.

```
1 const db = require('./db.config');
2
3 // Model User (berisi query dasar)
4 const User = {
5   getAll: callback => {
6     db.query('SELECT * FROM users', callback);
7   }
8 };
9
10 module.exports = User;
11
```

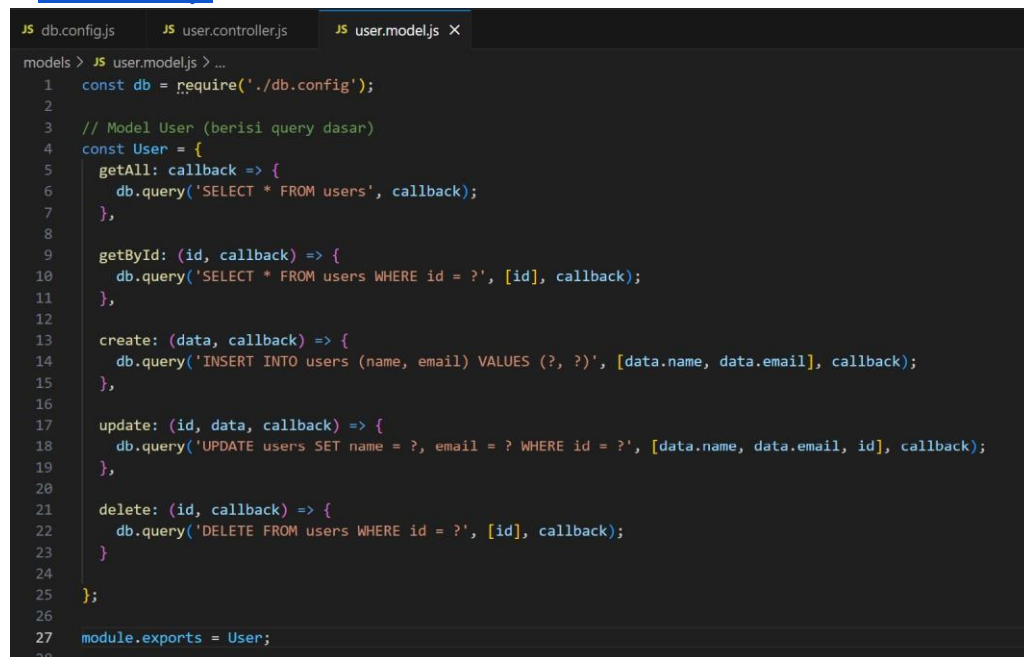
8. Jalankan atau restart ulang node [server.js](#)
(Pastikan mysql sudah running, user password mysql sudah benar)

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](#)



The screenshot shows the VS Code interface with the 'user.model.js' file open. The code now includes methods for 'getAll', 'getById', 'create', 'update', and 'delete' for the 'User' model. Line numbers 1 through 28 are visible. The 'create', 'update', and 'delete' methods are newly added.

```
1 const db = require('./db.config');
2
3 // Model User (berisi query dasar)
4 const User = {
5   getAll: callback => {
6     db.query('SELECT * FROM users', callback);
7   },
8   getById: (id, callback) => {
9     db.query('SELECT * FROM users WHERE id = ?', [id], callback);
10   },
11   create: (data, callback) => {
12     db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback);
13   },
14   update: (id, data, callback) => {
15     db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback);
16   },
17   delete: (id, callback) => {
18     db.query('DELETE FROM users WHERE id = ?', [id], callback);
19   }
20 };
21
22 module.exports = User;
23
24
25
26
27
28
```

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
```

```
.
```

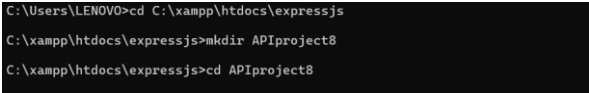
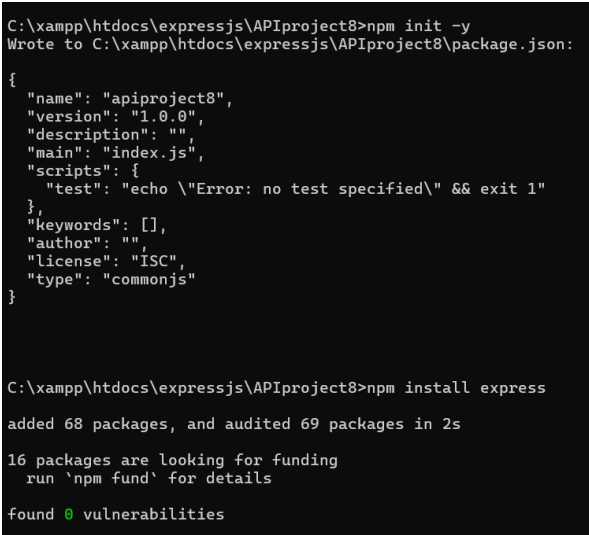
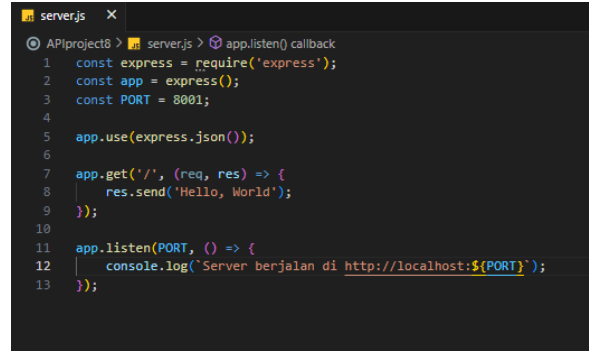
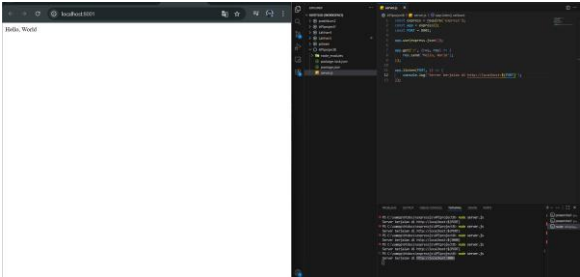
```
git commit -m "first commit"
```

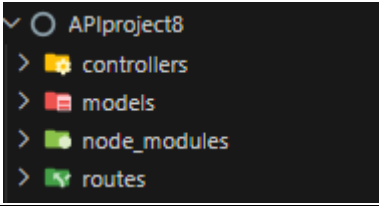
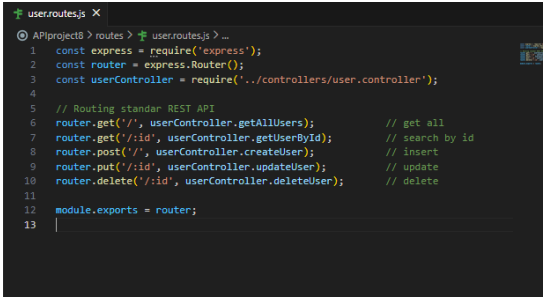
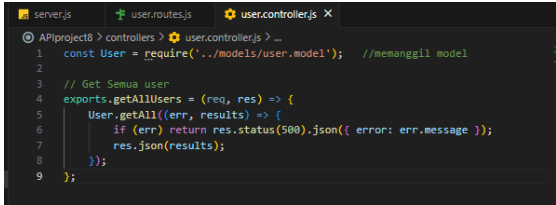
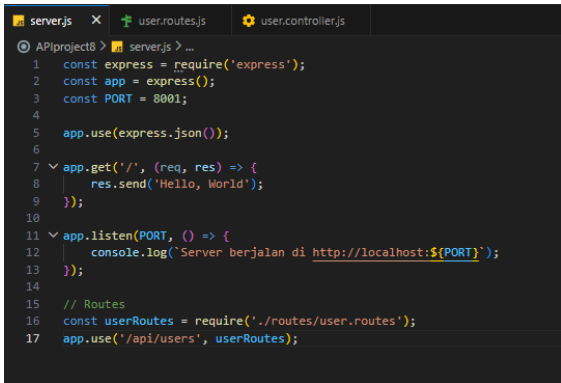

```
git branch -M main
```

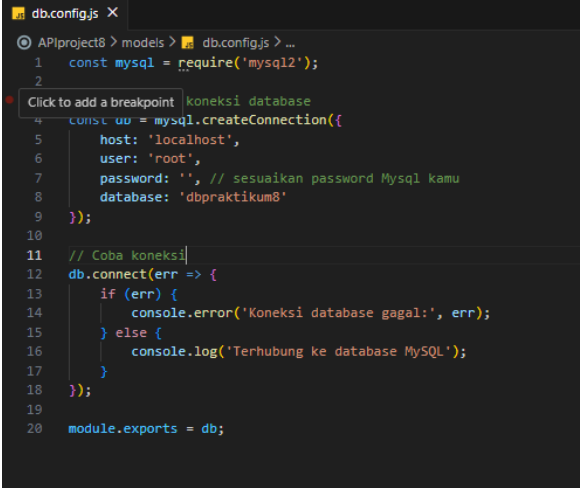
```
git remote add origin https://github.com/agunghakase/Latihan8.git
```

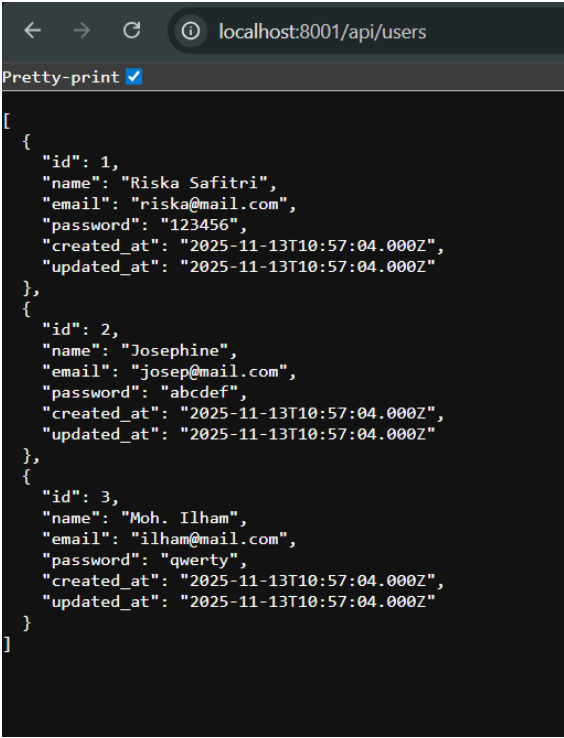
```
git push -u origin main
```

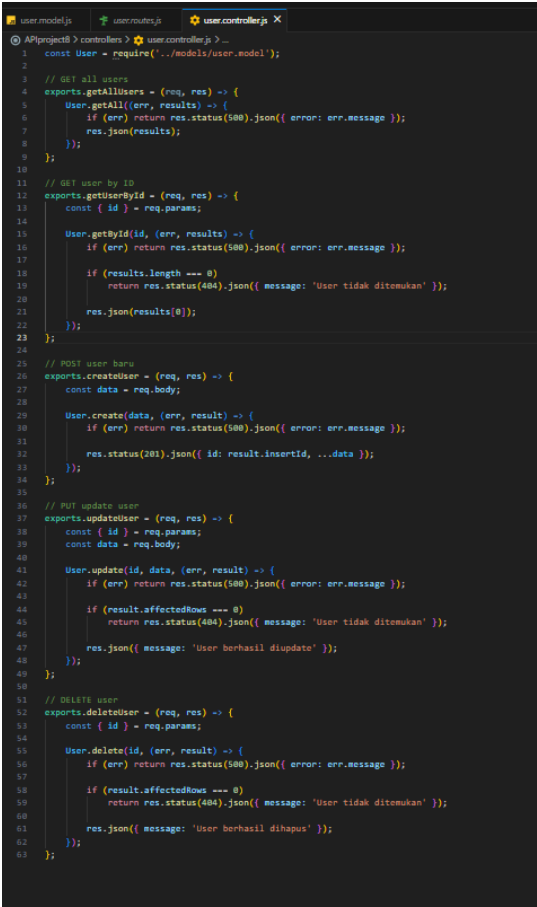
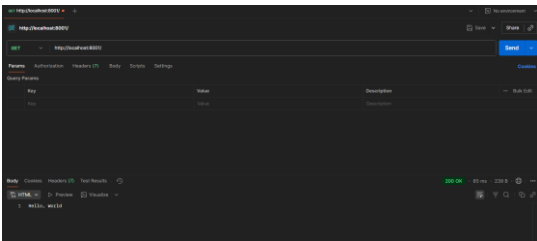
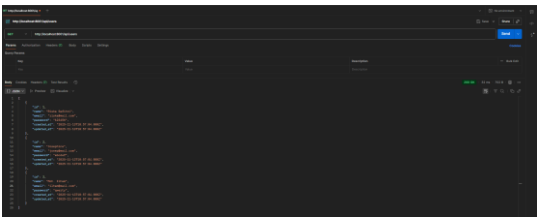
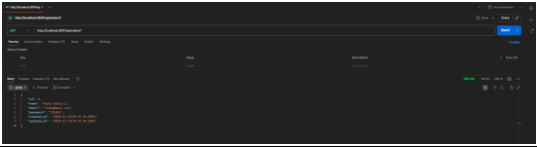
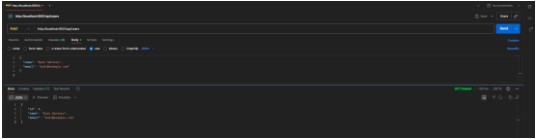
Hasil Pengerjaan

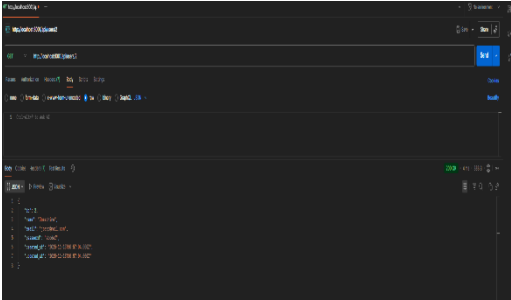
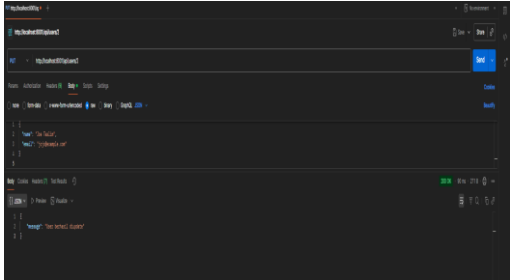
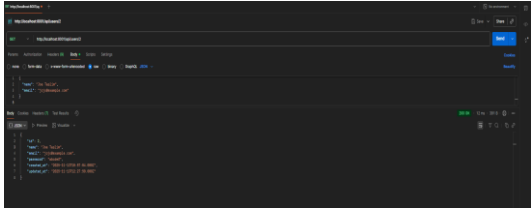
No.	Instruksi	Screenshot	Kendala/Saran
A.	Instalasi dan Konfigurasi		
1.	Membuat Folder APIproject8	 <pre>C:\Users\LENOVO>cd C:\xampp\htdocs\expressjs C:\xampp\htdocs\expressjs>mkdir APIproject8 C:\xampp\htdocs\expressjs>cd APIproject8</pre>	
2.	Instalasi expressjs	 <pre>C:\xampp\htdocs\expressjs\APIproject8>npm init -y Wrote to C:\xampp\htdocs\expressjs\APIproject8\package.json: { "name": "apiproject8", "version": "1.0.0", "description": "", "main": "index.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "keywords": [], "author": "", "license": "ISC", "type": "commonjs" } C:\xampp\htdocs\expressjs\APIproject8>npm install express added 68 packages, and audited 69 packages in 2s 16 packages are looking for funding run `npm fund` for details found 0 vulnerabilities</pre>	
3.	Membuat File server.js	 <pre>server.js APIproject8 > server.js > app.listen() callback 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 });</pre>	
4.	Menjalankan node server.js		

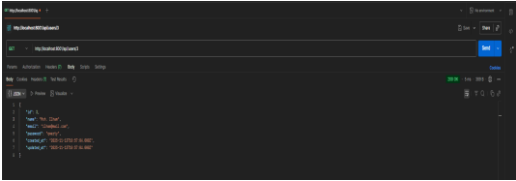
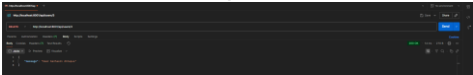
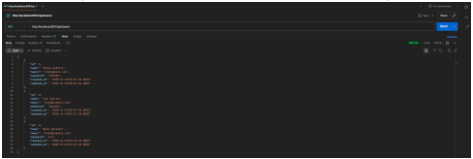
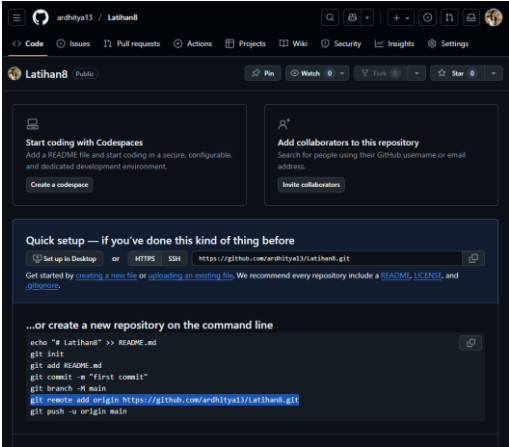
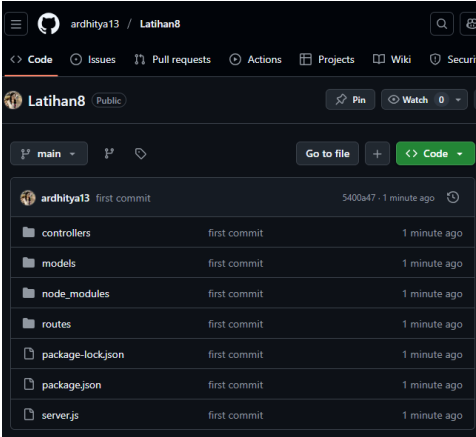
B.	Github dan Viscode		
1.	Membuat folder routes, controllers, models		
2.	Membuat File user.routes.js dan memasukan codingannya		
3.	Membuat user.controller.js		
4.	Menambahkan routes di server.js		
5.	Membuat database dbpraktikum8 dan mengisi sql nya		

6.	Memasukkan data dummynya ke dalam tabel users		
7.	Install module npm install express mysql2	<pre>PS C:\xampp\htdocs\expressjs\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</pre>	
8.	Membuat file db.config.js Dan mengisi codingan		
10.	Membuat file user.model.js dan mengisi codingan		
11.	Menjalankan server.js	<pre>PS C:\xampp\htdocs\expressjs\APIproject8> node server.js Server berjalan di http://localhost:8001 Terhubung ke database MySQL</pre>	Di bagian ini saya harus melengkapi user.model.js Dan user.controller.js nya karenaa kalo belum di lengkapi dia

		<pre>PS C:\xampp\htdocs\expressjs\APIproject8> node server.js C:\xampp\htdocs\expressjs\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] (C:\xampp\htdocs\expressjs\APIproject8\node_modules\rout er\lib\route.js:228:15) at Router.<computed> [as get] (C:\xampp\htdocs\expressjs\APIproject8\node_modules\rout er\index.js:448:19) at Object.<anonymous> (C:\xampp\htdocs\expressjs\APIproject8\routes\user.routes.js:7: 8) at Module._compile (node:internal/modules/cjs/loader:1769:14) at Object.<js> (node:internal/modules/cjs/loader:1893:10) at Module.load (node:internal/modules/cjs/loader:1480:32) at Module._load (node:internal/modules/cjs/loader:1299:12) at TracingChannel.traceSync (node:diagnostics_channel:322:14) at wrapModuleLoad (node:internal/modules/cjs/loader:244:24) at Module.require (node:internal/modules/cjs/loader:1503:12) Node.js v24.8.0</pre>	akan eror ada contohnya Jadi wajib di lengkapi dulu 2 file itu baru bisa di testing di browser
12.	Membuka browser http://localhost:8001/ap i/users	 <pre>[{ "id": 1, "name": "Riska Safitri", "email": "riska@mail.com", "password": "123456", "created_at": "2025-11-13T10:57:04.000Z", "updated_at": "2025-11-13T10:57:04.000Z" }, { "id": 2, "name": "Josephine", "email": "josep@mail.com", "password": "abcdef", "created_at": "2025-11-13T10:57:04.000Z", "updated_at": "2025-11-13T10:57:04.000Z" }, { "id": 3, "name": "Moh. Ilham", "email": "ilham@mail.com", "password": "qwerty", "created_at": "2025-11-13T10:57:04.000Z", "updated_at": "2025-11-13T10:57:04.000Z" }]</pre>	
13.	Melengkapi user.model.js	<pre>user.model.js 1 // Import mongoose 2 const db = require('./db.config'); 3 4 // Model user (berisi query dasar) 5 const User = { 6 getAll: callback => { 7 db.query('SELECT * FROM users', callback); 8 }, 9 getById: (id, callback) => { 10 db.query('SELECT * FROM users WHERE id = ?', [id], callback); 11 }, 12 create: (data, callback) => { 13 db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback); 14 }, 15 update: (id, data, callback) => { 16 db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback); 17 }, 18 delete: (id, callback) => { 19 db.query('DELETE FROM users WHERE id = ?', [id], callback); 20 } 21 }; 22 23 module.exports = User;</pre>	

14.	Melengkapi file user.controller.js	 <pre>1 const User = require('../models/user-model'); 2 3 // GET all users 4 exports.getAllUsers = (req, res) => { 5 User.getAll((err, results) => { 6 if (err) return res.status(500).json({ error: err.message }); 7 res.json(results); 8 }); 9 } 10 11 // GET user by ID 12 exports.getUserById = (req, res) => { 13 const { id } = req.params; 14 User.getById(id, (err, results) => { 15 if (err) return res.status(500).json({ error: err.message }); 16 if (results.length === 0) 17 return res.status(404).json({ message: 'User tidak ditemukan' }); 18 res.json(results[0]); 19 }); 20 } 21 22 // POST user baru 23 exports.createUser = (req, res) => { 24 const data = req.body; 25 User.create(data, (err, result) => { 26 if (err) return res.status(500).json({ error: err.message }); 27 res.status(201).json({ id: result.insertId, ...data }); 28 }); 29 } 30 31 // PUT update user 32 exports.updateUser = (req, res) => { 33 const { id } = req.params; 34 const data = req.body; 35 User.update(id, data, (err, result) => { 36 if (err) return res.status(500).json({ error: err.message }); 37 if (result.affectedRows === 0) 38 return res.status(404).json({ message: 'User tidak ditemukan' }); 39 res.json({ message: 'User berhasil diupdate' }); 40 }); 41 } 42 43 // DELETE user 44 exports.deleteUser = (req, res) => { 45 const { id } = req.params; 46 User.delete(id, (err, result) => { 47 if (err) return res.status(500).json({ error: err.message }); 48 if (result.affectedRows === 0) 49 return res.status(404).json({ message: 'User tidak ditemukan' }); 50 res.json({ message: 'User berhasil dihapus' }); 51 }); 52 };</pre>	
15.	Menguji endpoint /		
16.	Menguji endpoint /api/users (Method: GET)		
17.	Menguji endpoint /api/users/1 (Method: GET)		
18.	Menguji endpoint /api/users (Method: POST)		

	<div>Tambah body - > raw -> JSON</div> <div><pre>{ "name": "Budi Santos o", "email": "budi@ exampl e.com" }</pre></div>		
19.	<div>Menguji /api/users/2 (Method: PUT) Masukan Body -> raw -> JSON</div> <div><pre>{ "name": "Joe Taslim", "email": "jojo@e xample. com" }</pre></div>	<div>sebelum di update</div> <div></div> <div>Ketika data di update</div> <div></div> <div>Ini hasilnya</div> <div></div>	

20.	Menguji /api/users/3 (Method: DELETE)	<p>Data sebelum di hapus</p>  <p>Data Ketika di hapus</p>  <p>Cek apakah data sudah di hapus</p> 	
21.	Membuat repository Latihan8		
22.	Push ke github	<pre>PS C:\xampp\htdocs\expressjs\API\project8> git push -u origin main git: 'credential-manager-core' is not a git command. See 'git --help'. Enumerating objects: 963, done. Counting objects: 100% (963/963), done. Delta compression using up to 12 threads Compressing objects: 100% (893/893), done. Writing objects: 100% (963/963), 1.05 MiB 71.00 KiB/s, done. Total 963 (delta 159), reused 0 (delta 0), pack-reused 0 (from 0) remote: Resolving deltas: 100% (159/159), done. To https://github.com/ardhitya13/Latihan8.git * [new branch] main -> main branch 'main' set up to track 'origin/main'.</pre>	
23.	Berhasil push ke github		

24.	Link github	https://github.com/ardhitya13/Latihan8	
-----	-------------	---	--