SOURCE CODE

1. Source Code Arduino IDE

```
#include <ESP32Firebase.h>


// Konfigurasi WiFi
#define _SSID "????"
#define _PASSWORD "entahlah"


// Konfigurasi Firebase
#define REFERENCE_URL "https://emissionez-19b91-default-rtdb.asia-southeast1.firebasedatabase.app/"


Firebase firebase(REFERENCE_URL);


#define mq7Pin 35   // Pin untuk MQ-7
#define mq9Pin 32   // Pin untuk MQ-9
#define mq135Pin 33 // Pin untuk MQ-135
#define dustAnalogPin 34 // Pin analog untuk GP2Y1010
#define dustLEDPin 21 // Pin digital untuk mengontrol LED pada GP2Y1010
#define ledPin 18   // Pin untuk LED
#define buzzerPin 19 // Pin untuk buzzer


int mq7Value = 0;
int mq9Value = 0;
int mq135Value = 0;
int dustValue = 0;


// Faktor konversi dari ppm ke gram/km untuk setiap sensor
const float coConversionFactor = 0.001;  // Faktor konversi untuk CO (MQ-7)
const float hcConversionFactor = 0.001;  // Faktor konversi untuk HC (MQ-9)
```

```
const float noxConversionFactor = 0.001; // Faktor konversi untuk NOx (MQ-135)


unsigned long startTime;
const unsigned long duration = 30000; // 30 detik dalam milidetik


float maxCoConcentration = 0;
float maxHcConcentration = 0;
float maxNoxConcentration = 0;
float maxDustConcentration = 0; // Untuk menyimpan nilai PM2.5 maksimum


void setup() {
  Serial.begin(115200);
  Serial.println("MQ Sensor Reading Program");

  // Koneksi ke WiFi
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(1000);

  Serial.print("Connecting to WiFi: ");
  Serial.println(_SSID);
  WiFi.begin(_SSID, _PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected");

  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

```
  // Set pin mode untuk LED dan buzzer
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(dustLEDPin, OUTPUT); // Set pin mode untuk LED GP2Y1010

  startTime = millis(); // Catat waktu mulai

  // Nyalakan LED
  digitalWrite(ledPin, HIGH);
}

void loop() {
  if (millis() - startTime < duration) {
    // Baca nilai dari sensor
    mq7Value = analogRead(mq7Pin);
    mq9Value = analogRead(mq9Pin);
    mq135Value = analogRead(mq135Pin);

    // Baca nilai dari sensor debu GP2Y1010
    dustValue = readDustSensor();

    float coConcentration = calculateGasConcentration(mq7Value,
"MQ-7") * coConversionFactor;
    float hcConcentration = calculateGasConcentration(mq9Value,
"MQ-9") * hcConversionFactor;
    float noxConcentration = calculateGasConcentration(mq135Value,
"MQ-135") * noxConversionFactor;
    float dustConcentration =
calculateDustConcentration(dustValue); // Hitung konsentrasi debu

    // Update nilai maksimum
    if (coConcentration > maxCoConcentration) {
      maxCoConcentration = coConcentration;
    }
```

```
    if (hcConcentration > maxHcConcentration) {
      maxHcConcentration = hcConcentration;
    }
    if (noxConcentration > maxNoxConcentration) {
      maxNoxConcentration = noxConcentration;
    }
    if (dustConcentration > maxDustConcentration) {
      maxDustConcentration = dustConcentration;
    }

    // Cek apakah kadar gas melebihi threshold
    if (coConcentration >= 5.5 || hcConcentration >= 1.0 ||
noxConcentration >= 0.3) {
      // Nyalakan buzzer dengan interval 1 detik high dan 1 detik
low
      digitalWrite(buzzerPin, HIGH);
      delay(1000);
      digitalWrite(buzzerPin, LOW);
      delay(1000);
    }

    // Tampilkan data di Serial Monitor
    Serial.print("CO (MQ-7): ");
    Serial.print(coConcentration, 4); // Tampilkan dengan 4 angka
desimal
    Serial.print(" g/km\t");

    Serial.print("HC (MQ-9): ");
    Serial.print(hcConcentration, 4); // Tampilkan dengan 4 angka
desimal
    Serial.print(" g/km\t");

    Serial.print("NOx (MQ-135): ");
    Serial.print(noxConcentration, 4); // Tampilkan dengan 4 angka
desimal
```

```
    Serial.print(" g/km\t");


    Serial.print("PM2.5 (GP2Y1010): ");
    Serial.print(dustConcentration, 4); // Tampilkan dengan 4
angka desimal
    Serial.println(" ug/m3");


    delay(1000);
  } else {
    // Tampilkan nilai maksimum setelah 30 detik
    Serial.println("Data pengukuran selama 30 detik telah
selesai.");
    Serial.print("Nilai maksimum CO (MQ-7): ");
    Serial.print(maxCoConcentration, 4);
    Serial.println(" g/km");


    Serial.print("Nilai maksimum HC (MQ-9): ");
    Serial.print(maxHcConcentration, 4);
    Serial.println(" g/km");


    Serial.print("Nilai maksimum NOx (MQ-135): ");
    Serial.print(maxNoxConcentration, 4);
    Serial.println(" g/km");


    Serial.print("Nilai maksimum PM2.5 (GP2Y1010): ");
    Serial.print(maxDustConcentration, 4);
    Serial.println(" ug/m3");


    // Kirim data ke Firebase
    if (firebase.setFloat("kadar/CO", maxCoConcentration) &&
        firebase.setFloat("kadar/HC", maxHcConcentration) &&
        firebase.setFloat("kadar/NOX", maxNoxConcentration) &&
        firebase.setFloat("kadar/PM25", maxDustConcentration)) {
      Serial.println("Data berhasil dikirim ke Firebase");
```

```
    } else {
      Serial.println("Gagal mengirim data ke Firebase");
    }

    // Matikan LED
    digitalWrite(ledPin, LOW);

    while (true) {
      // Hentikan loop dengan menghentikan eksekusi
    }
  }
}

float calculateGasConcentration(int sensorValue, String
sensorType) {
  float factor = 0;
  if (sensorType == "MQ-7") {
    // Gunakan faktor kalibrasi khusus untuk MQ-7 (CO)
    factor = 0.1;
  } else if (sensorType == "MQ-9") {
    // Gunakan faktor kalibrasi khusus untuk MQ-9 (HC)
    factor = 0.1;
  } else if (sensorType == "MQ-135") {
    // Gunakan faktor kalibrasi khusus untuk MQ-135 (NOx)
    factor = 0.1;
  }
  return sensorValue * factor;
}

int readDustSensor() {
  digitalWrite(dustLEDPin, LOW); // Nyalakan LED
  delayMicroseconds(280);
  int dustVal = analogRead(dustAnalogPin); // Baca nilai analog
  delayMicroseconds(40);
```

```
  digitalWrite(dustLEDPin, HIGH); // Matikan LED
  delayMicroseconds(9680); // Tunggu sisa waktu dari 10ms
  return dustVal;
}

float calculateDustConcentration(int dustValue) {
  float voltage = dustValue * (5.0 / 1024.0); // Konversi nilai analog ke tegangan
  float dustDensity = 0.17 * voltage - 0.1; // Rumus konversi tegangan ke konsentrasi debu (mg/m3)
  return dustDensity * 1000; // Konversi mg/m3 ke ug/m3
}
```

2. Source Code Back End Dengan Node.js

2.1 Function Koneksi Ke Firebase

```
const admin = require('firebase-admin');

const serviceAccount = require('../key/emissionez-19b91-firebase-adminsdk-mixa3-5ca81b4a7e.json');

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: 'https://emissionez-19b91-default-rtdb.asia-southeast1.firebasedatabase.app//'
});

const db = admin.database();

db.ref('.info/connected').on('value', (snap) => {
  if (snap.val() === true) {
    console.log('Connected to the database');
  } else {
    console.error('Error connecting to database');
  }
});
```

2.2 Function Login Teknisi

```
const login = async (req, res) => {
  const { username, password } = req.body;
  if (!username) {
      return res.status(400).json({
          status: 'Error',
          message: 'Gagal masuk ke aplikasi, username
diperlukan!',
      });
  }
  if (!password) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Gagal masuk ke aplikasi, password
diperlukan!',
      });
  }
  try {
    const userRef = db.ref('users');

userRef.orderByChild('username').equalTo(username).once('value',
(snapshot) => {
        const userData = snapshot.val();
        if (!userData) {
            return res.status(404).json({ message: 'Username tidak
ditemukan!' });
        }
        const userId = Object.keys(userData)[0];
        const user = userData[userId];

        const auth = bcrypt.compareSync(password, user.password);
        if (!auth) {
            return res.status(404).json({ message: 'Password
salah!' });
```

```
        }
        const token = createToken(userId);
        res.cookie('jwt', token, { httpOnly: false, maxAge:
maxExpire * 1000 });
        return res.status(200).json({
            message: 'Logged in!',
            user_id: userId,
            name: user.name, // Mengambil nama dari data user
            token // Menambahkan token ke dalam respons
        });
    }, (error) => {
        console.error('Error reading user data:', error);
        return res.status(500).json({
            status: 'error',
            message: 'Terjadi kesalahan saat login',
        });
    });
  } catch (error) {
      console.error('Error during login:', error);
      return res.status(500).json({
          status: 'error',
          message: 'Terjadi kesalahan saat login',
      });
  }
};
```

2.3 Function Login Admin

```
const loginAdmin = async (req, res) => {
  const { username, password } = req.body;

  if (!username) {
      return res.status(400).json({
          status: 'Error',
```

```
          message: 'Gagal masuk ke aplikasi, username
diperlukan!',
      });
  }


  if (!password) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Gagal masuk ke aplikasi, password
diperlukan!',
      });
  }


  try {
    const adminRef = db.ref('admins');

adminRef.orderByChild('username').equalTo(username).once('value',
(snapshot) => {
        const adminData = snapshot.val();
        if (!adminData) {
            return res.status(404).json({ message: 'Username tidak
ditemukan!' });
        }

        const adminId = Object.keys(adminData)[0];
        const admin = adminData[adminId];

        const auth = bcrypt.compareSync(password, admin.password);
        if (!auth) {
            return res.status(404).json({ message: 'Password
salah!' });
        }

        const token = createTokenAdmin(adminId);
        res.cookie('jwt', token, { httpOnly: false, maxAge:
maxExpire * 1000 });
```

```
        return res.status(200).json({
            message: 'Logged in!',
            admin_id: adminId,
            name: admin.name, // Mengambil nama dari data admin
            token // Menambahkan token ke dalam respons
        });
    }, (error) => {
        console.error('Error reading admin data:', error);
        return res.status(500).json({
            status: 'error',
            message: 'Terjadi kesalahan saat login',
        });
    });

  } catch (error) {
      console.error('Error during login:', error);
      return res.status(500).json({
          status: 'error',
          message: 'Terjadi kesalahan saat login',
      });
  }
};

Function Get Kadar Gas
const getKadarGas = (req, res) => {
  db.ref('kadar').once('value', (snapshot) => {
    const data = snapshot.val();
    if (!data) {
      res.status(404).json({
        status: 'error',
        message: 'Tidak ditemukan data Kadar Gas',
      });
```

```
      return;
    }
    res.status(200).json({
      status: 'success',
      data,
    });
  }, (error) => {
    console.error('Error reading data:', error);
    res.status(500).json({
      status: 'error',
      message: 'Terjadi kesalahan saat mengambil data Kadar Gas',
    });
  });
};
```

2.4 Function Get GPT Response

```
const getGPTResponse = async (req, res) => {
  const {
    namaPengendara,
    merekKendaraan,
    usiaMesin,
    jenisBensin,
    terakhirService,
    kadarCO,
    kadarHC,
    kadarNOX,
    kadarPartikulat,
    pertanyaan
  } = req.body;


  const currentTime = new Date(); // Mendapatkan waktu saat ini
  const formattedTime = currentTime.toISOString(); // Mengonversi
waktu ke dalam format string ISO
```

```
  const prompt = `Merek kendaraan: ${merekKendaraan}, Usia mesin:
${usiaMesin}, Jenis bensin: ${jenisBensin}, Terakhir service:
${terakhirService}, Kadar CO: ${kadarCO}, Kadar HC: ${kadarHC},
Kadar NOX: ${kadarNOX}, Kadar Partikulat: ${kadarPartikulat}.
${pertanyaan}`;

  console.log('Mengirim prompt ke OpenAI:', prompt);


  try {

    // Mengirim prompt ke OpenAI untuk mendapatkan respons

    const response = await
axios.post('https://api.openai.com/v1/chat/completions', {

      model: 'gpt-3.5-turbo',

      messages: [

        { role: 'system', content: 'Kamu adalah seorang
profesional mekanik bengkel kendaraan bermotor. Jawablah dalam
bahasa Indonesia dengan menjelaskan bagian spesifik untuk
mengatasi masalah yang ada. jelaskan dengan to the point dengan
kurang dari 3 kalimat dan hilangkan intro. jika dibutuhkan di
service bagian pembakaran jelaskan komponen apa yang harus di
perbaiki atau diberikan perhatian' },

        { role: 'user', content: prompt }

      ],

      max_tokens: 100,

      temperature: 0.7,

      top_p: 0.9,

      frequency_penalty: 0.5,

      presence_penalty: 0.5

    }, {

      headers: {

        'Content-Type': 'application/json',

        'Authorization': `Bearer ${process.env.OPENAI_API_KEY}`

      }

    });


    console.log('Menerima Response dari OpenAI:', response.data);


    // Menambahkan data ke dalam riwayat
```

```
    const newData = {
      namaPengendara,
      merekKendaraan,
      usiaMesin,
      jenisBensin,
      terakhirService,
      kadarCO,
      kadarHC,
      kadarNOX,
      kadarPartikulat,
      response: response.data.choices[0].message.content.trim(),
      waktu: formattedTime // Menyimpan waktu saat ini
    };

    db.ref('history').push(newData, (error) => {
      if (error) {
        console.error('Error menambahkan data ke riwayat:',
error);
        return res.status(500).json({
          status: 'error',
          message: 'Terjadi kesalahan saat menambahkan data ke
riwayat',
        });
      }
      res.status(201).json({
        status: 'success',
        message: 'Data telah ditambahkan ke riwayat',
        data: newData,
      });
    });
  } catch (error) {
    console.error('Error mengambil respons GPT:', error.response ?
error.response.data : error.message);
    res.status(500).json({
```

```
      status: 'error',
      message: error.response ? error.response.data : 'Terjadi
kesalahan saat mengambil respons GPT',
    });
  }
};
```

2.5 Function Get History

```
const getAllHistory = (req, res) => {
  db.ref('history').once('value', (snapshot) => {
    const historyData = snapshot.val();
    if (!historyData) {
      return res.status(404).json({
        status: 'error',
        message: 'Tidak ditemukan data riwayat',
      });
    }
    const historyArray = Object.keys(historyData).map((key) => ({
      id: key,
      ...historyData[key]
    }));
    res.status(200).json({
      status: 'success',
      data: historyArray,
    });
  }, (error) => {
    console.error('Error membaca data riwayat:', error);
    res.status(500).json({
      status: 'error',
      message: 'Terjadi kesalahan saat mengambil data riwayat',
    });
  });
};
```

2.6 Function Create User

```
const createUser = async (req, res) => {
  const { username, password, name } = req.body;

  if (!username || !password || !name) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Username, password, dan nama diperlukan!',
      });
  }

  if (username.length < 6) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Panjang username harus 6 karakter atau
lebih!',
      });
  }

  if (password.length < 6) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Panjang password harus 6 karakter atau
lebih!',
      });
  }

  try {
      const hashedPassword = await bcrypt.hash(password, 10);
      const newUserRef = db.ref('users').push();
      newUserRef.set({
          username,
          password: hashedPassword,
```

```
            name
        }, (error) => {
            if (error) {
                console.error('Error adding new user:', error);
                return res.status(500).json({
                    status: 'error',
                    message: 'Terjadi kesalahan saat menambahkan
pengguna baru',
                });
            } else {
                console.log('User added successfully');
                return res.status(201).json({
                    status: 'success',
                    message: 'Pengguna berhasil ditambahkan',
                    user_id: newUserRef.key,
                    username,
                    name
                });
            }
        });
    } catch (error) {
        console.error('Error during user creation:', error);
        return res.status(500).json({
            status: 'error',
            message: 'Terjadi kesalahan saat menambahkan pengguna
baru',
        });
    }
};
```

2.7 Function Get Users

```
const getAllUsers = (req, res) => {
  db.ref('users').once('value', (snapshot) => {
    const usersData = snapshot.val();
```

```
    if (!usersData) {
      return res.status(404).json({
        status: 'error',
        message: 'Tidak ditemukan data pengguna',
      });
    }
    const usersArray = Object.keys(usersData).map((key) => ({
      id: key,
      username: usersData[key].username,
      name: usersData[key].name,
    }));
    res.status(200).json({
      status: 'success',
      data: usersArray,
    });
  }, (error) => {
    console.error('Error reading users data:', error);
    res.status(500).json({
      status: 'error',
      message: 'Terjadi kesalahan saat mengambil data pengguna',
    });
  });
};
```

2.8 Function Edit User

```
const editUserById = async (req, res) => {
  const userId = req.params.id;
  const { username, password, name } = req.body;

  if (!username || !password || !name) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Username, password, dan nama diperlukan!',
```

```
      });
  }


  if (username.length < 6) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Panjang username harus 6 karakter atau
lebih!',
      });
  }


  if (password.length < 6) {
      return res.status(400).json({
          status: 'Gagal',
          message: 'Panjang password harus 6 karakter atau
lebih!',
      });
  }


  try {
      const hashedPassword = await bcrypt.hash(password, 10);
      db.ref(`users/${userId}`).update({
          username,
          password: hashedPassword,
          name
      }, (error) => {
          if (error) {
              console.error('Error updating user data:', error);
              return res.status(500).json({
                  status: 'error',
                  message: 'Terjadi kesalahan saat mengupdate data
pengguna',
              });
          }
```

```
            res.status(200).json({
                status: 'success',
                message: 'Data pengguna berhasil diperbarui',
                user_id: userId,
                username,
                name
            });
        });
    } catch (error) {
        console.error('Error during user update:', error);
        return res.status(500).json({
            status: 'error',
            message: 'Terjadi kesalahan saat mengupdate data
pengguna',
        });
    }
};
```

2.9 Function Delete User

```
const deleteUserById = (req, res) => {
  const userId = req.params.id;

  db.ref(`users/${userId}`).remove((error) => {
    if (error) {
      console.error('Error deleting user:', error);
      return res.status(500).json({
        status: 'error',
        message: 'Terjadi kesalahan saat menghapus pengguna',
      });
    }
    res.status(200).json({
      status: 'success',
      message: 'Pengguna berhasil dihapus',
```

```
      user_id: userId,
    });
  });
};
```

2.10 Function Auth Member

```
const requireAuthMember = (req, res, next) => {
    const token = req.headers['authorization'];

    if (!token) {
        return res.status(400).json({ message: 'Token tidak
terdeteksi, harap login terlebih dahulu!' });
    }

    // Check JWT exist & is verified
    jwt.verify(token, process.env.JWT_SECRET, (err, decodedToken)
=> {
        if (err) {
            return res.status(400).json({ message: 'Anda tidak
memiliki hak untuk mengakses request ini!' });
        }
        // console.log(decodedToken);
        next();
    });
};
```

2.11 Function Auth Admin

```
const requireAuthAdmin = (req, res, next) => {
    const token = req.headers['authorization'];

    if (!token) {
        return res.status(400).json({ message: 'Token tidak
terdeteksi, harap login terlebih dahulu!' });
    }
```

```
    // Check JWT exist & is verified
    jwt.verify(token, process.env.JWT_SECRET_ADMIN, (err,
decodedToken) => {
        if (err) {
            return res.status(400).json({ message: 'Request ini
hanya bisa diakses oleh admin!' });
        }
        // console.log(decodedToken);
        next();
    });
};
```

3. Source Code Front End Mobile Dengan Kotlin
   3.1 Class Login Teknisi

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding : ActivityMainBinding
    private lateinit var mainViewModel: MainViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        supportActionBar?.hide()
        mainViewModel =
ViewModelProvider(this).get(MainViewModel::class.java)
        var token = ""
        binding.btnLogin.setOnClickListener{
            val username = binding.edtUsername.text.toString()
            val password = binding.edtPassword.text.toString()
                val client =
APIConfig.getAPIService().postLogin(username, password)
                client.enqueue(object : Callback<ResponseLogin> {
```

```
                    override fun onResponse(
                        call: Call<ResponseLogin>,
                        response: Response<ResponseLogin>
                    ) {
                        if (response.isSuccessful) {
                            token =
response.body()?.token.toString()
                            val sharedPreferences =
getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
                            val editor = sharedPreferences.edit()
                            editor.putString("token", token)
                            editor.apply()

startActivity(Intent(this@MainActivity,
DashboardActivity::class.java))
                        } else {
                        }
                    }
                    override fun onFailure(call:
Call<ResponseLogin>, t: Throwable) {
                        println("GAGAL KENAPA INI?")
                    }
                })
        }
    }
}
```

3.2 Class Login Admin

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding : ActivityMainBinding
    private lateinit var mainViewModel: MainViewModel



    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        supportActionBar?.hide()
        mainViewModel =
ViewModelProvider(this).get(MainViewModel::class.java)
        var token = ""
        binding.btnLogin.setOnClickListener{
            val username = binding.edtUsername.text.toString()
            val password = binding.edtPassword.text.toString()
                val client =
APIConfig.getAPIService().postLogin(username, password)
                client.enqueue(object :
Callback<ResponseLoginAdmin> {
                    override fun onResponse(
                        call: Call<ResponseLoginAdmin>,
                        response: Response<ResponseLoginAdmin>
                    ) {
                        if (response.isSuccessful) {
                            token =
response.body()?.token.toString()
                            val sharedPreferences =
getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
                            val editor = sharedPreferences.edit()
                            editor.putString("token", token)
                            editor.apply()

startActivity(Intent(this@MainActivity,
DashboardActivity::class.java))
                        } else {
                        }
                    }
                    override fun onFailure(call:
Call<ResponseLoginAdmin>, t: Throwable) {
                        println("GAGAL KENAPA INI?")
                    }
```

```
                })
            }
        }
}
```

3.3 Function Get Kadar Gas

```
override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentDashboardBinding.inflate(inflater,
container, false)
        val root: View = binding.root


        dashboardViewModel =
ViewModelProvider(this).get(DashboardViewModel::class.java)


        setupViewModel()


        binding.btnGetData.setOnClickListener{
            val sharedPreferences =
requireContext().getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE)
            val token = sharedPreferences.getString("token", null)


            val client =
APIConfig.getAPIService().getKadarGas(token.toString())
            client.enqueue(object : Callback<ResponseKadar> {
                @SuppressLint("SetTextI18n")
                override fun onResponse(
                    call: Call<ResponseKadar>,
                    response: Response<ResponseKadar>
                ) {
                    if (response.isSuccessful) {
```

```
                        binding.edtKadarCO.text =
response.body()!!.data!!.cO.toString()

                        binding.edtKadarHC.text =
response.body()!!.data!!.hC.toString() + " g/km"

                        binding.edtKadarNox.text =
response.body()!!.data!!.nOX.toString() + " g/km"

                        binding.edtKadarPartikulat.text =
response.body()!!.data!!.pM25.toString() + " ug/m3"

                    } else {

                    }

                }

                override fun onFailure(call: Call<ResponseKadar>,
t: Throwable) {

                }

            })

        }
```

3.4 Function Post GPT Response

```
binding.button.setOnClickListener{

            val sharedPreferences =
requireContext().getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE)

            val token = sharedPreferences.getString("token", null)

            println(token)

            val namaPengendara =
binding.edtNamaPengendara.text.toString()

            val merekKendaraan =
binding.edtMerekKendaraan.text.toString()

            val usiaMesin = binding.edtUsiaMesin.text.toString()

            val jenisBensin =
binding.edtJenisBensin.text.toString()

            val terakhirService =
binding.edtTerakhirService.text.toString()

            val kadarCO = binding.edtKadarCO.text.toString()

            val kadarHC = binding.edtKadarHC.text.toString()

            val kadarNOX = binding.edtKadarNox.text.toString()
```

```
            val kadarPartikulat =
binding.edtKadarPartikulat.text.toString()

            val pertanyaan = "Jika kendaraan bermotor memiliki
data seperti itu bagian mesin apa yang harus banget di service?()"


            val service =


APIConfig.getAPIService().postEmisi(token.toString(),
namaPengendara, merekKendaraan, usiaMesin, jenisBensin,
terakhirService, kadarCO,

                    kadarHC, kadarNOX, kadarPartikulat,
pertanyaan)

                service.enqueue(object : Callback<ResponseEmisi> {

                override fun onResponse(

                    call: Call<ResponseEmisi>,

                    response: Response<ResponseEmisi>

                ) {

                    if (response.isSuccessful) {

                        val responseBody = response.body()

                        if (responseBody != null) {

                            Toast.makeText(

                                requireActivity(),

                                responseBody.message,

                                Toast.LENGTH_SHORT

                            ).show()


                            binding.edtNamaPengendara.text.clear()

                            binding.edtMerekKendaraan.text.clear()

                            binding.edtUsiaMesin.text.clear()

                            binding.edtJenisBensin.text.clear()


binding.edtTerakhirService.text.clear()


                            val tesEmisi = Emisi(namaPengendara,
merekKendaraan, usiaMesin, jenisBensin, terakhirService, kadarCO,
```

```
                                kadarHC, kadarNOX,
kadarPartikulat, pertanyaan,
responseBody.data!!.response.toString())

                            val intentToDetail =
Intent(requireActivity(), DetailActivity::class.java)

                            intentToDetail.putExtra("DATA",
tesEmisi)

                            startActivity(intentToDetail)

                        }

                    } else {

                        Toast.makeText(requireActivity(), "GAGAL",
Toast.LENGTH_SHORT)

                            .show()

                        Log.e("Upload Activity ", "onFailure:
${response.message()}")

                        println(response.body())

                        println(response.body()?.message)

                    }

                }


                override fun onFailure(call: Call<ResponseEmisi>,
t: Throwable) {

                    Toast.makeText(

                        requireActivity(),

                        "getString(R.string.connection_failed)",

                        Toast.LENGTH_SHORT

                    ).show()

                }

            })

        }
```

3.5 Function Get History

```
override fun onCreateView(

        inflater: LayoutInflater,

        container: ViewGroup?,

        savedInstanceState: Bundle?
```

```
    ): View {
        _binding = FragmentNotificationsBinding.inflate(inflater,
container, false)
        val root: View = binding.root

        notificationsViewModel =
ViewModelProvider(this).get(NotificationsViewModel::class.java)

        val layoutManager = LinearLayoutManager(requireContext())
        binding.rvEmisi.layoutManager = layoutManager

        setupViewModel()

        notificationsViewModel.getHistory()

notificationsViewModel.listEmisi.observe(viewLifecycleOwner) {
setPlaceData(it) }

        return root
    }

    private fun setPlaceData(emisiData: List<DataItem>) {
        val listEmisi = ArrayList<Emisi>()
        for (i in emisiData.indices.reversed()) { // Loop dari
akhir ke awal
            val data = emisiData[i]
            val namaPengendara = data.namaPengendara
            val merekKendaraan = data.merekKendaraan
            val usiaMesin = data.usiaMesin
            val jenisBensin = data.jenisBensin
            val terakhirService = data.terakhirService
            val kadarCO = data.kadarCO
            val kadarHC = data.kadarHC
            val kadarNOX = data.kadarNOX
            val pertanyaan = ""
```

```
            val saranChatbot = data.response

            val kadarPartikulat = data.kadarPartikulat

            val emisi = Emisi(namaPengendara, merekKendaraan,
usiaMesin, jenisBensin, terakhirService, kadarCO, kadarHC,
kadarNOX, kadarPartikulat, pertanyaan, saranChatbot)

            listEmisi.add(emisi)

        }

        adapter = EmisiAdapter(listEmisi)

        binding.rvEmisi.adapter = adapter

        adapter.setOnItemClickCallback(object :
EmisiAdapter.OnItemClickCallback {

            override fun onItemClicked(data: Emisi) {

                val intentToDetail = Intent(requireContext(),
DetailActivity::class.java)

                intentToDetail.putExtra("DATA", data)

                startActivity(intentToDetail)

            }

        })

    }
```

3.6 Function Create User

```
override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)



        binding =
ActivityCreateAccountBinding.inflate(layoutInflater)

        setContentView(binding.root)



        binding.btnCreate.setOnClickListener{

            val sharedPreferences =
getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)

            val token = sharedPreferences.getString("token", null)

            val username = binding.edtUsername.text.toString()

            val password = binding.edtPassword.text.toString()

            val namaTeknisi =
binding.edtNamaTeknisi.text.toString()
```

```
            val client =
APIConfig.getAPIService().postUser(token.toString(), username,
password, namaTeknisi)
            client.enqueue(object : Callback<ResponseCreate> {
                override fun onResponse(
                    call: Call<ResponseCreate>,
                    response: Response<ResponseCreate>
                ) {
                    if (response.isSuccessful) {

startActivity(Intent(this@CreateAccountActivity,
DashboardActivity::class.java))
                        finish()
                    } else {
                        println(response)
                        println(response.body())
                        println(response.errorBody())
                    }
                }
                override fun onFailure(call: Call<ResponseCreate>,
t: Throwable) {
                    println("GAGAL CUI")
                }
            })
        }
    }
```

3.7 Class Get Users

```
class UsersAdapter(private val listusers: List<Users>) :
RecyclerView.Adapter<UsersAdapter.ListViewHolder>() {


    private lateinit var onItemClickCallback: OnItemClickCallback


    fun setOnItemClickCallback(onItemClickCallback:
OnItemClickCallback) {
```

```
        this.onItemClickCallback = onItemClickCallback
    }


    class ListViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        val tvNamaTeknisi: TextView =
itemView.findViewById(R.id.tvNamaTeknisi)
        val tvUsername: TextView =
itemView.findViewById(R.id.tvUsername)
    }


    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ListViewHolder {
        val view: View =
LayoutInflater.from(parent.context).inflate(R.layout.item_akun,
parent, false)
        return ListViewHolder(view)
    }


    override fun getItemCount(): Int = listusers.size


    override fun onBindViewHolder(holder: ListViewHolder,
position: Int) {
        val (id, username, name) = listusers[position]


        holder.tvNamaTeknisi.text = name
        holder.tvUsername.text = username
        holder.itemView.setOnClickListener {

onItemClickCallback.onItemClicked(listusers[holder.adapterPosition
])
        }
    }
    interface OnItemClickCallback {
        fun onItemClicked(data: Users)
    }
}
```

3.8 Function Edit User

```
super.onCreate(savedInstanceState)


        binding =
ActivityDetailUserBinding.inflate(layoutInflater)
        setContentView(binding.root)


        val data: Users? = intent.getParcelableExtra("DATA")


        val username = data!!.username.toString()
        val id = data.id.toString()
        val name = data.name.toString()


        binding.edtUsername.setText(username)
        binding.edtNamaTeknisi.setText(name)
        println(data)
        binding.buttonEdit.setOnClickListener{
            val sharedPreferences =
getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
            val token = sharedPreferences.getString("token", null)
            val username = binding.edtUsername.text.toString()
            val password = binding.edtPassword.text.toString()
            val namaTeknisi =
binding.edtNamaTeknisi.text.toString()


            val client =
APIConfig.getAPIService().editUser(token.toString(), id, id,
username, password, namaTeknisi)
            client.enqueue(object : Callback<ResponseEdit> {
                override fun onResponse(
                    call: Call<ResponseEdit>,
                    response: Response<ResponseEdit>
                ) {
                    if (response.isSuccessful) {
```

```
startActivity(Intent(this@DetailUserActivity,
DashboardActivity::class.java))
                        finish()
                    } else {
                        println(response)
                        println(response.body())
                        println(response.errorBody())
                    }
                }
                override fun onFailure(call: Call<ResponseEdit>,
t: Throwable) {
                    println("GAGAL CUI")
                }
            })
        }
```

3.9 Function Delete User

```
binding.buttonDelete.setOnClickListener{
            val sharedPreferences =
getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
            val token = sharedPreferences.getString("token", null)


            val client =
APIConfig.getAPIService().delete(token.toString(), id)
            client.enqueue(object : Callback<ResponseDelete> {
                override fun onResponse(
                    call: Call<ResponseDelete>,
                    response: Response<ResponseDelete>
                ) {
                    if (response.isSuccessful) {

startActivity(Intent(this@DetailUserActivity,
DashboardActivity::class.java))
                        finish()
                    } else {
```

```
                println(response)
                println(response.body())
                println(response.errorBody())
            }
        }
        override fun onFailure(call: Call<ResponseDelete>,
t: Throwable) {
            println("GAGAL CUI")
        }
    })
}
```