LAPORAN PRAKTIKUM INTERNET OF THINGS (IOT)

Fakultas Vokasi, Universitas Brawijaya **Praktik Simulasi ESP32 Relay, Button & LED**

*Raka Sanjaya – 233140701111018*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: sanjayaraka722@gmail.com*

**Abstract** (Abstrak)

Eksperimen ini bertujuan untuk mensimulasikan penggunaan ESP32 dalam mengontrol relay dan LED menggunakan tombol sebagai input. Sistem ini dirancang agar saat tombol ditekan, relay akan diaktifkan dan LED menyala, sedangkan saat tombol dilepas, relay kembali nonaktif dan LED mati. Simulasi dilakukan menggunakan platform Wokwi untuk menguji fungsionalitas tanpa memerlukan perangkat keras fisik. Implementasi kode dilakukan di VS Code dengan PlatformIO guna meningkatkan efisiensi pengembangan dan debugging. Hasil simulasi menunjukkan bahwa ESP32 mampu mengontrol relay dengan baik berdasarkan input dari tombol, memberikan dasar bagi implementasi pada perangkat keras nyata.

**Kata kunci:** ESP32, relay, tombol, LED, Wokwi, PlatformIO, simulasi.

1. **Pendahuluan**
2. **Latar Belakang**

Sistem otomatisasi banyak digunakan dalam berbagai bidang, seperti rumah pintar, industri, dan keamanan. Salah satu komponen penting dalam sistem ini adalah relay, yang memungkinkan kontrol perangkat listrik dengan sinyal tegangan rendah dari mikrokontroler seperti ESP32. Untuk memahami prinsip kerja relay dalam sistem otomatisasi, eksperimen ini dilakukan dengan mensimulasikan kendali relay menggunakan tombol dan LED sebagai indikator.

1. **Tujuan eksperimen**

•Mengimplementasikan sistem kendali relay menggunakan ESP32 dan tombol sebagai input.  
•Menganalisis respons ESP32 dalam mengaktifkan dan menonaktifkan relay berdasarkan tombol.  
•Menguji simulasi rangkaian di Wokwi sebelum diterapkan pada perangkat keras nyata.  
•Menggunakan PlatformIO di VS Code untuk pengembangan dan debugging kode.

1. **Metodologi**
2. **Alat dan Bahan**

Alat dan bahan yang digunakan dalam eksperimen ini meliputi:

• ESP32 (simulasi di Wokwi)  
• Modul relay  
• LED sebagai indikator status relay  
• Push button sebagai input kontrol  
• Software Wokwi untuk simulasi  
• VS Code dengan ekstensi PlatformIO  
• Kabel jumper (jika menggunakan perangkat keras nyata)

1. **Langkah Implementasi**
2. Menyiapkan lingkungan simulasi di Wokwi dengan menambahkan Relay Module, Push Button, LED, ke dalam skema rangkaian.

A computer screen shot of a computer

AI-generated content may be incorrect.

1. Menulis program untuk

A screenshot of a computer program

AI-generated content may be incorrect.

1. Membuat file diagram.json di folder root dan menuliskan kode ini

A screen shot of a computer program

AI-generated content may be incorrect.

1. Membuat file wokwi.toml di folder root dan mengetikkan kode ini

[wokwi]

version = 1

firmware = '.pio/build/esp32doit-devkit-v1/firmware.bin'

elf = '.pio/build/esp32doit-devkit-v1/firmware.elf'

1. Buat file platfotmio.ini di root dan mengetikkan seperti ini

A screen shot of a computer program

AI-generated content may be incorrect.

1. Build file main.cpp dan lihat file diagram.jsonnya

A screenshot of a computer

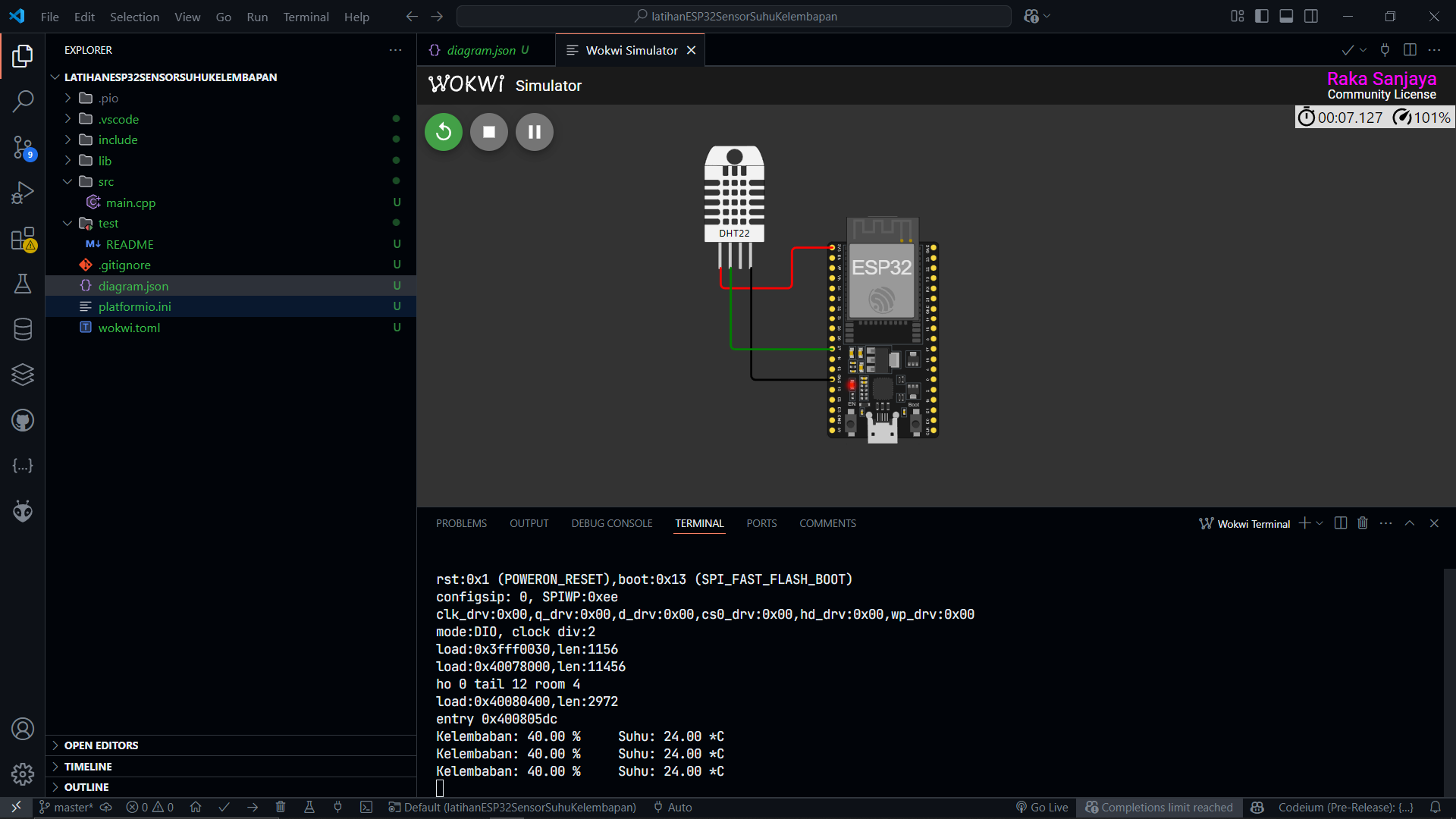
AI-generated content may be incorrect.

1. Anda bisa mengatur suhu dengan klik alat nya dan menggesernya

A computer screen shot of a computer

AI-generated content may be incorrect.

1. **Hasil dan Pembahasan**
   1. **Hasil Eksperimen**



Hasil Eksperimen Setelah melakukan simulasi ESP32 dengan sensor DHT22 menggunakan Wokwi dan PlatformIO, diperoleh hasil sebagai berikut:

* Data suhu dan kelembaban berhasil terbaca dan ditampilkan di serial monitor secara real-time.
* Simulasi di Wokwi memungkinkan perubahan suhu dan kelembaban secara dinamis dengan menggeser alat pengatur nilai sensor.
* Program yang dijalankan di VS Code dengan PlatformIO berhasil dikompilasi tanpa error, dan firmware dapat diunggah ke ESP32 dalam simulasi.
* File konfigurasi wokwi.toml, diagram.json, dan platformio.ini telah dibuat dan berfungsi sebagaimana mestinya.

Dengan demikian, eksperimen ini menunjukkan bahwa simulasi berbasis Wokwi dan pengembangan menggunakan PlatformIO dapat menjadi metode yang efisien dalam menguji sistem berbasis IoT sebelum diterapkan secara fisik.

LAPORAN PRAKTIKUM INTERNET OF THINGS (IOT)

Fakultas Vokasi, Universitas Brawijaya **Praktik Simulasi Sensor Jarak (Ultrasonic)**

*Raka Sanjaya – 233140701111018*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: sanjayaraka722@gmail.com*

**Abstract** (Abstrak)

Sensor ultrasonik banyak digunakan dalam berbagai aplikasi, seperti sistem parkir otomatis, robotika, dan pengukuran jarak. Eksperimen ini bertujuan untuk mensimulasikan kerja sensor ultrasonik menggunakan ESP32 dalam platform simulasi Wokwi. Dengan menggunakan sensor HC-SR04, ESP32 mengukur jarak berdasarkan waktu tempuh gelombang ultrasonik. Eksperimen ini juga mengimplementasikan pemrograman menggunakan PlatformIO di VS Code untuk debugging. Hasil simulasi menunjukkan bahwa sensor dapat mengukur jarak dengan akurasi yang cukup baik dalam lingkungan simulasi.

**Kata kunci:** Sensor ultrasonik, ESP32, HC-SR04, Wokwi, PlatformIO.

1. **Pendahuluan** 
   1. **Latar Belakang**

Dalam era otomatisasi dan Internet of Things (IoT), sensor ultrasonik memiliki peran penting dalam pengukuran jarak tanpa kontak fisik. Sensor HC-SR04 bekerja dengan prinsip pantulan gelombang suara untuk menentukan jarak suatu objek. Pada proyek ini, simulasi sensor ultrasonik dilakukan menggunakan ESP32 untuk memahami cara kerja dan penerapannya dalam aplikasi dunia nyata.

* 1. **Tujuan eksperimen**

•Mengimplementasikan sistem kendali relay menggunakan ESP32 dan tombol sebagai input.  
•Menganalisis respons ESP32 dalam mengaktifkan dan menonaktifkan relay berdasarkan tombol.  
•Menguji simulasi rangkaian di Wokwi sebelum diterapkan pada perangkat keras nyata.  
•Menggunakan PlatformIO di VS Code untuk pengembangan dan debugging kode.

1. **Metodologi**
   1. **Alat dan Bahan**

Alat dan bahan yang digunakan dalam eksperimen ini meliputi:

* ESP32 (simulasi di Wokwi)
* Sensor Ultrasonik HC-SR04
* Kabel jumper (jika menggunakan perangkat keras nyata)
* Software Wokwi untuk simulasi
* VS Code dengan ekstensi PlatformIO untuk pengembangan kode

1. **Langkah Implementasi**
   1. Menghubungkan sensor HC-SR04 ke ESP32 sesuai diagram berikut:
      * + - VCC → 3.3V ESP32
          - Trig → GPIO 5 ESP32
          - Echo → GPIO 18 ESP32
          - GND → GND ESP32

A computer screen shot of a computer

AI-generated content may be incorrect.

* 1. Menulis dan mengunggah kode program untuk membaca jarak menggunakan sensor ultrasonik di folder src/main.cpp.
* #include <Arduino.h>
* const int trigPin = 18;
* const int echoPin = 5;
* //define sound speed in cm/uS
* #define SOUND\_SPEED 0.034
* #define CM\_TO\_INCH 0.393701
* long duration;
* float distanceCm;
* float distanceInch;
* void setup() {
* Serial.begin(115200); // Starts the serial communication
* pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
* pinMode(echoPin, INPUT); // Sets the echoPin as an Input
* }
* void loop() {
* // Clears the trigPin
* digitalWrite(trigPin, LOW);
* delayMicroseconds(2);
* // Sets the trigPin on HIGH state for 10 micro seconds
* digitalWrite(trigPin, HIGH);
* delayMicroseconds(10);
* digitalWrite(trigPin, LOW);
* // Reads the echoPin, returns the sound wave travel time in microseconds
* duration = pulseIn(echoPin, HIGH);
* // Calculate the distance
* distanceCm = duration \* SOUND\_SPEED/2;
* // Convert to inches
* distanceInch = distanceCm \* CM\_TO\_INCH;
* // Prints the distance in the Serial Monitor
* Serial.print("Distance (cm): ");
* Serial.println(distanceCm);
* // Serial.print("Distance (inch): ");
* // Serial.println(distanceInch);
* delay(1000);
* }

**Kode diagram.json**

{

"version": 1,

"author": "Ramdan Hidayat",

"editor": "wokwi",

"parts": [

{ "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },

{ "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -94.5, "left": 149.5, "attrs": {} }

],

"connections": [

[ "esp:TX", "$serialMonitor:RX", "", [] ],

[ "esp:RX", "$serialMonitor:TX", "", [] ],

[ "ultrasonic1:VCC", "esp:5V", "red", [ "v230.4", "h-57.6", "v-86.4" ] ],

[ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],

[ "ultrasonic1:TRIG", "esp:18", "green", [ "v134.4", "h-134.8" ] ],

[ "ultrasonic1:ECHO", "esp:5", "green", [ "v0" ] ]

],

"dependencies": {}

}

**Kode toml.wokwi**

[wokwi]

version = 1

firmware = '.pio\build\esp32doit-devkit-v1\firmware.bin'

elf = '.pio\build\esp32doit-devkit-v1\firmware.elf'

**3. Hasil dan Pembahasan**

**3.1 Hasil Eksperimen**

A computer screen shot of a computer program

AI-generated content may be incorrect.

Setelah simulasi dijalankan, data jarak dapat diamati melalui Serial Monitor. Pengukuran menunjukkan bahwa ESP32 mampu membaca jarak dengan akurasi yang baik dalam lingkungan simulasi.

LAPORAN PRAKTIKUM INTERNET OF THINGS (IOT)

Fakultas Vokasi, Universitas Brawijaya **Praktik Pembuatan API Menggunakan Laravel 11 dan Ngrok**

*Raka Sanjaya – 233140701111018*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: sanjayaraka722@gmail.com*

**Abstract** (Abstrak)

Laporan praktikum ini menjelaskan proses pembuatan API untuk pengelolaan data sensor dalam proyek Internet of Things (IoT) menggunakan framework Laravel 11 dan layanan tunneling Ngrok. Eksperimen ini bertujuan untuk menghubungkan perangkat IoT dengan sistem backend secara real-time, sehingga data sensor dapat dikirim, disimpan, dan diakses melalui jaringan internet. Pada praktikum ini, dilakukan pembuatan database, pembuatan model, migrasi, pembuatan resource, controller, serta pengaturan route API. Selain itu, API diuji menggunakan aplikasi Postman dan kemudian di-online-kan menggunakan Ngrok untuk memastikan akses publik melalui URL yang diberikan.

**Kata kunci:** IoT, Laravel 11, API, Ngrok, Postman, Sensor, Web Service

* 1. **Pendahuluan**
  2. **Latar Belakang**

Internet of Things (IoT) merupakan konsep penghubungan berbagai perangkat melalui internet agar dapat saling bertukar data dan informasi. Dalam implementasinya, API (Application Programming Interface) berperan penting sebagai jembatan komunikasi antara perangkat dan server backend. Laravel, sebagai salah satu framework PHP modern, menyediakan kemudahan dalam pembuatan API dengan struktur MVC yang terorganisir dan fitur-fitur canggih seperti Eloquent ORM dan sistem routing. Penggunaan Ngrok memungkinkan API yang dibuat di lingkungan lokal diakses secara publik, sehingga dapat diuji secara langsung pada perangkat IoT atau simulasi perangkat seperti WOKWI ESP32.

* 1. **Tujuan eksperimen**

Adapun tujuan eksperimen ini adalah:

* Membangun API menggunakan Laravel 11 untuk mengelola data sensor.
* Mengimplementasikan migrasi database, model, resource, dan controller sesuai standar Laravel.
* Melakukan pengujian API menggunakan Postman untuk operasi CRUD (Create, Read, Update, Delete).
* Meng-online-kan API menggunakan Ngrok agar dapat diakses secara publik melalui internet.

**2. Metodologi**

* 1. **Alat dan Bahan**
* Software & Tools:
* XAMPP/LAMP (untuk menjalankan PHP dan MySQL)
* Laravel 11
* Ngrok
* Postman
* phpMyAdmin
* Hardware:
* Komputer/laptop
* Perangkat IoT (atau simulator seperti WOKWI ESP32)

**3. Langkah Implementasi**

1. Buat database melalui phpMyAdmin dengan nama **iot\_25**.
2. Install Laravel 11 dan konfigurasi filenya
   * Buat model **TransaksiSensor** dan isi file ini

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class TransaksiSensor extends Model

{

use HasFactory;

protected $table = 'transaksi\_sensor';

protected $fillable = [

'nama\_sensor',

'nilai1',

'nilai2',

];

protected $casts = [];

}

* + Buat migrasi

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

public function up(): void

{

Schema::create('transaksi\_sensor', function (Blueprint $table) {

$table->id('id')->startingValue(1); // Menetapkan AUTO\_INCREMENT dimulai dari 1

$table->string('nama\_sensor', 255); // varchar(255)

$table->integer('nilai1', false)->length(255); // int(255)

$table->integer('nilai2', false)->length(255); // int(255)

$table->timestamps(); // Menambahkan created\_at dan updated\_at

});

}

public function down(): void

{

Schema::dropIfExists('transaksi\_sensors');

}

};

* + Buat resource:

<?php

namespace App\Http\Resources;

use Illuminate\Http\Request;

use Illuminate\Http\Resources\Json\JsonResource;

class TransaksiSensorResource extends JsonResource

{

public function toArray($request)

{

return [

'id' => $this->id,

'nama\_sensor' => $this->nama\_sensor,

'nilai1' => $this->nilai1,

'nilai2' => $this->nilai2,

];

}

}

* + Buat API controller

<?php

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;

use App\Models\TransaksiSensor;

use App\Http\Controllers\Controller;

use App\Http\Resources\TransaksiSensorResource;

class TransaksiSensorController extends Controller

{

public function index()

{

$transaksiSensors = TransaksiSensor::latest()->paginate(5);

return TransaksiSensorResource::collection($transaksiSensors);

}

public function store(Request $request)

{

$validatedData = $request->validate([

'nama\_sensor' => 'required|string|max:255',

'nilai1' => 'required|integer',

'nilai2' => 'required|integer',

]);

$transaksiSensor = TransaksiSensor::create($validatedData);

return new TransaksiSensorResource($transaksiSensor);

}

public function show($id)

{

$transaksiSensor = TransaksiSensor::findOrFail($id);

return new TransaksiSensorResource($transaksiSensor);

}

public function update(Request $request, $id)

{

$validatedData = $request->validate([

'nama\_sensor' => 'required|string|max:255',

'nilai1' => 'required|integer',

'nilai2' => 'required|integer',

]);

$transaksiSensor = TransaksiSensor::findOrFail($id);

$transaksiSensor->update($validatedData);

return new TransaksiSensorResource($transaksiSensor);

}

public function destroy($id)

{

$transaksiSensor = TransaksiSensor::findOrFail($id);

$transaksiSensor->delete();

return response()->json(['message' => 'Deleted successfully'], 204);

}

}

* + Buat route khusus API

<?php

use Illuminate\Auth\Middleware\Authenticate;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

Route::get('/user', function (Request $request) {

return $request->user();

})->middleware(Authenticate::using('sanctum'));

Route::apiResource('/posts', App\Http\Controllers\Api\TransaksiSensorController::class);

1. **Pengujian API dengan Postman**
   1. **Pengujian POST**
   * Ubah method menjadi POST.
   * Pada bagian **Headers**, atur konten tipe (misalnya, Content-Type: application/json).
   * Pada bagian **Body**, masukkan parameter JSON sesuai validasi, misalnya

{

"nama\_sensor": "Sensor C",

"nilai1": 120,

"nilai2": 240

}

* + Klik **SEND** dan verifikasi bahwa data berhasil tersimpan pada database melalui phpMyAdmin.
  1. **Pengujian GET**
  + Buka aplikasi Postman dan masukkan URL: **http://127.0.0.1:8000/api/posts** atau [**http://localhost:8000/api/posts**](http://localhost:8000/api/posts)
  + Pilih method **GET** dan klik **SEND**.
  + Respon yang diharapkan berupa data JSON dengan format paginasi, misalnya

A screenshot of a computer

AI-generated content may be incorrect.

1. **Meng-online-kan API dengan Ngrok**
   * 1. **Instalasi Ngrok**

* Daftar dan download Ngrok melalui <https://dashboard.ngrok.com/signup>.
* Ekstrak file Ngrok sesuai dengan sistem operasi Anda.
  + 1. **Menjalankan Ngrok**
* Buka Command Prompt (atau Terminal) pada direktori Ngrok yang telah diekstrak.
* Jalankan perintah untuk menghubungkan ke akun Ngrok jika diperlukan (dengan token akun) dan kemudian jalankan:

ngrok http <http://localhost:8000>

* Ngrok akan memberikan URL publik (misalnya, <https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.ngrok-free.app>).
  + 1. **Pengujian API secara Publik:**
* Gunakan URL publik tersebut ditambah endpoint API. Contoh

<https://e521-2405-8740-6315-3520-2099-2415-5077-c12f.ngrok-free.app/api/posts>

* Lakukan pengujian GET dan POST melalui Postman dengan URL Ngrok untuk memastikan API dapat diakses dari luar jaringan lokal.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.