

Praktikum 8 - Matakuliah Pilihan 1 (Web)

Program Studi: Teknik Informatika

Nama: Salsa Putri Ajriyanti

Nim: 3312401043

Kelas: IF 3A Pagi

Lakukan praktikum dibawah ini, dan buat screenshot untuk pembuktian mengerjakan setiap poin dengan mengisi tabel dibawah, kemudian tunjukkan 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

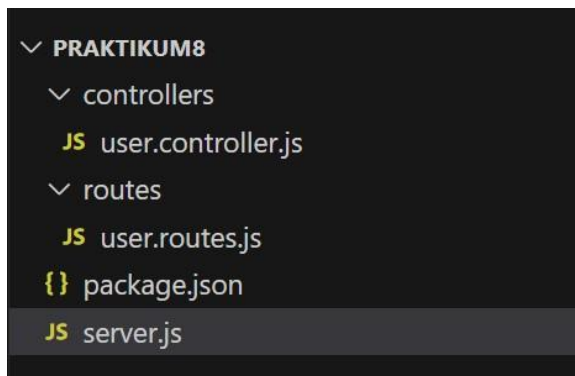
```
JS server.js > ...
1  const express = require('express');
2  const app = express();
3  const PORT = 8001;
4
5  app.use(express.json());
6
7  app.get('/', (req, res) => {
8    res.send('Hello, World');
9  });
10
11 app.listen(PORT, () => {
12   console.log(`Server berjalan di http://localhost:${PORT}`);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

```

7
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

```
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`

```
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](#)

```

JS db.config.js JS user.controller.js JS user.model.js X
models > JS user.model.js > ...
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    getById: (id, callback) => {
10     db.query('SELECT * FROM users WHERE id = ?', [id], callback);
11   },
12
13   create: (data, callback) => {
14     db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback);
15   },
16
17   update: (id, data, callback) => {
18     db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback);
19   },
20
21   delete: (id, callback) => {
22     db.query('DELETE FROM users WHERE id = ?', [id], callback);
23   }
24 }
25 };
26
27 module.exports = User;
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
```

```
.
```

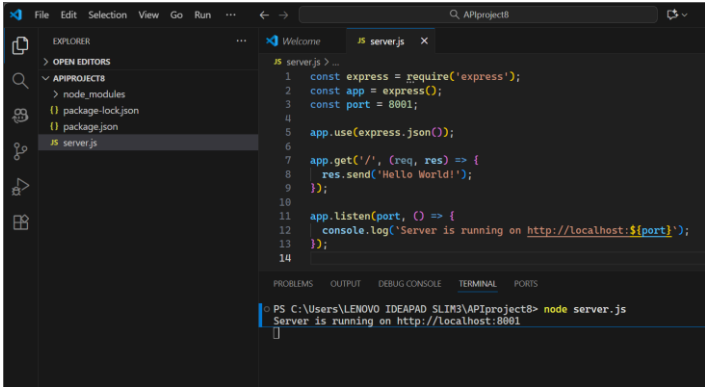
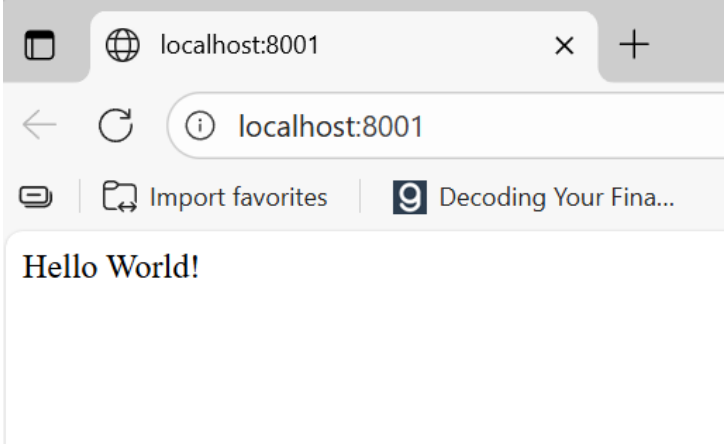
```
git commit -m "first commit"
```

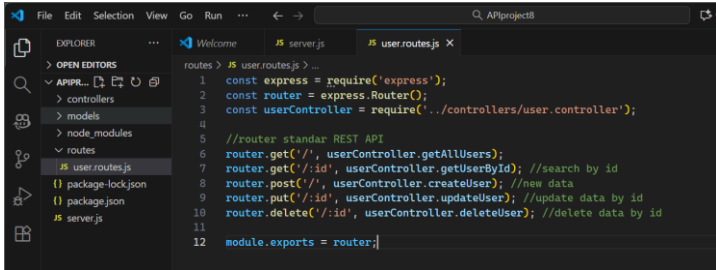
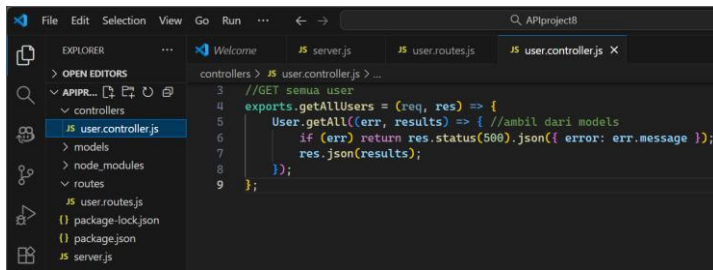
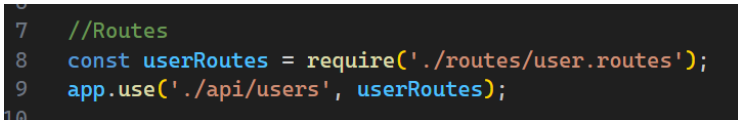
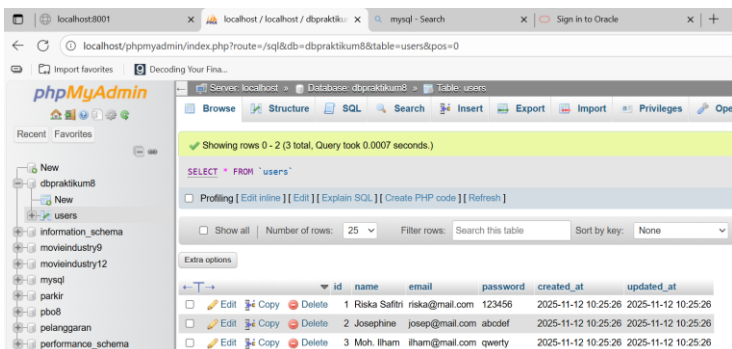
```
git branch -M main
```

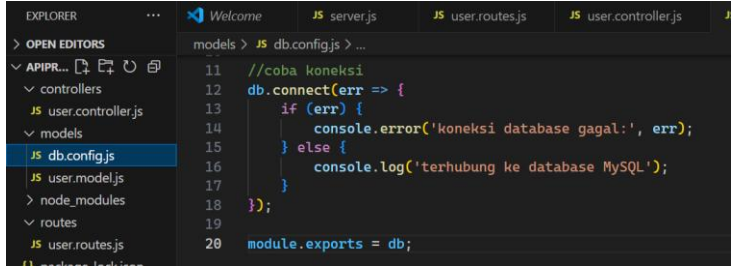
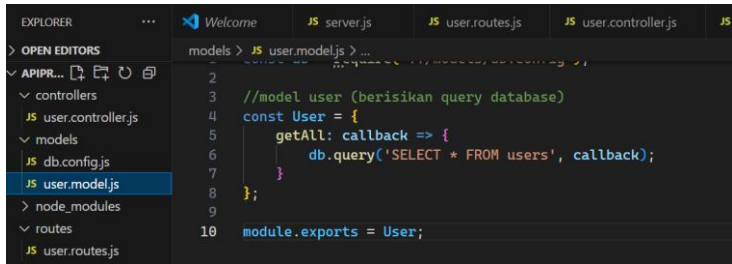
```
git remote add origin https://github.com/agunghakase/Latihan8.git
```

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

Hasil Pengerjaan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Membuat server api dengan express js		
1.	Membuat sebuah folder proyek dengan nama APIproject8, ketik npm init -y dan npm install express	<pre> C:\Users\LENOVO IDEAPAD SLIM3>mkdir APIproject8 C:\Users\LENOVO IDEAPAD SLIM3>cd APIproject8 C:\Users\LENOVO IDEAPAD SLIM3\APIproject8>npm init -y Wrote to C:\Users\LENOVO IDEAPAD SLIM3\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:\Users\LENOVO IDEAPAD SLIM3\APIproject8>npm install express added 68 packages, and audited 69 packages in 4s 16 packages are looking for funding run `npm fund` for details </pre>	
2.	Buat file server.js serta isi kode sesuai dengan perintah dan Jalankan server.js dengan mengetik node server.js	 <p>The screenshot shows the Visual Studio Code editor with a file named <code>server.js</code> open. The code in the file is as follows:</p> <pre> 1 const express = require('express'); 2 const app = express(); 3 const port = 8081; 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 is running on http://localhost:\${port}`); 13 }); 14 </pre> <p>Below the editor, the terminal window shows the command <code>node server.js</code> being executed, resulting in the output: <code>Server is running on http://localhost:8081</code>.</p>	
3.	Output dari node server.js	 <p>The screenshot shows a web browser window with the address bar set to <code>localhost:8001</code>. The page content displays <code>Hello World!</code> in a large, bold font.</p>	

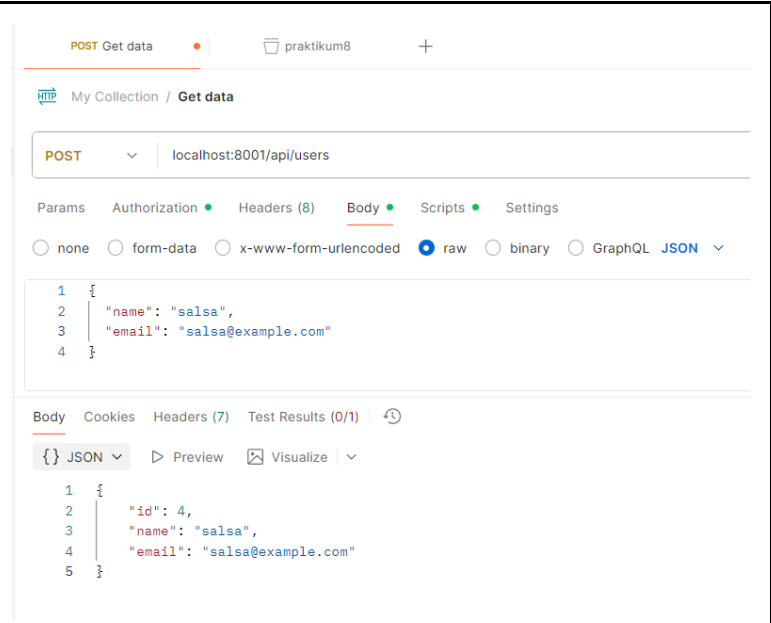
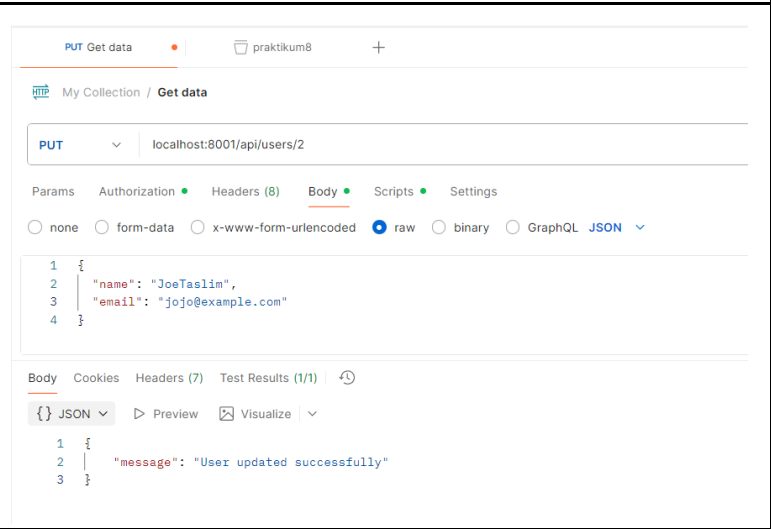
B.	Membuat struktur mvc (routes-controller)		
1.	Buat folder routes, controllers dan models kemudian di dalam folder routes buat sebuah file dengan nama user.routes.js dan tuliskan kode program di file user.routes.js	 <pre> 1 const express = require('express'); 2 const router = express.Router(); 3 const userController = require('../controllers/user.controller'); 4 5 //router standar REST API 6 router.get('/', userController.getAllUsers); 7 router.get('/:id', userController.getUserById); //search by id 8 router.post('/', userController.createUser); //new data 9 router.put('/:id', userController.updateUser); //update data by id 10 router.delete('/:id', userController.deleteUser); //delete data by id 11 12 module.exports = router; </pre>	
2.	Buat file didalam folder controllers dengan nama user.controllers.js dan tuliskan kode programnya	 <pre> 3 //GET semua user 4 exports.getAllUsers = (req, res) => { 5 User.getAll((err, results) => { //ambil dari models 6 if (err) return res.status(500).json({ error: err.message }); 7 res.json(results); 8 }); 9 }; </pre>	
3.	Update file server.js dengan menambahkan kode berikut. Kode tersebut untuk memberitahu ada routes Bernama userRoutes dengan Lokasi file di routes/user.routes	 <pre> 7 //Routes 8 const userRoutes = require('./routes/user.routes'); 9 app.use('./api/users', userRoutes); 10 </pre>	
C.	Membuat koneksi database dengan models		
1.	Nyalakan mysql service dan buatlah database dengan nama dbpraktikum8 dan masukkan data dummy ke dalamnya		

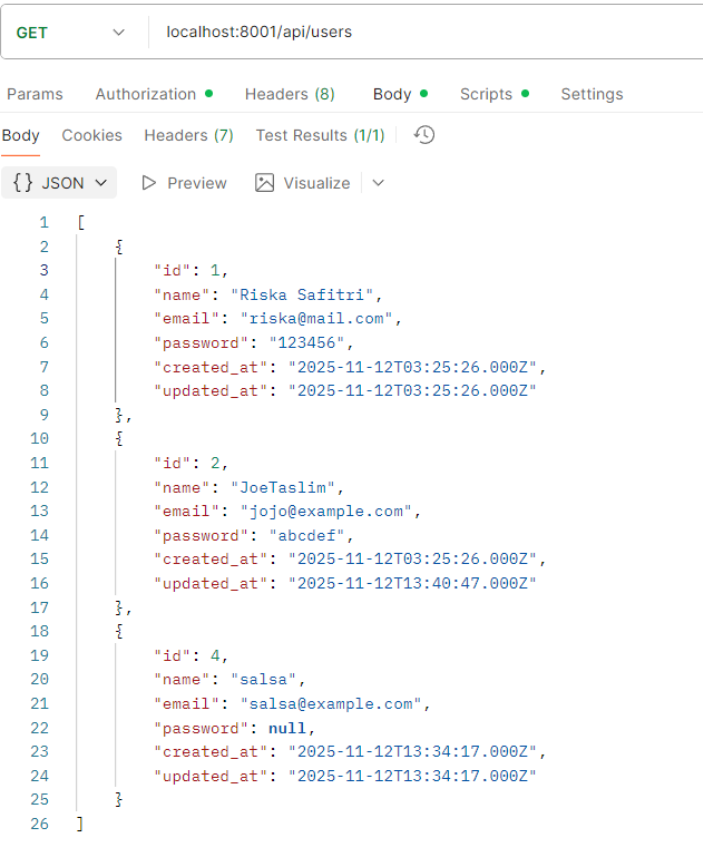
2.	Database sudah terisi, selanjutnya npm install express mysql2	<pre> PS C:\Users\LENOVO IDEAPAD SLIM3\APIproject8> npm install express mysql2 added 12 packages, and audited 81 packages in 4s 17 packages are looking for funding run `npm fund` for details found 0 vulnerabilities PS C:\Users\LENOVO IDEAPAD SLIM3\APIproject8> </pre>	
3.	Buat file db.config.js di folder model dan isikan kode seperti perintah	 <pre> models > JS db.config.js > ... 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; </pre>	
4.	Buat file user.model.js di folder model. Jalankan dan restart ulang node	 <pre> models > JS user.model.js > ... 2 //model user (berisikan query database) 3 4 const User = { 5 getAll: callback => { 6 db.query('SELECT * FROM users', callback); 7 } 8 }; 9 10 module.exports = User; </pre>	

5.	Setelah di jalankan ulang	 <pre>[{ "id": 1, "name": "Riska Safitri", "email": "riska@mail.com", "password": "123456", "created_at": "2025-11-12T03:25:26.000Z", "updated_at": "2025-11-12T03:25:26.000Z" }, { "id": 2, "name": "Josephine", "email": "josep@mail.com", "password": "abcdef", "created_at": "2025-11-12T03:25:26.000Z", "updated_at": "2025-11-12T03:25:26.000Z" }, { "id": 3, "name": "Moh. Ilham", "email": "ilham@mail.com", "password": "qwerty", "created_at": "2025-11-12T03:25:26.000Z", "updated_at": "2025-11-12T03:25:26.000Z" }]</pre>	
D.	Melakukan test API		

1.	Gunakan browser/postman untuk mendapatkan data getAll users dengan mengunjungi endpoints /api/users/	 <pre>1 [2 { 3 "id": 1, 4 "name": "Riska Safitri", 5 "email": "riska@mail.com", 6 "password": "123456", 7 "created_at": "2025-11-12T03:25:26.000Z", 8 "updated_at": "2025-11-12T03:25:26.000Z" 9 }, 10 { 11 "id": 2, 12 "name": "Josephine", 13 "email": "josep@mail.com", 14 "password": "abcdef", 15 "created_at": "2025-11-12T03:25:26.000Z", 16 "updated_at": "2025-11-12T03:25:26.000Z" 17 }, 18 { 19 "id": 3, 20 "name": "Moh. Ilham", 21 "email": "ilham@mail.com", 22 "password": "qwerty", 23 "created_at": "2025-11-12T03:25:26.000Z", 24 "updated_at": "2025-11-12T03:25:26.000Z" 25 } 26]</pre>	
E.	Lengkapi controllers dan model		
1.	Lengkapi kode yg ada di controllers	 <pre>1 const User = require('../models/user.model'); //memanggil model user 2 3 //GET semua user 4 exports.getAllUsers = (req, res) => { 5 User.getAll((err, results) => { //ambil dari models 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) return res.status(404).json({ message: 'User not found' }); 17 res.json(results[0]); 18 }); 19 }; 20 21 // post user baru 22 exports.createUser = (req, res) => { 23 const data = req.body; 24 User.create(data, (err, result) => { 25 if (err) return res.status(500).json({ error: err.message }); 26 res.status(201).json({ id: result.insertId, ...data }); 27 }); 28 }; 29</pre>	

		<pre>29 30 // put update user 31 exports.updateUser = (req, res) => { 32 const { id } = req.params; 33 const data = req.body; 34 User.update(id, data, (err, result) => { 35 if (err) return res.status(500).json({ error: err.message }); 36 if (result.affectedRows === 0) return res.status(404).json({ message: 'User not found' }); 37 res.json({ message: 'User updated successfully' }); 38 }); 39 }; 40 41 // delete user 42 exports.deleteUser = (req, res) => { 43 const { id } = req.params; 44 User.delete(id, (err, result) => { 45 if (err) return res.status(500).json({ error: err.message }); 46 if (result.affectedRows === 0) return res.status(404).json({ message: 'User not found' }); 47 res.json({ message: 'User deleted successfully' }); 48 }); 49 };</pre>	
2.	Lengkapi kode y gada di model	<pre>models > JS user.model.js > ... 1 const db = require('../models/db.config'); 2 3 //model user (berisikan query database) 4 const User = { 5 getAll: callback => { 6 db.query('SELECT * FROM users', callback); 7 }, 8 9 getById: (id, callback) => { 10 db.query('SELECT * FROM users WHERE id = ?', [id], callback); 11 }, 12 13 create: (data, callback) => { 14 db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback); 15 }, 16 17 update: (id, data, callback) => { 18 db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback); 19 }, 20 21 delete: (id, callback) => { 22 db.query('DELETE FROM users WHERE id = ?', [id], callback); 23 } 24 }; 25 26 module.exports = User;</pre>	
3.	Output get by id	<p>localhost / localhost phpMyAdmin x localhost:8001/api/users/1 x +</p> <p>localhost:8001/api/users/1</p> <p>Import favorites Decoding Your Fina...</p> <p>Pretty-print <input checked="" type="checkbox"/></p> <pre>{ "id": 1, "name": "Riska Safitri", "email": "riska@mail.com", "password": "123456", "created_at": "2025-11-12T03:25:26.000Z", "updated_at": "2025-11-12T03:25:26.000Z" }</pre>	

4.	Output tambah data baru	 <p>POST Get data • praktikum8 +</p> <p>My Collection / Get data</p> <p>POST localhost:8001/api/users</p> <p>Params Authorization Headers (8) Body Scripts Settings</p> <p>none form-data x-www-form-urlencoded raw binary GraphQL JSON</p> <pre>1 { 2 "name": "salsa", 3 "email": "salsa@example.com" 4 }</pre> <p>Body Cookies Headers (7) Test Results (0/1) ↺</p> <p>{ } JSON Preview Visualize</p> <pre>1 { 2 "id": 4, 3 "name": "salsa", 4 "email": "salsa@example.com" 5 }</pre>	
5.	Output update data	 <p>PUT Get data • praktikum8 +</p> <p>My Collection / Get data</p> <p>PUT localhost:8001/api/users/2</p> <p>Params Authorization Headers (8) Body Scripts Settings</p> <p>none form-data x-www-form-urlencoded raw binary GraphQL JSON</p> <pre>1 { 2 "name": "JoeTaslim", 3 "email": "jojo@example.com" 4 }</pre> <p>Body Cookies Headers (7) Test Results (1/1) ↺</p> <p>{ } JSON Preview Visualize</p> <pre>1 { 2 "message": "User updated successfully" 3 }</pre>	
6.	Output hapus data	 <p>DEL Get data • praktikum8 +</p> <p>My Collection / Get data</p> <p>DELETE localhost:8001/api/users/3</p> <p>Params Authorization Headers (8) Body Scripts Settings</p> <p>Body Cookies Headers (7) Test Results (1/1) ↺</p> <p>{ } JSON Preview Visualize</p> <pre>1 { 2 "message": "User deleted successfully" 3 }</pre>	

7.	Output akhir	 <pre>1 [2 { 3 "id": 1, 4 "name": "Riska Safitri", 5 "email": "riska@mail.com", 6 "password": "123456", 7 "created_at": "2025-11-12T03:25:26.000Z", 8 "updated_at": "2025-11-12T03:25:26.000Z" 9 }, 10 { 11 "id": 2, 12 "name": "JoeTaslim", 13 "email": "jojo@example.com", 14 "password": "abcdef", 15 "created_at": "2025-11-12T03:25:26.000Z", 16 "updated_at": "2025-11-12T13:40:47.000Z" 17 }, 18 { 19 "id": 4, 20 "name": "salsa", 21 "email": "salsa@example.com", 22 "password": null, 23 "created_at": "2025-11-12T13:34:17.000Z", 24 "updated_at": "2025-11-12T13:34:17.000Z" 25 } 26]</pre>	
F.	Commit github		
1.	Git init Git add . Git commit -m "first commit" Git branch -M main Git remote Git add origin Git push -u origin main	