

# 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
```

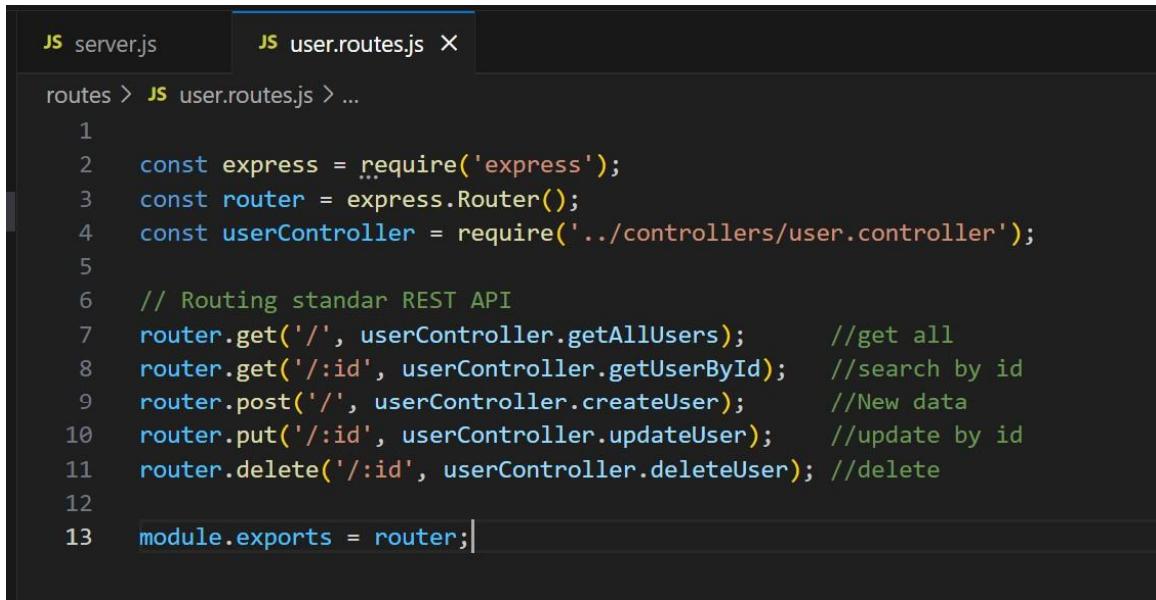
4. Jalankan [server.js](#) dengan mengetik  
Ketik: node [server.js](#)

### 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](#)

```
▽ PRAKTIKUM8
  ▽ controllers
    JS user.controller.js
  ▽ routes
    JS user.routes.js
  {} package.json
  JS server.js
```

3. Tulis kode program di file [user.routes.js](#) seperti pada gambar dibawah ini



The screenshot shows a code editor with two tabs: 'server.js' and 'user.routes.js'. The 'user.routes.js' tab is active, showing the following code:

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



The screenshot shows a code editor with a single file 'user.controller.js' open. The code defines a function 'getAllUsers' that returns all users from a 'User' model.

```
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



The screenshot shows a code editor with a single file 'server.js' open. It includes a code snippet for adding routes to the application.

```
/ 
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', 'riskasafitri@mail.com', '123456'), ('Josephine',  
'josephine@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

```
EXPLORER          ...
└ PRAKTIKUM8
  └ controllers
    └ user.controller.js
  └ models
    └ db.js
  └ routes
    └ user.routes.js
  package.json
  server.js

JS db.js          X
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 a code editor with the following structure:

- EXPLORER** pane on the left:
  - PRAKTIKUM8
    - controllers
    - JS user.controller.js**
    - models
      - JS db.js**
      - JS user.model.js** (highlighted)
    - routes
    - { } package.json
    - JS server.js
- ...**  button
- JS db.js**, **JS user.model.js X**, **JS user.controller.js** tabs at the top right.
- models > JS user.model.js > ...** in the status bar.
- Code Area:**

```

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 `user.model.js` file with the following updated code:

```

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  module.exports = User;
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"  
}
```

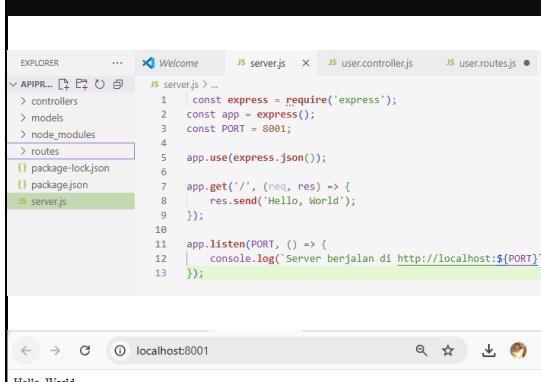
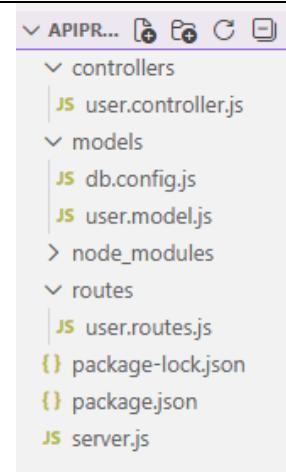
6. Menguji /api/users/3 (Method: DELETE)

## F. Github + Visual Code

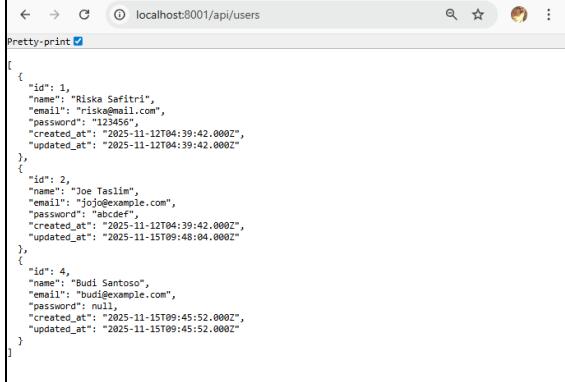
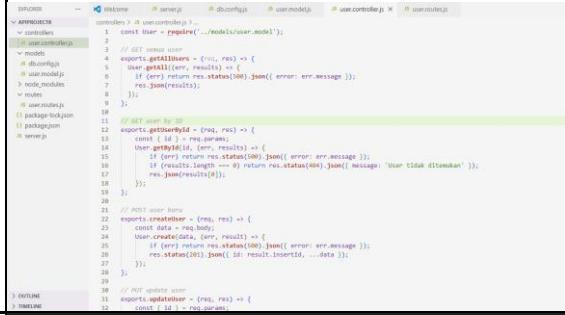
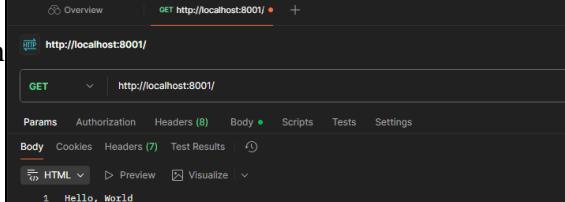
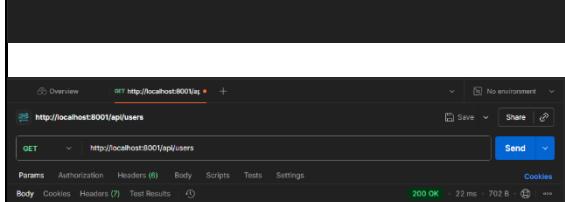
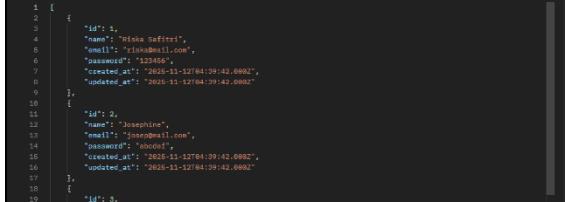
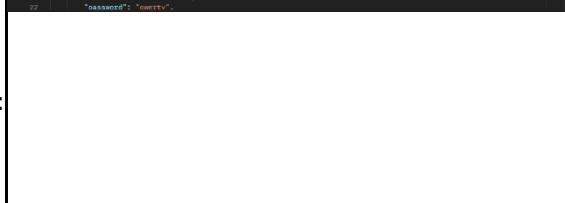
1. Buat proyek di Github dengan nama **Latihan8**

```
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 Penggerjaan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Installasi dan Konfigurasi		
1.	<p>1. Buat sebuah folder proyek API dengan nama APIproject8</p> <p>2. Lakukan seperti pada praktikum 3 Ketik: npm init -y , setelah itu npm install express</p> <p>3. Buat file server.js</p> <p>4. Jalankan server.js dengan mengetik Ketik: node server.js</p>	<pre>D:\APIproject8&gt;npm init -y Wrote to D:\APIproject8\package.json:  {   "name": "apiproject8",   "version": "1.0.0",   "description": "",   "main": "index.js",   "scripts": {     "test": "echo \\"Error: no test specified\\" &amp;&amp; exit 1"   },   "keywords": [],   "author": "",   "license": "ISC" }  D:\APIproject8&gt;npm install express added 68 packages, and audited 69 packages in 6s  16 packages are looking for funding   run `npm fund` for details  found 0 vulnerabilities  D:\APIproject8&gt;</pre> 	
2.	<p>1. Buat folder routes, controllers dan models</p> <p>2. Kemudian didalam folder routes buat sebuah file dengan nama user.routes.js</p> <p>3. Tulis kode program di file user.routes.js</p> <p>4. Buat file di dalam folder controllers dengan nama user.controller.js</p> <p>5. Tulis kode program di dalam file user.controller.js Karena pada controller user tersebut require model bernama User, maka kita siapkan Model user, yang</p>		

berkaitan dengan database.	<p>6. Update file server.js. Kode diatas pada file server.js untuk memberitahu ada routes bernama userRoutes dengan lokasi file di routes/user.routes (tidak perlu ditulis .js)</p>	<pre> APPLI... --&gt; Welcome server.js db.config.js user.model.js user.controller.js user.routes.js  controllers   1 user.controller.js   2 db.config.js   3 user.model.js   4 user.routes.js   5 package-lock.json   6 package.json   7 server.js  node_modules   8 routes     9 user.routes.js  routes   10 user.routes.js   11 package-lock.json   12 package.json   13 server.js    module.exports = router;   </pre> <pre> APPLI... --&gt; Welcome server.js db.config.js user.model.js user.controller.js user.routes.js  controllers   1 user.controller.js   2 db.config.js   3 user.model.js   4 user.routes.js   5 package-lock.json   6 package.json   7 server.js  models   8 user.model.js   9 user.routes.js  node_modules   10 routes     11 user.routes.js  routes   12 user.routes.js   13 package-lock.json   14 package.json   15 server.js    module.exports = router;   </pre> <pre> APPLI... --&gt; Welcome server.js db.config.js user.model.js user.controller.js user.routes.js  controllers   1 user.controller.js   2 db.config.js   3 user.model.js   4 user.routes.js   5 package-lock.json   6 package.json   7 server.js  models   8 user.model.js   9 user.routes.js  node_modules   10 routes     11 user.routes.js  routes   12 user.routes.js   13 package-lock.json   14 package.json   15 server.js    module.exports = router;   </pre>																								
3.	<p>1. Nyalakan mysql service dan buatlah sebuah database dengan nama dbpraktikum8.</p> <p>2. Lalu masukan data dummy ke dalamnya.</p>	<table border="1"> <thead> <tr> <th>ID</th> <th>name</th> <th>email</th> <th>password</th> <th>created_at</th> <th>updated_at</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Rika Sutti</td> <td>riska.sutti@gmail.com</td> <td>123456</td> <td>2025-12-11 13:42:42</td> <td>2025-12-12 11:39:42</td> </tr> <tr> <td>2</td> <td>Joesper</td> <td>joesper@gmail.com</td> <td>123456</td> <td>2025-12-11 13:42:42</td> <td>2025-12-12 11:39:42</td> </tr> <tr> <td>3</td> <td>Mith Item</td> <td>mith.item@gmail.com</td> <td>secret</td> <td>2025-12-11 13:42:42</td> <td>2025-12-12 11:39:42</td> </tr> </tbody> </table>	ID	name	email	password	created_at	updated_at	1	Rika Sutti	riska.sutti@gmail.com	123456	2025-12-11 13:42:42	2025-12-12 11:39:42	2	Joesper	joesper@gmail.com	123456	2025-12-11 13:42:42	2025-12-12 11:39:42	3	Mith Item	mith.item@gmail.com	secret	2025-12-11 13:42:42	2025-12-12 11:39:42
ID	name	email	password	created_at	updated_at																					
1	Rika Sutti	riska.sutti@gmail.com	123456	2025-12-11 13:42:42	2025-12-12 11:39:42																					
2	Joesper	joesper@gmail.com	123456	2025-12-11 13:42:42	2025-12-12 11:39:42																					
3	Mith Item	mith.item@gmail.com	secret	2025-12-11 13:42:42	2025-12-12 11:39:42																					
	<p>3. Jika database sudah terisi data di tabel users, lalu kita persiapkan kembali di express.js</p> <p>4. Install Module mysql2 dengan menggunakan node. Masih di folder project ketik perintah berikut: <code>npm install express mysql2</code></p>	<pre>D:\APIproject8&gt;npm install express mysql2 added 12 packages, and audited 81 packages in 5s  17 packages are looking for funding   run 'npm fund' for details  found 0 vulnerabilities</pre>																								
	<p>5. Kemudian buat sebuah file di dalam folder models, dengan nama db.config.js</p> <p>6. File db.config.js adalah sebagai class connector antara express dan database</p>	<pre> APPLI... --&gt; Welcome server.js db.config.js user.model.js user.controller.js user.routes.js  controllers   1 user.controller.js   2 db.config.js   3 user.model.js   4 user.routes.js   5 package-lock.json   6 package.json   7 server.js  models   8 db.config.js   9 user.model.js  node_modules   10 routes     11 user.routes.js  routes   12 user.routes.js   13 package-lock.json   14 package.json   15 server.js    module.exports = db;   </pre>																								
	<p>7. Buat file lagi untuk model user, di dalam folder models. Dengan nama user.model.js</p> <p>8. Jalankan atau restart ulang node server.js (Pastikan mysql sudah running, user password mysql sudah benar)</p>	<pre> APPLI... --&gt; Welcome server.js db.config.js user.model.js user.controller.js user.routes.js  controllers   1 user.controller.js   2 db.config.js   3 user.model.js   4 user.routes.js   5 package-lock.json   6 package.json   7 server.js  models   8 db.config.js   9 user.model.js  node_modules   10 routes     11 user.routes.js  routes   12 user.routes.js   13 package-lock.json   14 package.json   15 server.js    module.exports = db;   </pre>																								
	<p>9. Gunakan browser/postman untuk mendapatkan data getAll users dengan mengunjungi endpoints /api/users/</p>																									

			
4.	<p>1. Tambahkan class untuk model baru, agar terhubung dengan controller. Ubah pada file user.model.js</p> <p>2. Tambahkan class baru untuk routes yang sudah dipersiapkan lainnya,</p>	 <pre> // Model User (entity query done) const User = () =&gt; {   const db = require('../db/config');   const dbConfig = db();   const User = db(dbConfig);   User.getOne = (id, callback) =&gt; {     User.getOne(id).then(result =&gt; {       if (!result) return callback(null);       else callback(result);     });   };   User.getByEmail = (email, callback) =&gt; {     User.getByEmail(email).then(result =&gt; {       if (!result) return callback(null);       else callback(result);     });   };   User.create = (data, callback) =&gt; {     User.insert(data.name, data.email).then(result =&gt; {       if (!result) return callback(null);       else callback(result);     });   };   User.update = (id, data, callback) =&gt; {     User.update(id, data.name, data.email).then(result =&gt; {       if (!result) return callback(null);       else callback(result);     });   };   User.delete = (id, callback) =&gt; {     User.delete(id).then(result =&gt; {       if (!result) return callback(null);       else callback(result);     });   }; };  module.exports = User; </pre>  <pre> // GET /api/users exports.getUsers = (req, res) =&gt; {   User.getUsers().then(result =&gt; {     if (!result) return res.status(404).json({ message: 'User tidak ditemukan' });     else res.json(result);   }); };  // GET /api/users/:id exports.getUserById = (req, res) =&gt; {   const id = req.params.id;   User.getUserById(id).then(result =&gt; {     if (!result) return res.status(404).json({ message: 'User tidak ditemukan' });     else res.json(result);   }); };  // POST /api/users exports.createUser = (req, res) =&gt; {   User.createUser(req.body).then(result =&gt; {     if (!result) return res.status(400).json({ message: 'User gagal dibuat' });     else res.status(201).json({ id: result.insertId, ...req.body });   }); };  // PUT /api/users/:id exports.updateUser = (req, res) =&gt; {   const id = req.params.id;   User.updateUser(id, req.body).then(result =&gt; {     if (!result) return res.status(404).json({ message: 'User tidak ditemukan' });     else res.json(result);   }); };  // DELETE /api/users/:id exports.deleteUser = (req, res) =&gt; {   const id = req.params.id;   User.deleteUser(id).then(result =&gt; {     if (!result) return res.status(404).json({ message: 'User tidak ditemukan' });     else res.json(result);   }); }; </pre>	
5.	<p>Dengan menggunakan POSTMAN, lakukan pengujian berikut:</p> <ol style="list-style-type: none"> <li>Menguji endpoint /</li> <li>Menguji endpoint /api/users (Method: GET)</li> <li>Menguji endpoint /api/users/1 (Method: GET)</li> <li>Menguji endpoint /api/users (Method: POST) Tambah body -&gt; raw -&gt; JSON { "name": "Budi Santoso", "email": "budi@example.com" }</li> <li>Menguji /api/users/2 (Method: PUT) Masukan Body -&gt; raw -&gt; JSON { "name": "Joe Taslim", "email": "jojo@example.com" }</li> <li>Menguji /api/users/3 (Method: DELETE)</li> </ol>	    	

B.	Github dan Viscode		
1.	<p>1. Buat proyek di Github dengan nama Latihan8</p> <pre>git init git add . git commit -m "first commit" git branch -M main git remote add origin <a href="https://github.com/anastasyafdg/Latihan8.git">https://github.com/anastasyafdg/Latihan8.git</a> git push -u origin main</pre>	<p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS</p> <pre>create mode 100644 package.json create mode 100644 routes/user.routes.js create mode 100644 server.js PS D:\API\project&gt; git branch -M main PS D:\API\project&gt; git remote add origin https://github.com/anastasyafdg/Latihan8.git PS D:\API\project&gt; git push -u origin main Enumerating objects: 963, done. Counting objects: 100% (963/963), done. Delta compression using up to 4 threads Compressing objects: 100% (893/893), done. Writing objects: 100% (963/963), 1.05 MiB   705.00 KiB/s, done. Total 963 (delta 161), reused 0 (delta 0), pack-reused 0 remote: Resolving deltas: 100% (161/161), done. To https://github.com/anastasyafdg/Latihan8.git  [new branch] main -&gt; main branch 'main' set up to track 'origin/main'. PS D:\API\project&gt;</pre>	

Link Github :  
<https://github.com/anastasyafdg/Latihan8.git>

