LAPORAN
TUGAS AHIR
“ Program Bereorientasi Obyek ”

Dosen Pengampu : M Bahrul Subkhi, M.Kom

Disusun Oleh :

Reihan Rafi Rahmatuloh (2213020244)

Nugroho Bimantoro (2213020224)

Moch. Saefudin Yuhri (2213020205)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS NUSANTARA PGRI KEDIRI 2023

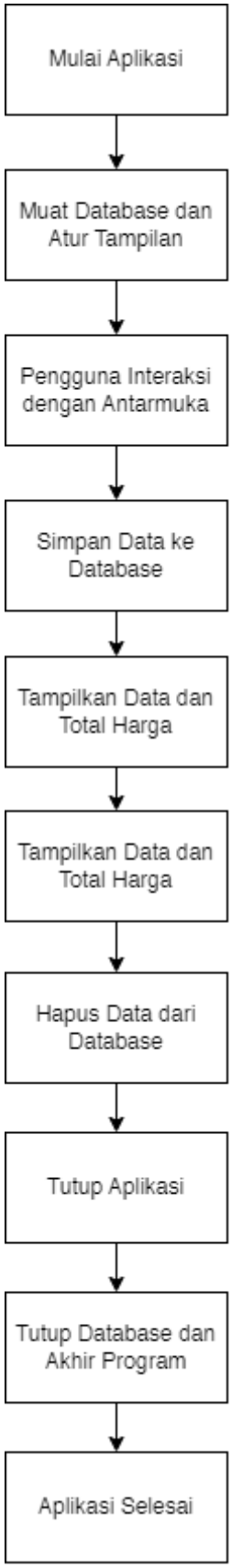
LATAR BELAKANG

- Latar belakang pembuatan aplikasi parkir dapat bermacam-macam, tetapi beberapa alasan umum termasuk meningkatkan efisiensi pengelolaan parkir, memberikan kemudahan bagi pengguna untuk menemukan dan membayar tempat parkir, serta mengurangi kemacetan dan polusi udara akibat pencarian tempat parkir yang tidak efisien. Selain itu, aplikasi parkir juga dapat membantu mengoptimalkan pemanfaatan ruang parkir dan meningkatkan pengalaman pengguna dalam mengakses fasilitas parkir.
- Selain itu, latar belakang pembuatan aplikasi parkir juga melibatkan adaptasi terhadap perkembangan teknologi untuk memenuhi tuntutan zaman. Aplikasi parkir modern tidak hanya menyediakan informasi mengenai ketersediaan tempat parkir, tetapi juga dapat memanfaatkan sensor dan teknologi cerdas untuk mengelola parkir secara lebih efisien, mempercepat proses pembayaran, dan meningkatkan keamanan area parkir. Dengan demikian, aplikasi parkir berperan dalam menciptakan solusi terintegrasi untuk memecahkan permasalahan parkir perkotaan yang semakin kompleks.

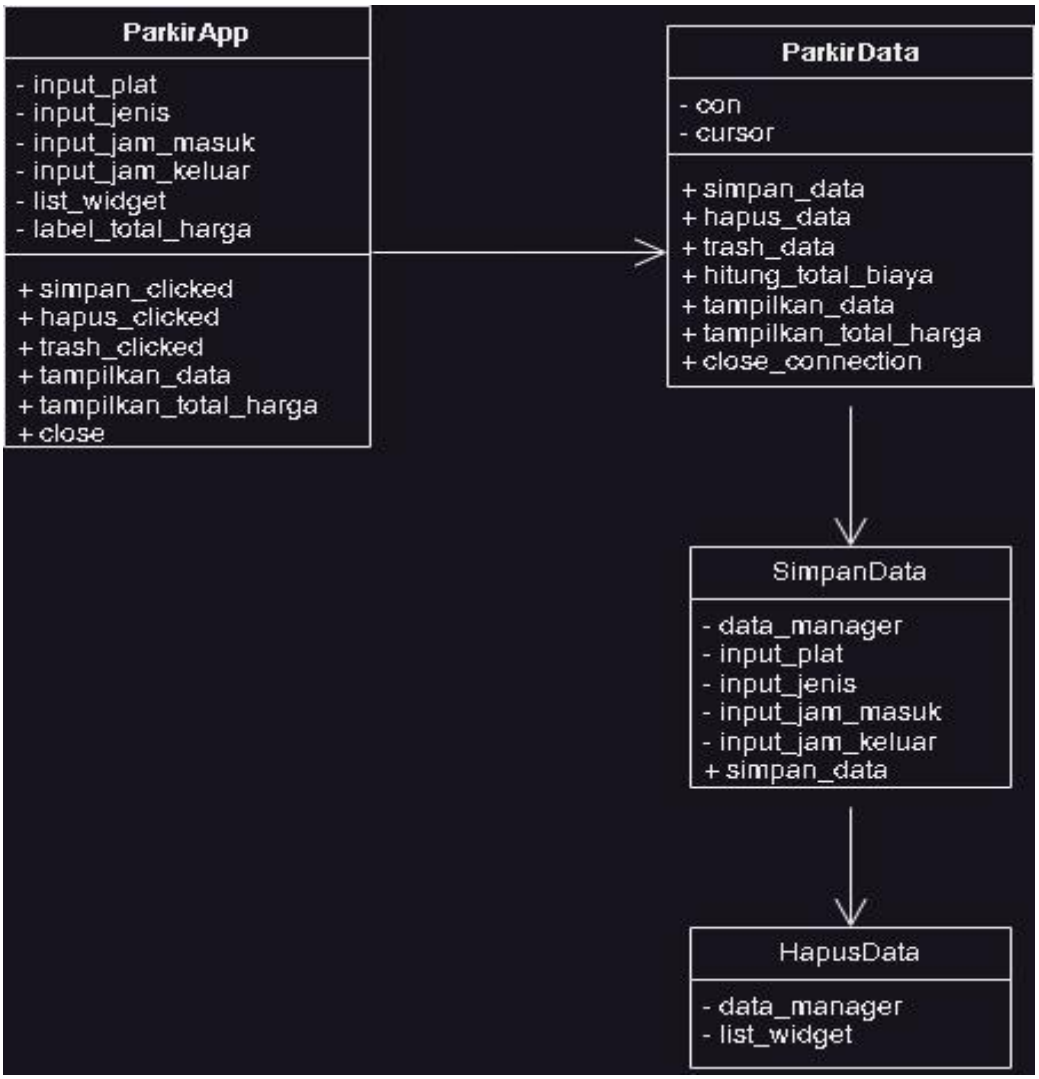
DAFTAR ISI

BAB I

Flowchat Sistem



Clas Diagram



BAB II
Hasil Program dan Penjelasan

D: > PRAK PBO > Project UAS PBO smst3.py > ...

```python
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QDateTimeEdit, QPushButton, QVBoxLayout, QListWidget, QListWidgetItem, QGroupB
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import QSize
import sqlite3

class ParkirData:
    def __init__(self):
        self.conn = sqlite3.connect("data_parkir.db")
        self.cursor = self.conn.cursor()

        # Perbarui skema tabel dengan menambahkan kolom "deleted"
        self.cursor.execute('''
            PRAGMA foreign_keys=off;
        ''')

        # Buat tabel sementara baru dengan skema yang diperbarui
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS temp_data_parkir (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                plat_kendaraan TEXT,
                jenis_kendaraan TEXT,
                jam_masuk TEXT,
                jam_keluar TEXT,
                harga INTEGER,
                deleted INTEGER DEFAULT 0
            );
        ''')

        # Salin data dari tabel lama ke tabel sementara
        self.cursor.execute('''
            INSERT INTO temp_data_parkir (plat_kendaraan, jenis_kendaraan, jam_masuk, jam_keluar, harga, deleted)
            SELECT plat_kendaraan, jenis_kendaraan, jam_masuk, jam_keluar, harga, 0
            FROM data_parkir;
        ''')

        # Hapus tabel lama
        self.cursor.execute('''
```

D: > PRAK PBO > Project UAS PBO smst3.py > ...

```python
        # Hapus tabel lama
        self.cursor.execute('''
            DROP TABLE data_parkir;
        ''')

        # Ubah nama tabel sementara menjadi nama tabel yang baru
        self.cursor.execute('''
            ALTER TABLE temp_data_parkir RENAME TO data_parkir;
        ''')

        self.conn.commit()
        self.conn.execute('PRAGMA foreign_keys=on;')

    def simpan_data(self, plat_kendaraan, jenis_kendaraan, jam_masuk, jam_keluar, harga):
        self.cursor.execute("INSERT INTO data_parkir (plat_kendaraan, jenis_kendaraan, jam_masuk, jam_keluar, harga) VALUES (?, ?, ?, ?, ?)",
                            (plat_kendaraan, jenis_kendaraan, jam_masuk, jam_keluar, harga))
        self.conn.commit()

    def hapus_data(self, plat_kendaraan):
        self.cursor.execute("UPDATE data_parkir SET deleted = 1 WHERE plat_kendaraan=?", (plat_kendaraan,))
        self.conn.commit()

    def trash_data(self, plat_kendaraan):
        # Implementasi trash atau operasi hapus permanen dapat ditambahkan di sini
        pass

    def hitung_total_biaya(self):
        return 5000

    def tampilkan_data(self):
        self.cursor.execute("SELECT * FROM data_parkir WHERE deleted = 0")
        data = self.cursor.fetchall()
        return data

    def tampilkan_total_harga(self):
        self.cursor.execute("SELECT SUM(harga) FROM data_parkir WHERE deleted = 0")
        total_harga = self.cursor.fetchone()[0]
```

D: > PRAK PBO > Project UAS PBO smst3.py > ...

```
    def tampilkan_total_harga(self):
        self.cursor.execute("SELECT SUM(harga) FROM data_parkir WHERE deleted = 0")
        total_harga = self.cursor.fetchone()[0]
        return total_harga

    def close_connection(self):
        self.conn.close()

class SimpanDataHandler:
    def __init__(self, data_manager, input_plat, input_jenis, input_jam_masuk, input_jam_keluar):
        self.data_manager = data_manager
        self.input_plat = input_plat
        self.input_jenis = input_jenis
        self.input_jam_masuk = input_jam_masuk
        self.input_jam_keluar = input_jam_keluar

    def simpan_data(self):
        plat_kendaraan = self.input_plat.text()
        jenis_kendaraan = self.input_jenis.text()
        jam_masuk = self.input_jam_masuk.dateTime().toString('yyyy-MM-dd HH:mm:ss')
        jam_keluar = self.input_jam_keluar.dateTime().toString('yyyy-MM-dd HH:mm:ss')
        harga = self.data_manager.hitung_total_biaya()

        self.data_manager.simpan_data(plat_kendaraan, jenis_kendaraan, jam_masuk, jam_keluar, harga)

class HapusDataHandler:
    def __init__(self, data_manager, list_widget):
        self.data_manager = data_manager
        self.list_widget = list_widget

    def hapus_data(self):
        selected_item = self.list_widget.currentItem()
        if selected_item:
            plat_kendaraan = selected_item.text().split(",")[0].split(":")[1].strip()
            self.data_manager.hapus_data(plat_kendaraan)
```

D: > PRAK PBO > Project UAS PBO smst3.py > ...

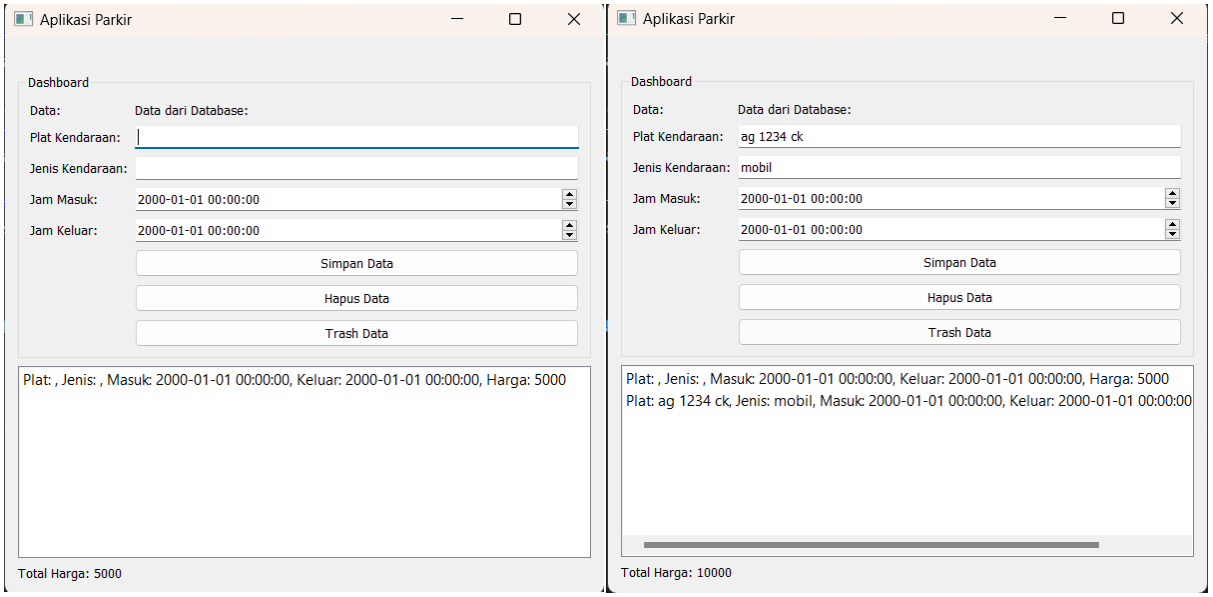
```
class TrashDataHandler:
    def __init__(self, data_manager, list_widget):
        self.data_manager = data_manager
        self.list_widget = list_widget

    def trash_data(self):
        selected_item = self.list_widget.currentItem()
        if selected_item:
            plat_kendaraan = selected_item.text().split(",")[0].split(":")[1].strip()
            self.data_manager.trash_data(plat_kendaraan)

class ParkirApp(QWidget):
    def __init__(self):
        super().__init__()

        self.data_manager = ParkirData()

        self.setMinimumSize(QSize(600, 400))
        self.setWindowTitle("Aplikasi Parkir")

        background_label = QLabel(self)
        pixmap = QPixmap("background_image.jpg")
        background_label.setPixmap(pixmap)

        dashboard_group = QGroupBox("Dashboard")
        dashboard_layout = QFormLayout()

        self.label_data = QLabel("Data dari Database: ")
        dashboard_layout.addRow("Data:", self.label_data)

        self.input_plat = QLineEdit()
        dashboard_layout.addRow("Plat Kendaraan:", self.input_plat)

        self.input_jenis = QLineEdit()
        dashboard_layout.addRow("Jenis Kendaraan:", self.input_jenis)
```

```
        self.input_jam_masuk = QDateTimeEdit(self)
        self.input_jam_masuk.setDisplayFormat("yyyy-MM-dd HH:mm:ss")
        dashboard_layout.addRow("Jam Masuk:", self.input_jam_masuk)

        self.input_jam_keluar = QDateTimeEdit(self)
        self.input_jam_keluar.setDisplayFormat("yyyy-MM-dd HH:mm:ss")
        dashboard_layout.addRow("Jam Keluar:", self.input_jam_keluar)

        btn_simpan = QPushButton("Simpan Data")
        btn_simpan.clicked.connect(self.on_simpan_clicked)
        dashboard_layout.addRow("", btn_simpan)

        btn_hapus = QPushButton("Hapus Data")
        btn_hapus.clicked.connect(self.on_hapus_clicked)
        dashboard_layout.addRow("", btn_hapus)

        btn_trash = QPushButton("Trash Data")
        btn_trash.clicked.connect(self.on_trash_clicked)
        dashboard_layout.addRow("", btn_trash)

        dashboard_group.setLayout(dashboard_layout)

        layout = QVBoxLayout(self)
        layout.addWidget(background_label)
        layout.addWidget(dashboard_group)

        self.list_widget = QListWidget()
        layout.addWidget(self.list_widget)

        self.label_total_harga = QLabel("Total Harga: 0")
        layout.addWidget(self.label_total_harga)

        # Perbarui tampilan data dan total harga pada saat aplikasi dimulai
        self.tampilkan_data()
        self.tampilkan_total_harga()
```



```
    def on_simpan_clicked(self):
        simpan_handler = SimpanDataHandler(self.data_manager, self.input_plat, self.input_jenis, self.input_jam_masuk, self.input_jam_keluar)
        simpan_handler.simpan_data()
        self.tampilkan_data()
        self.tampilkan_total_harga()

    def on_hapus_clicked(self):
        hapus_handler = HapusDataHandler(self.data_manager, self.list_widget)
        hapus_handler.hapus_data()
        self.tampilkan_data()
        self.tampilkan_total_harga()

    def on_trash_clicked(self):
        trash_handler = TrashDataHandler(self.data_manager, self.list_widget)
        trash_handler.trash_data()
        self.tampilkan_data()
        self.tampilkan_total_harga()

    def tampilkan_data(self):
        data = self.data_manager.tampilkan_data()

        self.list_widget.clear()

        for row in data:
            item = QListWidgetItem(f"Plat: {row[1]}, Jenis: {row[2]}, Masuk: {row[3]}, Keluar: {row[4]}, Harga: {row[5]}")
            self.list_widget.addItem(item)

    def tampilkan_total_harga(self):
        total_harga = self.data_manager.tampilkan_total_harga()
        self.label_total_harga.setText(f"Total Harga: {total_harga}")

    def closeEvent(self, event):
        self.data_manager.close_connection()
        event.accept()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = ParkirApp()
    window.show()
    sys.exit(app.exec_())
```

Hasil Coding



Penjelasan Coding

Kodingan ini adalah dasar dari sebuah aplikasi parkir sederhana menggunakan bahasa pemrograman Python dan library PyQt5 untuk pembuatan antarmuka grafis. Aplikasi ini menggunakan SQLite sebagai database untuk menyimpan data parkir.

Berikut adalah poin-poin utama dari kodingan ini:

1. Manajemen Database:
   - Ada kelas ParkirData yang bertanggung jawab atas inisialisasi dan manajemen database SQLite.
   - Tabel temp_data_parkir dibuat dengan tambahan kolom deleted.
   - Skema tabel data_parkir diperbarui, dan tabel lama dihapus.

2. Fungsi-fungsi di dalam ParkirData:
   - Fungsi simpan_data untuk menyimpan data kendaraan.
   - Fungsi hapus_data untuk menandai data sebagai dihapus.
   - Fungsi trash_data (belum diimplementasikan) mungkin untuk menghapus data secara permanen.
   - Fungsi hitung_total_biaya mengembalikan biaya parkir tetap (mungkin biaya per jam).
   - Fungsi tampilkan_data untuk mendapatkan data kendaraan yang tidak dihapus dari database.
   - Fungsi tampilkan_total_harga untuk mendapatkan total harga dari data kendaraan yang tidak dihapus.

3. Handler untuk Simpan, Hapus, dan Trash Data:
   - SimpanDataHandler menangani penyimpanan data baru ke dalam database.
   - HapusDataHandler menangani penandaan data sebagai dihapus.
   - TrashDataHandler (belum diimplementasikan) mungkin untuk menghapus data secara permanen.

4. Kelas ParkirApp (subclass dari QWidget):
   - Mewarisi dari QWidget untuk membuat antarmuka grafis.
   - Menampilkan elemen-elemen seperti label, input fields, dan tombol untuk mengelola data parkir.
   - Membuat objek ParkirData sebagai manajer data.
   - Menggunakan handler untuk simpan, hapus, dan trash data.
   - Menampilkan daftar data parkir dan total harga pada antarmuka grafis.
   - Mengimplementasikan fungsi untuk memperbarui antarmuka dengan data terbaru.

5. Eksekusi Aplikasi:
   - Membuat objek QApplication.
   - Membuat objek ParkirApp dan menampilkannya.
   - Memulai loop eksekusi aplikasi dengan app.exec_().

Aplikasi ini memungkinkan pengguna untuk menambahkan, menghapus (dengan penandaan), dan (potensial) menghapus permanen data parkir. Data parkir ditampilkan dalam daftar, dan total harga diupdate secara real-time. Aplikasi juga memiliki fitur untuk menangani waktu masuk dan keluar kendaraan serta menampilkan data pada antarmuka pengguna.