LAPORAN DOKUMENTASI TUGAS KRPL 1
MEMBANGUN SISTEM INPUT AGENDA HARIAN, MENYUSUN BERDASARKAN WAKTU DAN TINGKAT PRIORITAS

Disusun Untuk Memenuhi Mata Pelajaran KRPL

Nama Keloompok:

Alfian Prada A (06)

Ashila Putri N (08)

Fira Maya A (19)

Lang Natanegara (21)

M Galih Satriya (25)

PROGRAM KEAHLIAN PENGEMBANGAN PERANGKAT LUNAK DAN GIM

KOMPETENSI KEAHLIAN REKAYASA PERANGKAT LUNAK

SMK BRANTAS KARANGKATES

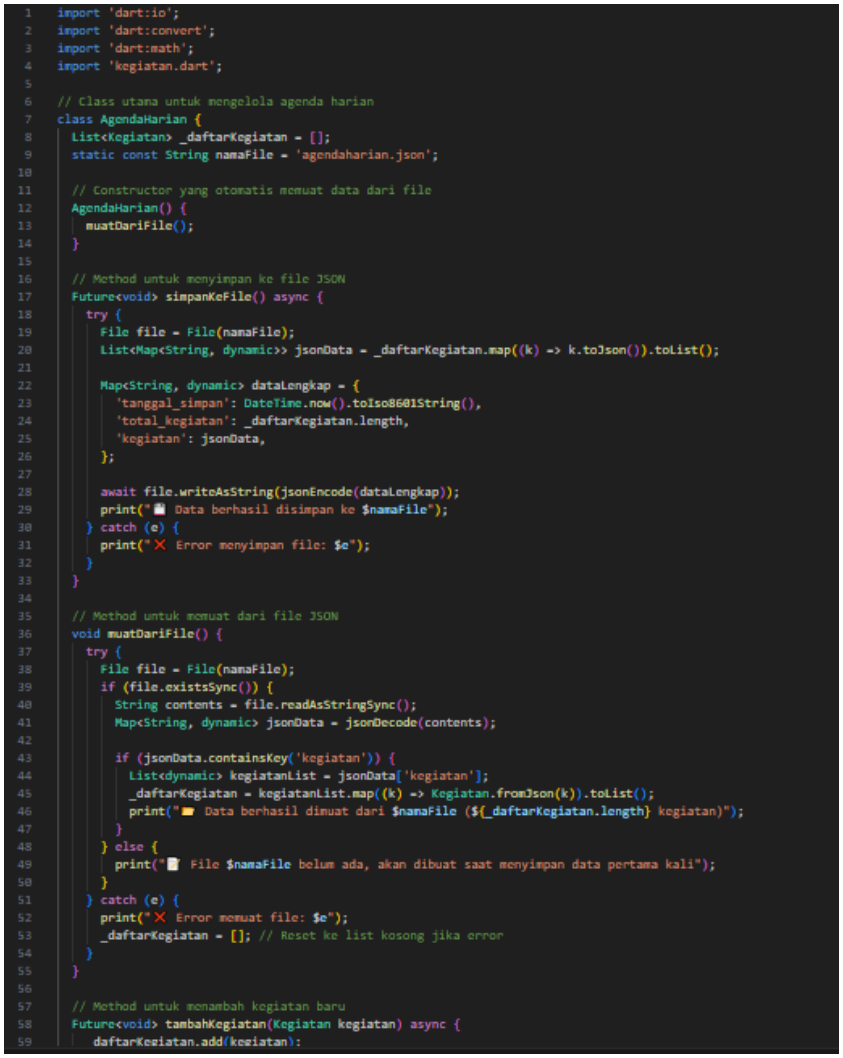
2025

Dokumentasi Penjelasan

1. Langkah pertama :
   a) Buka aplikasi Visual Code Studio (Vscode)
   b) Buat folder baru di folder dengan nama Agenda Harian
   c) Buka folder tersebut di Vscode

Sebelum membuat file dengan akhiran .dart, pastikan sdk tersebut sudah terinstal agar code bisa berjalan

2. Input Kegiatan Dan Prioritas
   A. Buat file terpisah untuk mempermudah penataan code\
      - ❖ Buat file dengan nama agenda.dart untuk mengkonfrensi agenda



```dart
import 'dart:io';
import 'dart:convert';
import 'dart:math';
import 'kegiatan.dart';

// Class utama untuk mengelola agenda harian
class AgendaHarian {
  List<Kegiatan> _daftarKegiatan = [];
  static const String namaFile = 'agendaharian.json';

  // Constructor yang otomatis memuat data dari file
  AgendaHarian() {
    muatDariFile();
  }

  // Method untuk menyimpan ke file JSON
  Future<void> simpanKeFile() async {
    try {
      File file = File(namaFile);
      List<Map<String, dynamic>> jsonData = _daftarKegiatan.map((k) => k.toJson()).toList();

      Map<String, dynamic> dataLengkap = {
        'tanggal_simpan': DateTime.now().toIso8601String(),
        'total_kegiatan': _daftarKegiatan.length,
        'kegiatan': jsonData,
      };

      await file.writeAsString(jsonEncode(dataLengkap));
      print("💾 Data berhasil disimpan ke $namaFile");
    } catch (e) {
      print("❌ Error menyimpan file: $e");
    }
  }

  // Method untuk memuat dari file JSON
  void muatDariFile() {
    try {
      File file = File(namaFile);
      if (file.existsSync()) {
        String contents = file.readAsStringSync();
        Map<String, dynamic> jsonData = jsonDecode(contents);

        if (jsonData.containsKey('kegiatan')) {
          List<dynamic> kegiatanList = jsonData['kegiatan'];
          _daftarKegiatan = kegiatanList.map((k) => Kegiatan.fromJson(k)).toList();
          print("📂 Data berhasil dimuat dari $namaFile (${_daftarKegiatan.length} kegiatan)");
        }
      } else {
        print("📝 File $namaFile belum ada, akan dibuat saat menyimpan data pertama kali");
      }
    } catch (e) {
      print("❌ Error memuat file: $e");
      _daftarKegiatan = []; // Reset ke list kosong jika error
    }
  }

  // Method untuk menambah kegiatan baru
  Future<void> tambahKegiatan(Kegiatan kegiatan) async {
    daftarKegiatan.add(kegiatan);
```

```dart
  Future<void> tambahKegiatan(Kegiatan kegiatan) async {
    _daftarKegiatan.add(kegiatan);
    await simpanKeFile(); // Otomatis simpan setelah menambah
    print("✓ Kegiatan '${kegiatan.nama}' berhasil ditambahkan dan disimpan!");
  }

  // Method untuk mengurutkan kegiatan berdasarkan prioritas dan waktu
  void urutkanKegiatan() {
    _daftarKegiatan.sort((a, b) {
      // Prioritas utama: tingkat prioritas (tinggi ke rendah)
      int perbandinganPrioritas = b.nilaiPrioritas.compareTo(a.nilaiPrioritas);

      // Jika prioritas sama, urutkan berdasarkan waktu (awal ke akhir)
      if (perbandinganPrioritas == 0) {
        return a.waktu.compareTo(b.waktu);
      }

      return perbandinganPrioritas;
    });
  }

  // Method untuk menghitung lebar kolom yang dinamis
  Map<String, int> _hitungLebarKolom() {
    if (_daftarKegiatan.isEmpty) {
      return {
        'no': 4,
        'waktu': 11,
        'kegiatan': 12,
        'prioritas': 11,
        'deskripsi': 13
      };
    }

    // Lebar minimum untuk setiap kolom
    int lebarNo = max(4, _daftarKegiatan.length.toString().length + 2);
    int lebarWaktu = max(11, "Waktu".length + 2);
    int lebarKegiatan = max(12, "Kegiatan".length + 2);
    int lebarPrioritas = max(11, "Prioritas".length + 2);
    int lebarDeskripsi = max(13, "Deskripsi".length + 2);

    // Hitung lebar maksimum berdasarkan konten
    for (Kegiatan k in _daftarKegiatan) {
      lebarWaktu = max(lebarWaktu, k.formatWaktu.length + 2);
      lebarKegiatan = max(lebarKegiatan, k.nama.length + 2);
      lebarPrioritas = max(lebarPrioritas, k.formatPrioritas.length + 2);
      lebarDeskripsi = max(lebarDeskripsi, (k.deskripsi ?? "").length + 2);
    }

    return {
      'no': lebarNo,
      'waktu': lebarWaktu,
      'kegiatan': lebarKegiatan,
      'prioritas': lebarPrioritas,
      'deskripsi': lebarDeskripsi
    };
  }

  // Method untuk membuat separator tabel
```

```dart
  String _buatSeparator(Map<String, int> lebar) {
    String separator = "|";
    separator += "-" * lebar['no']! + "|";
    separator += "-" * lebar['waktu']! + "|";
    separator += "-" * lebar['kegiatan']! + "|";
    separator += "-" * lebar['prioritas']! + "|";
    separator += "-" * lebar['deskripsi']! + "|";
    return separator;
  }

  // Method untuk memformat baris tabel
  String _formatBaris(String no, String waktu, String kegiatan, String prioritas, String deskripsi, Map<String, int> lebar) {
    return "|${no.padLeft(lebar['no']! - 1)} |${waktu.padLeft(lebar['waktu']! - 1)} |${kegiatan.padLeft(lebar['kegiatan']! - 1)} |${prioritas.padLeft(lebar['prioritas']! - 1)} |${deskripsi.padLeft(lebar['deskripsi']! - 1)} |";
  }

  // Method untuk menampilkan agenda dalam bentuk tabel CLI dinamis
  void tampilkanAgenda() {
    if (_daftarKegiatan.isEmpty) {
      print("\n❌ Tidak ada kegiatan dalam agenda!");
      return;
    }

    urutkanKegiatan();

    // Hitung lebar kolom yang dinamis
    Map<String, int> lebarKolom = _hitungLebarKolom();

    // Hitung total lebar tabel
    int totalLebar = lebarKolom.values.reduce((a, b) => a + b) + 5; // +5 untuk separator

    print("\n" + "="*totalLebar);
    print("📅 AGENDA HARIAN PRIORITAS".padLeft((totalLebar + "📅 AGENDA HARIAN PRIORITAS".length) ~/ 2));
    print("="*totalLebar);

    // Header tabel
    print(_formatBaris("No", "Waktu", "Kegiatan", "Prioritas", "Deskripsi", lebarKolom));
    print(_buatSeparator(lebarKolom));

    // Isi tabel
    for (int i = 0; i < _daftarKegiatan.length; i++) {
      Kegiatan k = _daftarKegiatan[i];
      String no = (i + 1).toString();
      String waktu = k.formatWaktu;
      String nama = k.nama;
      String prioritas = k.formatPrioritas;
      String deskripsi = k.deskripsi ?? "";

      print(_formatBaris(no, waktu, nama, prioritas, deskripsi, lebarKolom));
    }

    print("="*totalLebar);
    print("Total kegiatan: ${_daftarKegiatan.length}");
  }

  // Method untuk menampilkan agenda dengan mode compact (jika terlalu lebar untuk terminal)
  void tampilkanAgendaCompact() {
```

```dart
void tampilkanAgendaCompact() {
  if (_daftarKegiatan.isEmpty) {
    print("\n❌ Tidak ada kegiatan dalam agenda!");
    return;
  }

  urutkanKegiatan();

  // Batasi lebar maksimum untuk mode compact
  const int maxLebarKegiatan = 20;
  const int maxLebarDeskripsi = 25;

  Map<String, int> lebarKolom = _hitungLebarKolom();

  // Batasi lebar jika terlalu panjang
  if (lebarKolom['kegiatan']! > maxLebarKegiatan) {
    lebarKolom['kegiatan'] = maxLebarKegiatan;
  }
  if (lebarKolom['deskripsi']! > maxLebarDeskripsi) {
    lebarKolom['deskripsi'] = maxLebarDeskripsi;
  }

  int totalLebar = lebarKolom.values.reduce((a, b) => a + b) + 5;

  print("\n" + "="*totalLebar);
  print("📅 AGENDA HARIAN PRIORITAS".padLeft((totalLebar + "📅 AGENDA HARIAN PRIORITAS".length) ~/ 2));
  print("="*totalLebar);

  // Header tabel
  print(_formatBaris("No", "Waktu", "Kegiatan", "Prioritas", "Deskripsi", lebarKolom));
  print(_buatSeparator(lebarKolom));

  // Isi tabel dengan pemotongan teks jika perlu
  for (int i = 0; i < _daftarKegiatan.length; i++) {
    Kegiatan k = _daftarKegiatan[i];
    String no = (i + 1).toString();
    String waktu = k.formatWaktu;
    String nama = k.nama.length > maxLebarKegiatan - 2
        ? k.nama.substring(0, maxLebarKegiatan - 5) + "..."
        : k.nama;
    String prioritas = k.formatPrioritas;
    String deskripsi = (k.deskripsi ?? "").length > maxLebarDeskripsi - 2
        ? (k.deskripsi ?? "").substring(0, maxLebarDeskripsi - 5) + "..."
        : (k.deskripsi ?? "");

    print(_formatBaris(no, waktu, nama, prioritas, deskripsi, lebarKolom));
  }

  print("="*totalLebar);
  print("Total kegiatan: ${_daftarKegiatan.length}");
}

// Method untuk menghapus kegiatan
Future<bool> hapusKegiatan(int index) async {
  if (index >= 0 && index < _daftarKegiatan.length) {
    String namaKegiatan = _daftarKegiatan[index].nama;
    _daftarKegiatan.removeAt(index);
```

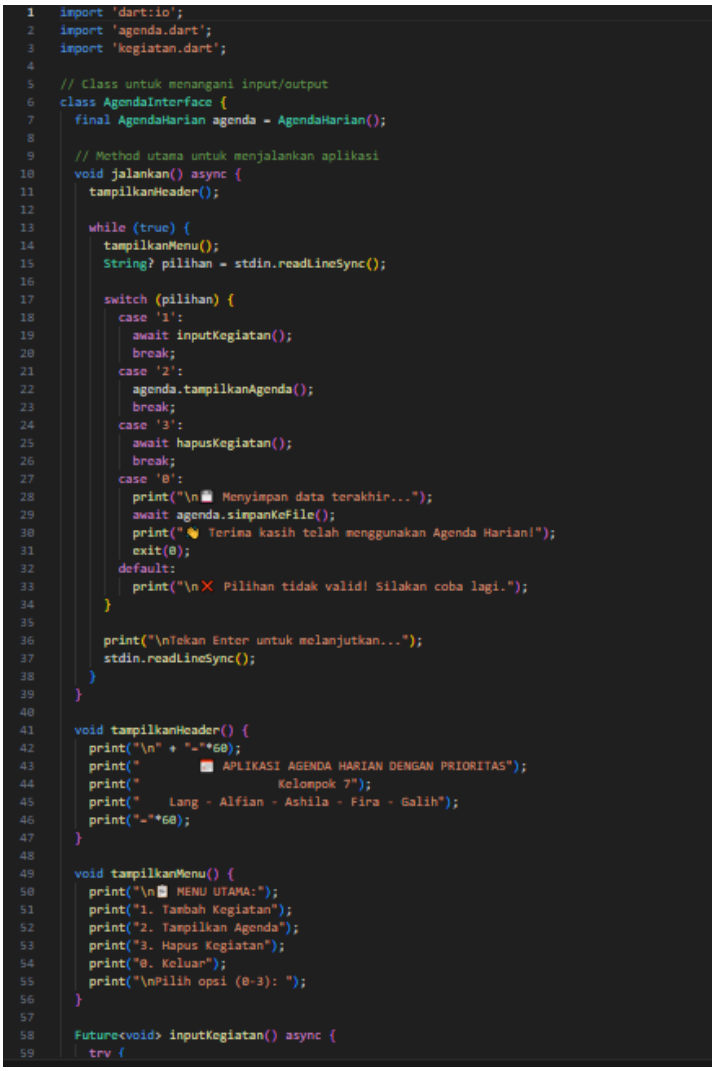
```dart
      await simpanKeFile(); // Otomatis simpan setelah menghapus
      print("✓ Kegiatan '$namaKegiatan' berhasil dihapus dan disimpan!");
      return true;
    }
    return false;
  }

  // Getter untuk mendapatkan jumlah kegiatan
  int get jumlahKegiatan => _daftarKegiatan.length;
}
```

❖ Buat file dengan nama kegiatan.dart untuk mengkonfrensi agenda kegiatan

```dart
// Enum untuk tingkat prioritas
enum Prioritas { rendah, sedang, tinggi, mendesak }

// Class untuk merepresentasikan kegiatan
class Kegiatan {
  String nama;
  DateTime waktu;
  Prioritas prioritas;
  String? deskripsi;

  Kegiatan({
    required this.nama,
    required this.waktu,
    required this.prioritas,
    this.deskripsi,
  });

  // Getter untuk mendapatkan nilai numerik prioritas (untuk sorting)
  int get nilaiPrioritas {
    switch (prioritas) {
      case Prioritas.rendah:
        return 1;
      case Prioritas.sedang:
        return 2;
      case Prioritas.tinggi:
        return 3;
      case Prioritas.mendesak:
        return 4;
    }
  }

  // Method untuk format waktu
  String get formatWaktu {
    return "${waktu.hour.toString().padLeft(2, '0')}:${waktu.minute.toString().padLeft(2, '0')}";
  }

  // Method untuk format prioritas
  String get formatPrioritas {
    switch (prioritas) {
      case Prioritas.rendah:
        return "Rendah";
      case Prioritas.sedang:
        return "Sedang";
      case Prioritas.tinggi:
        return "Tinggi";
      case Prioritas.mendesak:
        return "Mendesak";
    }
  }

  // Method untuk convert ke JSON
  Map<String, dynamic> toJson() {
    return {
      'nama': nama,
      'waktu': waktu.millisecondsSinceEpoch,
      'prioritas': prioritas.index,
      'deskripsi': deskripsi,
    };
  }
```

```dart
  // Method untuk convert ke JSON
  Map<String, dynamic> toJson() {
    return {
      'nama': nama,
      'waktu': waktu.millisecondsSinceEpoch,
      'prioritas': prioritas.index,
      'deskripsi': deskripsi,
    };
  }

  // Factory method untuk membuat dari JSON
  factory Kegiatan.fromJson(Map<String, dynamic> json) {
    return Kegiatan(
      nama: json['nama'],
      waktu: DateTime.fromMillisecondsSinceEpoch(json['waktu']),
      prioritas: Prioritas.values[json['prioritas']],
      deskripsi: json['deskripsi'],
    );
  }

  @override
  String toString() {
    return 'Kegiatan: $nama, Waktu: $formatWaktu, Prioritas: $formatPrioritas';
  }
}
```

❖ Buat file dengan nama main.dart dengan isi seperti ini



```dart
import 'dart:io';
import 'agenda.dart';
import 'kegiatan.dart';

// Class untuk menangani input/output
class AgendaInterface {
  final AgendaHarian agenda = AgendaHarian();

  // Method utama untuk menjalankan aplikasi
  void jalankan() async {
    tampilkanHeader();

    while (true) {
      tampilkanMenu();
      String? pilihan = stdin.readLineSync();

      switch (pilihan) {
        case '1':
          await inputKegiatan();
          break;
        case '2':
          agenda.tampilkanAgenda();
          break;
        case '3':
          await hapusKegiatan();
          break;
        case '0':
          print("\n💾 Menyimpan data terakhir...");
          await agenda.simpanKeFile();
          print("👋 Terima kasih telah menggunakan Agenda Harian!");
          exit(0);
        default:
          print("\n❌ Pilihan tidak valid! Silakan coba lagi.");
      }

      print("\nTekan Enter untuk melanjutkan...");
      stdin.readLineSync();
    }
  }

  void tampilkanHeader() {
    print("\n" + "=" * 60);
    print("          📅 APLIKASI AGENDA HARIAN DENGAN PRIORITAS");
    print("                          Kelompok 7");
    print("    Lang - Alfian - Ashila - Fira - Galih");
    print("=" * 60);
  }

  void tampilkanMenu() {
    print("\n📋 MENU UTAMA:");
    print("1. Tambah Kegiatan");
    print("2. Tampilkan Agenda");
    print("3. Hapus Kegiatan");
    print("0. Keluar");
    print("\nPilih opsi (0-3): ");
  }

  Future<void> inputKegiatan() async {
    try {
```

```dart
      print("\n➕ TAMBAH KEGIATAN BARU");
      print("-" * 25);

      // Input nama kegiatan
      print("Nama kegiatan: ");
      String? nama = stdin.readLineSync();
      if (nama == null || nama.trim().isEmpty) {
        print("❌ Nama kegiatan tidak boleh kosong!");
        return;
      }

      // Input waktu
      print("Waktu (format HH:MM, contoh: 14:30): ");
      String? inputWaktu = stdin.readLineSync();
      if (inputWaktu == null) {
        print("❌ Waktu tidak valid!");
        return;
      }

      DateTime waktu = parseWaktu(inputWaktu);

      // Input prioritas
      print("Prioritas:");
      print("1. Rendah");
      print("2. Sedang");
      print("3. Tinggi");
      print("4. Mendesak");
      print("Pilih (1-4): ");
```

```
        String? inputPrioritas = stdin.readLineSync();
        Prioritas prioritas = parsePrioritas(inputPrioritas);

        // Input deskripsi (opsional)
        print("Deskripsi (opsional): ");
        String? deskripsi = stdin.readLineSync();

        // Buat dan tambah kegiatan
        Kegiatan kegiatan = Kegiatan(
          nama: nama.trim(),
          waktu: waktu,
          prioritas: prioritas,
          deskripsi: deskripsi?.trim().isEmpty == true ? null : deskripsi?.trim(),
        );

        await agenda.tambahKegiatan(kegiatan);

      } catch (e) {
        print("❌ Error: $e");
      }
    }

    DateTime parseWaktu(String input) {
      try {
        List<String> bagian = input.split(':');
        if (bagian.length != 2) throw FormatException("Format waktu salah");

        int jam = int.parse(bagian[0]);
        int menit = int.parse(bagian[1]);

        if (jam < 0 || jam > 23 || menit < 0 || menit > 59) {
          throw FormatException("Waktu tidak valid");
        }

        DateTime sekarang = DateTime.now();
        return DateTime(sekarang.year, sekarang.month, sekarang.day, jam, menit);
      } catch (e) {
        throw FormatException("Format waktu salah! Gunakan format HH:MM");
      }
    }

    Prioritas parsePrioritas(String? input) {
      switch (input) {
        case '1':
          return Prioritas.rendah;
        case '2':
          return Prioritas.sedang;
        case '3':
          return Prioritas.tinggi;
        case '4':
          return Prioritas.mendesak;
        default:
          print("⚠️  Prioritas tidak valid, menggunakan prioritas sedang");
          return Prioritas.sedang;
      }
    }

    Future<void> hapusKegiatan() async {
      if (agenda.jumlahKegiatan == 0) {
        print("\n❌ Tidak ada kegiatan untuk dihapus!");
        return;
      }

      agenda.tampilkanAgenda();
      print("\nMasukkan nomor kegiatan yang akan dihapus (1-${agenda.jumlahKegiatan}): ");
      String? input = stdin.readLineSync();

      try {
        int index = int.parse(input!) - 1;
        if (!await agenda.hapusKegiatan(index)) {
          print("❌ Nomor kegiatan tidak valid!");
        }
      } catch (e) {
        print("❌ Input tidak valid!");
      }
    }
  }

  void main() {
    AgendaInterface app = AgendaInterface();
    app.jalankan();
  }
```