LAPORAN TUGAS PEMROGAMAN VISUAL

SISTEM MANAJEMEN PERPUSTAKAAN BERBASIS DESKTOP

Dosen Pengampun : Pahrul Irfan, S.Kom., M.Kom.

Oleh :

Dewi Agustin Asri

F1D022039

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MATARAM

2025

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Perpustakaan merupakan sarana penting dalam mendukung kegiatan belajar, penelitian, dan pengembangan ilmu pengetahuan. Di dalamnya terdapat berbagai macam koleksi buku yang harus dikelola dengan baik, mulai dari pencatatan data buku, proses peminjaman, pengembalian, hingga penghitungan denda. Namun, dalam praktiknya, banyak perpustakaan masih mengelola data secara manual menggunakan pencatatan buku atau aplikasi spreadsheet yang sederhana. Hal ini menimbulkan sejumlah permasalahan seperti data yang mudah hilang, pencarian buku yang memakan waktu, serta kesalahan dalam pencatatan peminjaman dan pengembalian.

Seiring dengan perkembangan teknologi informasi, kebutuhan akan sistem manajemen perpustakaan yang efisien dan terkomputerisasi menjadi semakin penting. Sistem ini diharapkan mampu mengotomatisasi proses manajemen buku, memberikan kemudahan dalam pencatatan data, serta meningkatkan keakuratan dan kecepatan dalam pengolahan informasi perpustakaan.

Untuk menjawab kebutuhan tersebut, dibangunlah sebuah Sistem Manajemen Perpustakaan berbasis desktop menggunakan bahasa pemrograman Python dengan library PyQt5 untuk antarmuka grafis (GUI) dan SQLite sebagai basis data. Aplikasi ini dirancang untuk membantu pustakawan dalam melakukan pengelolaan data buku, termasuk menambah, mengedit, menghapus, serta melakukan pencarian dan filter berdasarkan status buku (tersedia atau dipinjam). Selain itu, sistem ini juga mendukung pencatatan tanggal pinjam dan kembali, perhitungan denda otomatis, ekspor data ke dalam format CSV, serta menampilkan informasi statistik dalam bentuk kartu (card) seperti total buku, buku yang dipinjam, dan yang tersedia.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah yang dapat diidentifikasi dalam pembangunan sistem manajemen perpustakaan ini adalah sebagai berikut:

1. Bagaimana membangun sebuah aplikasi desktop yang dapat memudahkan proses pengelolaan data buku di perpustakaan?
2. Bagaimana mengelola data peminjaman dan pengembalian buku secara terstruktur dan akurat?
3. Bagaimana menghitung denda keterlambatan pengembalian buku secara otomatis?
4. Bagaimana menampilkan statistik data buku secara ringkas dan informatif kepada pengguna?
5. Bagaimana cara menyimpan, mencari, dan menyaring data buku dengan efektif dan efisien?

1.3 Tujuan

Tujuan dari pengembangan aplikasi Sistem Manajemen Perpustakaan ini adalah sebagai berikut:

1. Membangun sebuah aplikasi desktop yang dapat mempermudah pengelolaan data buku di perpustakaan.
2. Menyediakan fitur pencatatan data peminjaman dan pengembalian buku yang lengkap dan akurat.
3. Mengotomatisasi perhitungan denda keterlambatan pengembalian buku.
4. Menyediakan statistik jumlah total buku, jumlah buku yang sedang dipinjam, dan buku yang tersedia dalam bentuk kartu informasi (cards).

5. Menyediakan fitur pencarian dan filter buku berdasarkan status atau kata kunci tertentu.
6. Menyediakan fitur ekspor data buku ke dalam format CSV.

1.4 Manfaat

Adapun manfaat yang dapat diperoleh adalah sebagai berikut:

1. Aplikasi ini membantu dalam mengelola data buku, peminjaman, pengembalian, dan perhitungan denda secara lebih praktis dan efisien. Petugas tidak perlu lagi melakukan pencatatan manual, sehingga dapat meminimalisir kesalahan dan mempercepat proses pelayanan kepada peminjam.
2. Sistem yang rapi dan terkomputerisasi membuat proses pencarian, peminjaman, dan pengembalian buku menjadi lebih cepat. Informasi tentang ketersediaan buku pun dapat diketahui secara real-time.
3. Data yang terstruktur dan dapat diekspor dalam format CSV memudahkan analisis koleksi buku, laporan bulanan, dan pengambilan keputusan terkait manajemen koleksi.

BAB II PEMBAHASAN

2.1 Alat dan Bahan

Dalam pembuatan aplikasi Sistem Manajemen Perpustakaa*n* berbasis PyQt5, digunakan beberapa alat dan bahan sebagai berikut:

1. Perangkat Keras
   Perangkat keras yang digunakan berupa laptop dengan spesifikasi Intel Core i3, Windows 11, dan RAM 8 GB
2. Perangkat Lunak
   Pengembangan aplikasi ini mengandalkan Python 3.8+ sebagai bahasa pemrograman utama, didukung oleh PyQt5 sebagai *framework* GUI untuk membangun antarmuka interaktif yang dirancang visual menggunakan Qt Designer. Untuk penyimpanan data, SQLite3 digunakan sebagai database lokal yang ringan, sementara editor Python seperti VS Code sebagai lingkungan kerja utama untuk menulis dan mengelola kode program.
3. Library Tambahan
   Sistem ini menggunakan beberapa *library* untuk melengkapi fungsionalitas aplikasi yaitu csv digunakan khusus untuk fitur ekspor data ke format file CSV, datetime menangani semua operasi terkait tanggal pinjam dan kembali buku, sedangkan os dan sys menyediakan kemampuan interaksi standar dengan sistem operasi dan lingkungan Python.
4. File Pendukung
   Sistem ini memiliki dua jenis file utama yang mendukung struktur aplikasi yaitu, file `.ui` yang dihasilkan dari Qt Designer menyimpan desain antarmuka pengguna dalam format XML, dan file `.db` merupakan database SQLite yang secara fisik menyimpan seluruh data buku, menjadi pusat manajemen informasi dalam aplikasi.

2.2 Langkah-langkah Pembuatan

1. Perencanaan Aplikasi
   Pada tahap awal pembangunan aplikasi ini, dilakukan perencanaan fitur-fitur utama yang akan diterapkan untuk memenuhi kebutuhan manajemen data buku perpustakaan. Fitur-fitur tersebut meliputi penambahan, pengeditan, dan penghapusan data buku, pencarian dan filter berdasarkan status, penghitungan denda keterlambatan, pengubahan status pinjam/tersedia, serta fitur ekspor data ke format CSV. Selain itu, aplikasi juga dirancang menampilkan statistik visual berupa jumlah total buku, buku yang dipinjam, dan buku yang tersedia.
2. Desain Antarmuka dengan Qt Designer
   Qt Designer digunakan untuk merancang antarmuka pengguna secara visual, memungkinkan pengembang menyusun elemen GUI seperti tombol, tabel, input, dan label secara drag-and-drop. Dua file utama yang dibuat adalah main_window.ui untuk jendela utama dan add_edit_buku.ui untuk dialog tambah/edit buku. Komponen seperti QTableWidget, QPushButton, QComboBox, dan QLabel ditempatkan dan disesuaikan untuk menyajikan tampilan yang intuitif dan terorganisir.
3. Implementasi Kode Python
   Setelah desain antarmuka selesai, file .ui dimuat ke dalam Python menggunakan fungsi loadUi. Selanjutnya, semua komponen antarmuka dihubungkan ke logika aplikasi

dengan PyQt5. Class utama Perpustakaan dibuat sebagai turunan dari QMainWindow, yang menangani berbagai interaksi pengguna seperti menekan tombol tambah, memilih filter, atau mengedit status buku. Masing-masing tombol disambungkan ke fungsi tertentu melalui sinyal dan slot, memungkinkan interaksi dinamis antara pengguna dan sistem.

4. Pengelolaan Database SQLite
   SQLite digunakan sebagai database lokal untuk menyimpan informasi buku. Pada saat aplikasi pertama kali dijalankan, dilakukan inisialisasi tabel buku jika belum tersedia, dengan kolom-kolom seperti id, judul, penulis, kategori, status, tanggal pinjam, tanggal kembali, dan lokasi rak. Seluruh operasi data seperti tambah, ubah, dan hapus buku dilakukan melalui perintah SQL, serta dibungkus dalam penanganan kesalahan agar aplikasi tetap stabil saat terjadi error.
5. Statistik dan Visualisasi
   Untuk memberikan gambaran cepat kepada pengguna, tiga kartu statistik dibuat menggunakan frame, masing-masing menampilkan total buku, jumlah buku yang dipinjam, dan jumlah yang tersedia. Nilai dalam kartu ini diambil dan dihitung secara dinamis setiap kali data dimuat ulang. Setiap kartu diberikan warna dan teks yang kontras agar tampil informatif sekaligus menarik secara visual, serta diletakkan dalam layout horizontal di bagian atas tampilan utama.
6. Penyempurnaan Antarmuka
   Tampilan aplikasi diperindah dengan penggunaan stylesheet CSS PyQt5 untuk menyesuaikan warna tombol, ikon, status bar, dan teks. Tombol-tombol seperti Tambah, Edit, dan Hapus diberi warna berbeda dan efek hover agar lebih interaktif. Status bar digunakan untuk menampilkan identitas pembuat aplikasi di sisi tengah bawah dengan teks berwarna hitam dan ukuran font kecil dan *bold* agar mudah dilihat tetapi tidak mengganggu tampilan utama.
7. Pengujian Aplikasi
   Setelah semua fitur terpasang, dilakukan serangkaian pengujian untuk memastikan semua fungsi berjalan sebagaimana mestinya. Uji coba mencakup menambahkan buku, mengedit informasi, menghapus data, mengubah status buku, menghitung denda otomatis berdasarkan tanggal kembali, serta mengecek hasil ekspor CSV. Jika ditemukan bug atau tampilan tidak sesuai harapan, dilakukan perbaikan baik dari sisi kode maupun antarmuka di Qt Designer.

2.3 Penjelasan Fungsi Utama

```
class Perpustakaan(QMainWindow):
    def __init__(self):
        super().__init__()
        loadUi("main_window.ui", self)
        self.setWindowTitle("Sistem Manajemen Perpustakaan")

        self.db = sqlite3.connect("perpustakaan.db")
        self.init_db()


self.gambar_judul.setPixmap(QPixmap("gambar/icon_judul.png").scaled(60, 60,
```

```
Qt.KeepAspectRatio, Qt.SmoothTransformation))

        self.export_action.triggered.connect(self.export_csv)
        self.exit_action.triggered.connect(self.close)
        self.add_action.triggered.connect(self.tambah_buku)
        self.edit_action.triggered.connect(self.edit_buku_dialog)
        self.delete_action.triggered.connect(self.hapus_buku)

self.change_status_action.triggered.connect(self.ubah_status_buku_via_menu)

        self.btn_tambah.clicked.connect(self.tambah_buku)
        self.btn_tambah.setCursor(Qt.PointingHandCursor)

        self.title_label.setAlignment(Qt.AlignCenter)

        stats_cards_layout = QHBoxLayout(self.card_container)
        stats_cards_layout.setSpacing(20)
        stats_cards_layout.setContentsMargins(0, 10, 0, 20)

        self.card_total_buku = self._create_stat_card("Total Buku", "0",
"#03254c", "#ffffff", "total_buku_val")
        stats_cards_layout.addWidget(self.card_total_buku)

        self.card_dipinjam = self._create_stat_card("Dipinjam", "0",
"#03254c", "#ffffff", "dipinjam_val")
        stats_cards_layout.addWidget(self.card_dipinjam)

        self.card_tersedia = self._create_stat_card("Tersedia", "0",
"#03254c", "#ffffff", "tersedia_val")
        stats_cards_layout.addWidget(self.card_tersedia)

        self.search_input.textChanged.connect(self.load_data)
        self.filter_status.currentIndexChanged.connect(self.load_data)
        self.filter_status.setToolTip("Filter berdasarkan status Buku")

        self.table.horizontalHeader().setSectionResizeMode(0,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(1,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(2,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(3,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(4,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(5,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(6,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(7,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(8,
QHeaderView.Interactive)
        self.table.horizontalHeader().setSectionResizeMode(9,
QHeaderView.Stretch)

        self.table.setSelectionBehavior(QTableWidget.SelectRows)
```

```
        self.table.setEditTriggers(QTableWidget.NoEditTriggers)
        self.table.horizontalHeader().setSectionsMovable(False)

        label_nama = QLabel("Dewi Agustin Asri | F1D022039")
        label_nama.setStyleSheet("color: #000000;font-
size:10pt;padding:5px;font-weight:bold")
        label_nama.setAlignment(Qt.AlignCenter)
        self.statusbar = self.statusBar()
        self.statusbar.setFixedHeight(50)
        self.statusbar.addPermanentWidget(label_nama)

        self.load_data()
```

Fungsi __init__ adalah konstruktor utama aplikasi, yang bertanggung jawab untuk menyiapkan seluruh antarmuka pengguna dan koneksi database saat program dimulai. Di sini, aplikasi memuat desain UI dari file main_window.ui, mengatur judul jendela, dan membangun koneksi ke database SQLite bernama perpustakaan.db. Fungsi ini juga memanggil init_db() untuk memastikan tabel database sudah ada. Berbagai elemen UI, seperti logo, kartu statistik, dan tabel data, diinisialisasi dan diatur tampilannya. Bagian penting lainnya adalah menghubungkan aksi pengguna yaitu klik tombol, dan perubahan teks di kolom pencarian dengan fungsi-fungsi yang relevan dalam aplikasi seperti tambah_buku, load_data, export_csv, memastikan interaktivitas. Terakhir, load_data() dipanggil untuk mengisi tabel dengan data buku yang ada saat aplikasi pertama kali dibuka.

```
    def init_db(self):
            cursor = self.db.cursor()
            cursor.execute('''
                CREATE TABLE IF NOT EXISTS buku (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    judul TEXT NOT NULL,
                    penulis TEXT NOT NULL,
                    kategori TEXT,
                    status TEXT DEFAULT 'Tersedia',
                    tanggal_pinjam TEXT,
                    tanggal_kembali TEXT,
                    lokasi_rak TEXT
                )
            ''')
            try:
                cursor.execute("ALTER TABLE buku ADD COLUMN lokasi_rak
    TEXT")
                self.db.commit()
            except sqlite3.OperationalError as e:
                if "duplicate column name" not in str(e):
                    raise
                pass
      self.db.commit()
```

Fungsi init_db adalah penanggung jawab utama untuk memastikan struktur database kita siap digunakan. Fungsi ini akan membuat tabel buku jika tabel tersebut belum ada. Tabel buku ini dirancang untuk menyimpan detail setiap buku, termasuk ID unik, judul, penulis, kategori, status (apakah tersedia atau sedang dipinjam), tanggal peminjaman, tanggal pengembalian, dan lokasi rak. Fungsi ini juga dapat mengisi kolom lokasi_rak dengan *none* jika kolom tersebut

belum ada. Ini penting agar data baru bisa disimpan tanpa mengganggu data yang sudah ada sebelumnya.

```
    def _create_stat_card(self, title, value, bg_color, text_color,
    value_object_name):
            card_widget = QWidget()
            card_layout = QVBoxLayout(card_widget)
            card_layout.setAlignment(Qt.AlignCenter)
            card_layout.setContentsMargins(20, 20, 20, 20)

            title_label = QLabel(title)
            title_label.setStyleSheet(f"font-size: 18px; font-weight: 600;
    color: {text_color}; border: none;")
            card_layout.addWidget(title_label)

            value_label = QLabel(value)
            value_label.setObjectName(value_object_name)
            value_label.setStyleSheet(f"font-size: 38px; font-weight:
    bold; color: {text_color}; margin-top: 5px; border: none;")
            card_layout.addWidget(value_label)

            card_widget.setStyleSheet(f"""
                QWidget {{
                    background-color: {bg_color};
                    border-radius: 15px;
                    border: 1px solid #ced4da;
                }}
            """)
        return card_widget
```

Fungsi _create_stat_card adalah pembantu yang sangat berguna untuk membuat tampilan ringkasan statistik yang menarik di bagian atas aplikasi. Fungsi ini mengambil beberapa parameter seperti judul, nilai yang akan ditampilkan, warna latar belakang, dan warna teks. Dengan parameter ini, ia merakit sebuah QWidget yang berisi dua label, satu untuk judul dan satu lagi untuk nilai, lalu menerapkan gaya CSS agar kartu terlihat modern dengan sudut membulat dan warna yang kontras. Ini membantu memberikan gambaran sekilas tentang status koleksi buku secara visual.

```
    Def ubah_status_db(self, id_buku, new_status, tgl_pinjam=None,
    tgl_kembali=None):
            cursor = self.db.cursor()
            try:
                if new_status == "Dipinjam":
                    cursor.execute('''
                        UPDATE buku SET status=?, tanggal_pinjam=?,
    tanggal_kembali=? WHERE id=?
                    ''', (new_status, tgl_pinjam, tgl_kembali, id_buku))
                else:
                    cursor.execute('''
                        UPDATE buku SET status=?, tanggal_pinjam=NULL,
    tanggal_kembali=NULL WHERE id=?
                    ''', (new_status, id_buku))
                self.db.commit()
                QMessageBox.information(self, "Berhasil", f"Status buku
    berhasil diperbarui menjadi '{new_status}'.", QMessageBox.Ok)
                self.load_data()
            except sqlite3.Error as e:
```

```
                QMessageBox.critical(self, "Database Error", f"Terjadi
    kesalahan saat memperbarui status buku: {e}", QMessageBox.Ok)
            except Exception as e:
          QMessageBox.critical(self, "Error", f"Terjadi kesalahan tidak
terduga: {e}", QMessageBox.Ok)
```

Fungsi ubah_status_db bertanggung jawab untuk mengubah status buku di database. Ketika sebuah buku dipinjam, fungsi ini akan memperbarui statusnya menjadi dipinjam dan mencatat tanggal peminjaman serta tanggal pengembalian yang diharapkan. Namun, jika buku dikembalikan, statusnya akan diubah kembali menjadi tersedia, dan kolom tanggal pinjam serta tanggal kembali akan dikosongkan. Fungsi ini menggunakan parameterized queries untuk mencegah serangan SQL Injection, menjadikannya lebih aman. Setelah pembaruan berhasil, sebuah pesan informasi akan muncul, dan tabel data akan dimuat ulang secara otomatis untuk mencerminkan perubahan terbaru.

```
    def ubah_status_dropdown(self, book_id, judul_buku, old_status,
    new_status_text, combo_box):
            if new_status_text == old_status:
                return

            combo_box.blockSignals(True)

            if new_status_text == "Dipinjam":
                date_edit = QDateEdit(QDate.currentDate().addDays(7))
                date_edit.setCalendarPopup(True)
                date_edit.setDisplayFormat("yyyy-MM-dd")

                msg_box_layout = QVBoxLayout()
                msg_box_layout.addWidget(QLabel("Tanggal Kembali (YYYY-MM-
    DD):"))
                msg_box_layout.addWidget(date_edit)

                msg_box = QMessageBox(self)
                msg_box.setIcon(QMessageBox.Question)
                msg_box.setWindowTitle("Konfirmasi Tanggal Kembali")
                msg_box.setText(f"Pilih tanggal pengembalian untuk
    '{judul_buku}':")
                msg_box_widget = QWidget()
                msg_box_widget.setLayout(msg_box_layout)
                msg_box.layout().addWidget(msg_box_widget)
                msg_box.setStandardButtons(QMessageBox.Ok |
    QMessageBox.Cancel)

                ret = msg_box.exec_()

                if ret == QMessageBox.Ok:
                    tgl_pinjam = datetime.now().strftime("%Y-%m-%d")
                    tgl_kembali = date_edit.date().toString("yyyy-MM-dd")
                    self._update_book_status_in_db(book_id,
    new_status_text, tgl_pinjam, tgl_kembali)
                else:
                    combo_box.setCurrentText(old_status)
            else:
                self._update_book_status_in_db(book_id, new_status_text)
      combo_box.blockSignals(False)
```

Fungsi ubah_status_dropdown adalah fungsi yang secara otomatis dipicu setiap kali pengguna mengubah status buku melalui dropdown di dalam tabel. Jika status diubah menjadi dipinjam, sebuah kotak dialog akan muncul, meminta pengguna untuk memilih tanggal pengembalian buku, dengan tanggal pinjam otomatis disetel ke tanggal hari ini. Ini memastikan bahwa semua informasi terkait peminjaman tercatat dengan lengkap. Namun, jika status diubah menjadi tfersedia, prosesnya lebih sederhana tanpa dialog tambahan. Fungsi ini penting untuk menjaga konsistensi data status buku dan memicu pembaruan yang sesuai di database.

```
    def load_data(self):
            keyword = self.search_input.text().lower().strip()
            status_filter = self.filter_status.currentText()

            query = "SELECT id, judul, penulis, kategori, status,
    tanggal_pinjam, tanggal_kembali, lokasi_rak FROM buku WHERE 1=1"
            params = []

            if keyword:
                if keyword.isdigit():
                    query += " AND id = ?"
                    params.append(int(keyword))
                else:
                    query += " AND (LOWER(judul) LIKE ? OR LOWER(penulis)
    LIKE ? OR LOWER(lokasi_rak) LIKE ?)"
                    params.append(f"%{keyword}%")
                    params.append(f"%{keyword}%")
                    params.append(f"%{keyword}%")

            if status_filter != "Semua":
                query += " AND status = ?"
                params.append(status_filter)

            cursor = self.db.cursor()
            cursor.execute(query, tuple(params))
            buku_list = cursor.fetchall()

            self.table.setRowCount(0)
            total_buku = 0
            total_dipinjam = 0
            total_tersedia = 0

            for row_data in buku_list:
                total_buku += 1
                if row_data[4] == "Dipinjam":
                    total_dipinjam += 1
                else:
                    total_tersedia += 1

                row = self.table.rowCount()
                self.table.insertRow(row)

                for col_idx in [0, 1, 2, 3]:
                    item = QTableWidgetItem(str(row_data[col_idx]) if
    row_data[col_idx] is not None else "-")
                    item.setFlags(item.flags() | Qt.TextWordWrap)
                    self.table.setItem(row, col_idx, item)
```

```
            status_combo_box = QComboBox()
            status_combo_box.addItems(["Tersedia", "Dipinjam"])
            current_status = str(row_data[4]) if row_data[4] is not
None else "Tersedia"
            status_combo_box.setCurrentText(current_status)

            try:
                status_combo_box.currentIndexChanged.disconnect()
            except TypeError:
                pass

            book_id = row_data[0]
            judul_buku = row_data[1]
            status_combo_box.currentIndexChanged.connect(
                lambda index, b_id=book_id, j_buku=judul_buku,
old_stat=current_status, cb=status_combo_box:
                self._on_table_status_dropdown_changed(b_id, j_buku,
old_stat, cb.currentText(), cb)
            )
            self.table.setCellWidget(row, 4, status_combo_box)

            item_tgl_pinjam = QTableWidgetItem(str(row_data[5]) if
row_data[5] is not None else "-")
            item_tgl_pinjam.setFlags(item_tgl_pinjam.flags() |
Qt.TextWordWrap)
            self.table.setItem(row, 5, item_tgl_pinjam)

            item_tgl_kembali = QTableWidgetItem(str(row_data[6]) if
row_data[6] is not None else "-")
            item_tgl_kembali.setFlags(item_tgl_kembali.flags() |
Qt.TextWordWrap)
            self.table.setItem(row, 6, item_tgl_kembali)

            denda = "-"
            if row_data[4] == "Dipinjam" and row_data[6]:
                try:
                    tgl_kembali_str = row_data[6]
                    if tgl_kembali_str:
                        tgl_kembali                                    =
datetime.strptime(tgl_kembali_str, "%Y-%m-%d")
                        selisih = (datetime.now() - tgl_kembali).days
                        if selisih > 0:
                            denda = f"Rp {selisih * 1000}"
                except ValueError:
                    denda = "Tanggal invalid"
            item_denda = QTableWidgetItem(denda)
            item_denda.setFlags(item_denda.flags() | Qt.TextWordWrap)
            self.table.setItem(row, 7, item_denda)

            item_lokasi_rak = QTableWidgetItem(str(row_data[7]) if
row_data[7] is not None else "-")
            item_lokasi_rak.setFlags(item_lokasi_rak.flags() |
Qt.TextWordWrap)
            self.table.setItem(row, 8, item_lokasi_rak)

            action_widget = QWidget()
```

```
                action_layout = QHBoxLayout(action_widget)
                action_layout.setContentsMargins(0, 0, 0, 0)
                action_layout.setSpacing(5)

                btn_edit_row = QPushButton("Edit")
                btn_edit_row.setStyleSheet("""
                    QPushButton {
                        background-color: #1167b1;
                        color: white;
                        border-radius: 5px;
                        padding: 5px 8px;
                        font-size: 14px;
                        border: none;
                    }
                    QPushButton:hover {
                        background-color: #2a9df4;
                    }
                """)
                btn_edit_row.setCursor(Qt.PointingHandCursor)
                btn_edit_row.clicked.connect(lambda _, r=row:
    self.edit_buku_dialog_from_row(r))
                action_layout.addWidget(btn_edit_row)

                btn_delete_row = QPushButton("Hapus")
                btn_delete_row.setStyleSheet("""
                    QPushButton {
                        background-color: #E74C3C;
                        color: white;
                        border-radius: 5px;
                        padding: 5px 8px;
                        font-size: 14px;
                        border: none;
                    }
                    QPushButton:hover {
                        background-color: #C0392B;
                    }
                    QPushButton:pressed {
                        background-color: #A93226;
                    }
                """)
                btn_delete_row.setCursor(Qt.PointingHandCursor)
                btn_delete_row.clicked.connect(lambda _, r=row:
    self.hapus_buku_from_row(r))
                action_layout.addWidget(btn_delete_row)

                self.table.setCellWidget(row, 9, action_widget)

            self.card_total_buku.findChild(QLabel,
    "total_buku_val").setText(str(total_buku))
            self.card_dipinjam.findChild(QLabel,
    "dipinjam_val").setText(str(total_dipinjam))
            self.card_tersedia.findChild(QLabel,
    "tersedia_val").setText(str(total_tersedia))

            self.table.resizeColumnsToContents()
            self.table.resizeRowsToContents()
```

```
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Interac
        tive)

self.table.horizontalHeader().setSectionResizeMode(self.table.columnCount()
- 1, QHeaderView.Stretch)
```

Fungsi load_data adalah inti tampilan informasi buku di aplikasi. Fungsi ini bertanggung jawab mengambil semua data buku dari database dan menampilkannya di QTableWidget utama. Fungsi ini dapat memfilter data berdasarkan kata kunci pencarian seperti ID, judul, penulis, atau lokasi rak dan juga berdasarkan status buku tersedia atau dipinjam. Setelah data diambil, load_data mengisi tabel baris demi baris, lengkap dengan elemen interaktif seperti dropdown untuk mengubah status buku, perhitungan denda otomatis untuk buku yang terlambat, serta tombol edit dan hapus yang muncul di setiap baris. Selain itu, fungsi ini juga memperbarui kartu statistik di bagian atas, memberikan gambaran cepat tentang total buku, buku yang dipinjam, dan buku yang tersedia.

```
        def ubah_status_buku_via_menu(self):
                selected_rows = self.table.selectedItems()
                if not selected_rows:
                    QMessageBox.warning(self, "Peringatan", "Pilih buku yang
        ingin diubah statusnya.", QMessageBox.Ok)
                    return

                row = selected_rows[0].row()
                self.ubah_status_buku(row)

            def ubah_status_buku(self, row):
                combo_box = self.table.cellWidget(row, 4)
                if isinstance(combo_box, QComboBox):
                    combo_box.showPopup()
                else:
              QMessageBox.critical(self,   "Error",   "Kolom   status   tidak
ditemukan atau bukan dropdown.", QMessageBox.Ok)
```

Fungsi ubah_status_buku_via_menu dan ubah_status_buku bekerja bersama untuk mempermudah perubahan status buku, terutama ketika diakses dari menu utama aplikasi. Ketika Anda memilih sebuah buku di tabel dan menggunakan opsi ubah status dari menu, ubah_status_buku_via_menu akan memverifikasi bahwa ada buku yang dipilih. Selanjutnya, ia akan memanggil ubah_status_buku yang akan menemukan *dropdown* status yang relevan di baris buku yang dipilih. Tujuan utamanya adalah untuk secara langsung menampilkan *pop-up* dropdown status tersebut, memungkinkan pengguna mengubah status buku dari tersedia menjadi dipinjam atau sebaliknya, tanpa perlu membuka dialog terpisah, menjadikannya lebih efisien.

```
        def edit_buku_dialog_from_row(self, row):
                id_buku = self.table.item(row, 0).text()
                judul_lama = self.table.item(row, 1).text()
                penulis_lama = self.table.item(row, 2).text()
                kategori_lama = self.table.item(row, 3).text()
                lokasi_rak_lama = self.table.item(row, 8).text()

                dialog     =     BookFormDialog(judul_lama,     penulis_lama,
        kategori_lama, lokasi_rak_lama, self)
```

```
            if dialog.exec_() == QDialog.Accepted:
                new_data = dialog.get_data()
                ok_judul = new_data["judul"]
                ok_penulis = new_data["penulis"]
                ok_kategori = new_data["kategori"]
                ok_lokasi_rak = new_data["lokasi_rak"]

                if not ok_judul or not ok_penulis:
                    QMessageBox.warning(self, "Input Error", "Judul dan
Penulis harus diisi.", QMessageBox.Ok)
                    return

                cursor = self.db.cursor()
                try:
                    cursor.execute('''
                        UPDATE buku SET judul=?, penulis=?, kategori=?,
lokasi_rak=? WHERE id=?
                    ''',      (ok_judul,      ok_penulis,      ok_kategori,
ok_lokasi_rak, id_buku))
                    self.db.commit()
                    QMessageBox.information(self, "Berhasil", "Detail buku
berhasil diperbarui.", QMessageBox.Ok)
                    self.load_data()
                except sqlite3.Error as e:
                    QMessageBox.critical(self, "Database Error", f"Terjadi
kesalahan saat memperbarui buku: {e}", QMessageBox.Ok)

        def edit_buku_dialog(self):
            selected_rows = self.table.selectedItems()
            if not selected_rows:
                QMessageBox.warning(self, "Peringatan", "Pilih buku yang
ingin diedit dari tabel.", QMessageBox.Ok)
                return

          self.edit_buku_dialog_from_row(selected_rows[0].row())
```

Fungsi edit_buku_dialog dan edit_buku_dialog_from_row adalah fitur utama untuk memperbarui informasi buku yang sudah ada. Ketika Anda memilih buku di tabel dan memilih opsi edit, edit_buku_dialog akan memastikan ada buku yang terpilih. Kemudian, edit_buku_dialog_from_row akan mengambil semua data buku yang ada dari baris tabel tersebut dan membukanya di BookFormDialog, sebuah dialog khusus untuk menambah/mengedit buku. pengguna dapat mengubah detail seperti judul, penulis, kategori, dan lokasi rak. Setelah pengguna mengonfirmasi perubahan, aplikasi akan memvalidasi input dan menyimpan pembaruan ke database menggunakan *parameterized queries* untuk keamanan. Sebuah pesan sukses akan ditampilkan, dan tabel buku akan diperbarui secara otomatis.

```
    def hapus_buku_from_row(self, row):
            id_buku = self.table.item(row, 0).text()
            judul_buku = self.table.item(row, 1).text()

            reply = QMessageBox.question(self, "Konfirmasi Hapus",
                                         f"Anda yakin ingin menghapus buku
'{judul_buku}'?",
                                         QMessageBox.Yes | QMessageBox.No,
QMessageBox.No)
```

```
            if reply == QMessageBox.Yes:
                cursor = self.db.cursor()
                try:
                    cursor.execute("DELETE   FROM   buku   WHERE   id=?",
(id_buku,))
                    self.db.commit()
                    QMessageBox.information(self,    "Berhasil",    "Buku
berhasil dihapus!", QMessageBox.Ok)
                    self.load_data()
                except sqlite3.Error as e:
                    QMessageBox.critical(self, "Database Error", f"Terjadi
kesalahan saat menghapus buku: {e}", QMessageBox.Ok)

        def hapus_buku(self):
            selected_rows = self.table.selectedItems()
            if not selected_rows:
                QMessageBox.warning(self, "Peringatan", "Pilih buku yang
ingin dihapus dari tabel.", QMessageBox.Ok)
                return
          self.hapus_buku_from_row(selected_rows[0].row())
```

Fungsi hapus_buku dan hapus_buku_from_row adalah fitur penting untuk menghapus entri buku dari sistem. Ketika pengguna ingin menghapus buku, hapus_buku memastikan bahwa pengguna telah memilih buku dari tabel. Kemudian, hapus_buku_from_row akan mengambil ID dan judul buku yang dipilih dan akan menampilkan kotak pesan konfirmasi. Ini adalah langkah keamanan untuk mencegah penghapusan yang tidak disengaja. Jika pengguna mengonfirmasi penghapusan, buku tersebut akan dihapus secara permanen dari database, dan tabel akan diperbarui untuk mencerminkan perubahan tersebut, disertai pesan keberhasilan.

```
    def tambah_buku(self):
            dialog = BookFormDialog(parent=self)
            if dialog.exec_() == QDialog.Accepted:
                new_data = dialog.get_data()
                judul = new_data["judul"]
                penulis = new_data["penulis"]
                kategori = new_data["kategori"]
                lokasi_rak = new_data["lokasi_rak"]

                if not judul or not penulis:
                    QMessageBox.warning(self,  "Input  Error",  "Judul  dan
Penulis harus diisi.", QMessageBox.Ok)
                    return

                cursor = self.db.cursor()
                try:
                    cursor.execute('''
                        INSERT   INTO   buku   (judul,   penulis,   kategori,
status, lokasi_rak)
                        VALUES (?, ?, ?, 'Tersedia', ?)
                    ''', (judul, penulis, kategori, lokasi_rak))
                    self.db.commit()
                    QMessageBox.information(self,    "Berhasil",    "Buku
berhasil ditambahkan!", QMessageBox.Ok)
                    self.load_data()
                except sqlite3.Error as e:
              QMessageBox.critical(self,   "Database   Error",   f"Terjadi
```

```
kesalahan saat menambahkan buku: {e}", QMessageBox.Ok)
```

Fungsi tambah_buku adalah cara utama untuk menambahkan koleksi buku baru ke database. Saat dipanggil, fungsi ini akan membuka sebuah dialog BookFormDialog yang menyediakan formulir untuk pengguna mengisi detail buku seperti judul, penulis, kategori, dan lokasi rak. Setelah pengguna mengisi informasi yang diperlukan dan mengkli ok, fungsi ini akan memvalidasi input dasar, memastikan judul dan penulis tidak kosong sebelum menyimpan data buku baru ke database dengan status awal tersedia. Setelah penambahan berhasil, sebuah pesan informasi akan muncul, dan tabel buku akan diperbarui secara otomatis untuk menampilkan entri terbaru

```
      def export_csv(self):
              path,  _  =  QFileDialog.getSaveFileName(self,  "Simpan  CSV",
      "data_buku.csv", "CSV Files (*.csv)")
              if path:
                  cursor = self.db.cursor()
                  cursor.execute("SELECT  id,  judul,  penulis,  kategori,
      status, tanggal_pinjam, tanggal_kembali, lokasi_rak FROM buku")
                  data = cursor.fetchall()
                  try:
                      with open(path, "w", newline="", encoding="utf-8") as
      f:
                          writer = csv.writer(f)
                          writer.writerow(["ID",     "Judul",     "Penulis",
      "Kategori",  "Status",  "Tgl  Pinjam",  "Tgl  Kembali",  "Denda  (Saat
      Export)", "Lokasi Rak"])

                          for row_data in data:
                              denda = "-"
                              if row_data[4] == "Dipinjam" and row_data[6]:
                                  try:
                                      tgl_kembali_str = row_data[6]
                                      if tgl_kembali_str:
                                          tgl_kembali                      =
      datetime.strptime(tgl_kembali_str, "%Y-%m-%d")
                                          selisih   =   (datetime.now()   -
      tgl_kembali).days
                                          if selisih > 0:
                                              denda = f"Rp {selisih * 1000}"
                                  except ValueError:
                                      denda = "Tanggal invalid"

                              row_to_write  =  list(row_data[:7])  +  [denda,
      row_data[7]]
                              writer.writerow(row_to_write)

                      QMessageBox.information(self, "Export Berhasil", "Data
      berhasil diekspor ke CSV!", QMessageBox.Ok)
                  except IOError as e:
                 QMessageBox.critical(self,  "Error",  f"Tidak  dapat  menulis
file: {e}", QMessageBox.Ok)
```

Fungsi export_csv digunakan untuk menyimpan seluruh koleksi buku ke dalam file CSV, yang sangat berguna untuk pelaporan atau analisis eksternal. Saat dipicu, sistem akan membuka dialog simpan file di mana pengguna dapat menentukan nama dan lokasi penyimpanan. Setelah itu, fungsi ini akan mengambil semua data buku dari database, termasuk informasi denda yang

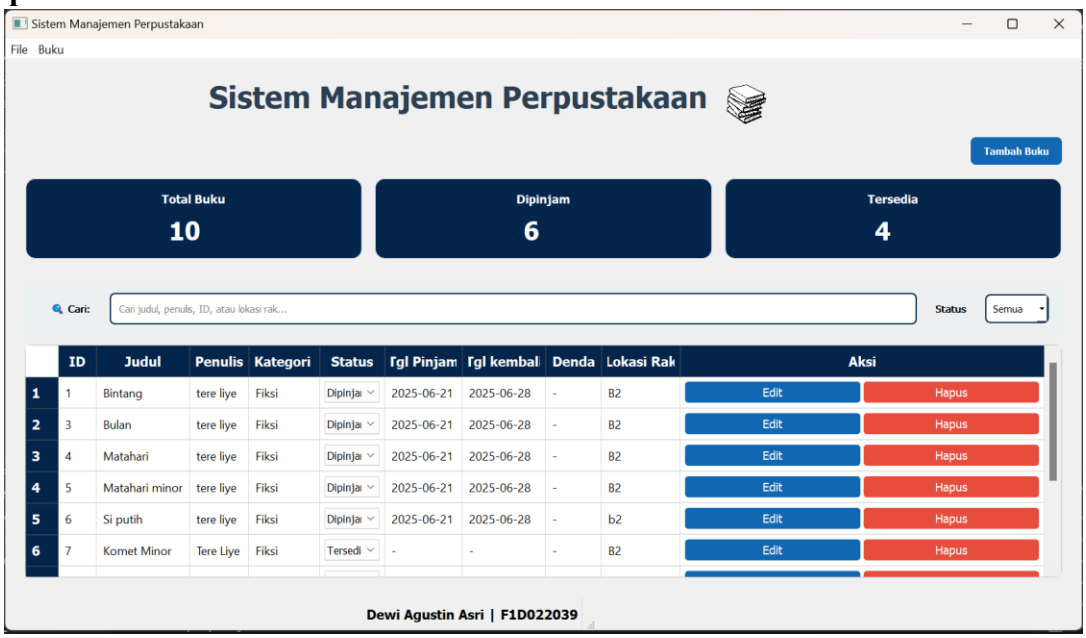
dihitung pada saat ekspor, dan menuliskannya ke file CSV yang ditentukan. Ini memastikan bahwa semua detail penting dari perpustakaan Anda dapat dengan mudah diakses dan digunakan di luar aplikasi.

```
class BookFormDialog(QDialog):
    """
    Dialog untuk menambah atau mengedit detail buku.
    Memuat UI dari 'add_edit_buku.ui'.
    """
    def __init__(self, judul="", penulis="", kategori="",
lokasi_rak="", parent=None):
        super().__init__(parent)

        # Memuat UI dari file .ui
        loadUi("add_edit_buku.ui", self)
        self.setWindowTitle("Tambah/Edit Buku")

        # Mengisi field dengan data yang ada jika dalam mode edit
        self.edit_judul.setText(judul)
        self.edit_penulis.setText(penulis)
        self.edit_kategori.setCurrentText(kategori)
        self.edit_lokasi_rak.setText(lokasi_rak)

        # Mengatur lebar label dan placeholder untuk input
        self.judul_label.setFixedWidth(80)
        self.edit_judul.setPlaceholderText("Judul Buku")

        self.penulis_label.setFixedWidth(80)
        self.edit_penulis.setPlaceholderText("Penulis")

        self.kategori_label.setFixedWidth(80)
        self.edit_kategori.setToolTip("Pilih kategori buku")

        self.lokasi_rak_label.setFixedWidth(80)
        self.edit_lokasi_rak.setPlaceholderText("Lokasi Rak (e.g., A1,
B2)")

        # Menghubungkan tombol OK/Cancel
        self.buttonBox.accepted.connect(self.accept)
        self.buttonBox.rejected.connect(self.reject)

    def get_data(self):
        """
        Mengambil data yang diisi dari formulir dialog.
        Menggunakan .strip() untuk membersihkan spasi.
        """
        return {
            "judul": self.edit_judul.text().strip(),
            "penulis": self.edit_penulis.text().strip(),
            "kategori": self.edit_kategori.currentText(),
            "lokasi_rak": self.edit_lokasi_rak.text().strip()
  }
```

Kelas BookFormDialog adalah sebuah jendela dialog kecil yang dirancang khusus untuk menambah atau mengedit detail buku. Ketika pengguna ingin menambahkan buku baru atau mengubah informasi buku yang sudah ada, dialog inilah yang akan muncul. Kelas ini terdiri dari buat buah fungsi yaitu fungsi __init__ yang merupakan konstruktur dari kelas ini. Fungi ini

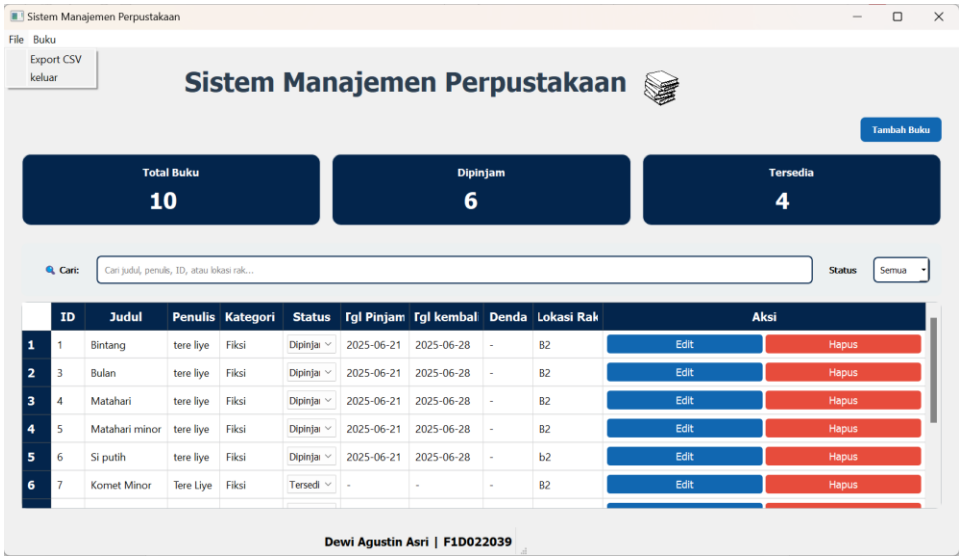
memuat desain antarmuka penggunanya dari file add_edit_buku.ui. Jika dialog ini dipanggil untuk mengedit buku yang sudah ada, konstruktor akan secara otomatis mengisi kolom-kolom input seperti judul, penulis, kategori, dan lokasi rak, dengan data buku yang relevan. Selain itu, beberapa penyesuaian visual dilakukan, seperti mengatur lebar label dan menambahkan *placeholder text* di kolom input untuk memandu pengguna. Yang penting, tombol ok dan Cancel pada dialog dihubungkan sehingga saat pengguna menekan ok, dialog akan mengonfirmasi input dan menutup self.accept(), sementara menekan cancle akan membatalkan aksi self.reject(). Kemudian fugsi yang kedua adalah get_data(). Fungsinya digunakan untuk mengambil semua teks yang telah diisi pengguna dari kolom-kolom input dialog judul, penulis, kategori, dan lokasi rak dan mengembalikannya dalam bentuk *dictionary*. Metode ini juga secara otomatis membersihkan spasi berlebih di awal atau akhir teks menggunakan .strip(), memastikan data yang disimpan ke database rapi dan konsisten.

2.4 Tampilan Antar Muka



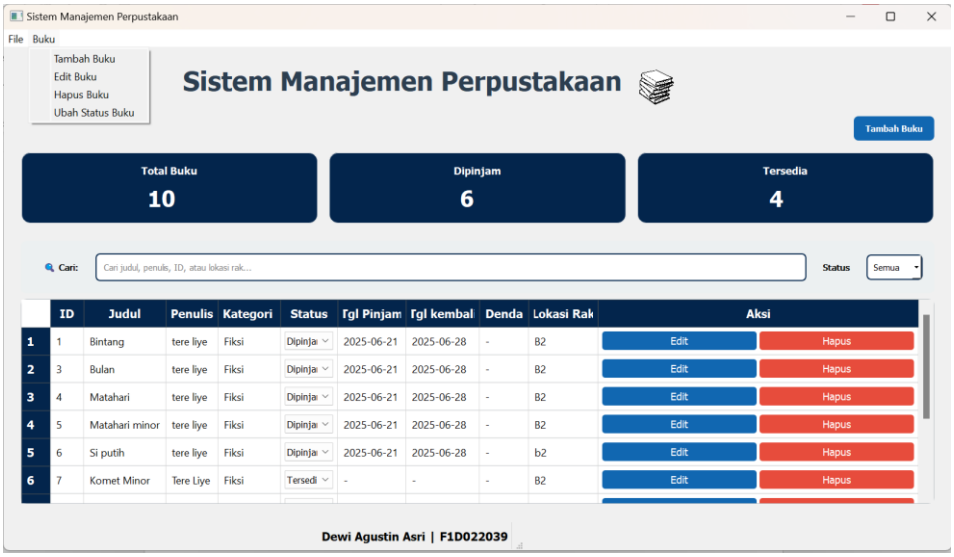
Gambar 1.1 Dashboard Aplikasi

Gambar 1.1 diatas merupakan tampilan awal Ketika sistem dibuka. Pada bagian atas merupakan judul sistem yaitu sistem manajemen perpustakaan. Di bawahnya terdapat tombol berwarna biru yang berfungsi untuk menambahkan data buku baru ke sistem. Sistem ini menampilkan ringkasan statistic buku, yaitu berapa total buku yang ditambahkan, berapa buku yang dipinjam dan berapa buku yang tersedia. Terdapat juga beberapa fitur untuk memudahkan pengguna memfilter buku seperti pencarian, yang dapat memfilter buku berdasarkan judul, id, penulis atau dimana rak buku itu berada. Pengguna juga dapat melakukan filtering berdasarkan status buku tersedia atau dipinjam. Data buku yang telah dimasukkan ditampilkan dalam format tabel yang terdiri dari ID, judul buku, penulis, kategori, status buku yang dapat dirubah jika dilakukan peminjaman, tanggal pinjam, tanggal kembali, denda, Lokasi rak, dan kolom aksi yang terdapat fitur edit dan hapus pada setiap barisnya. Pada bagian bawah sistem terdapat nama pembuat sistem, hanya sebagai indetitas.



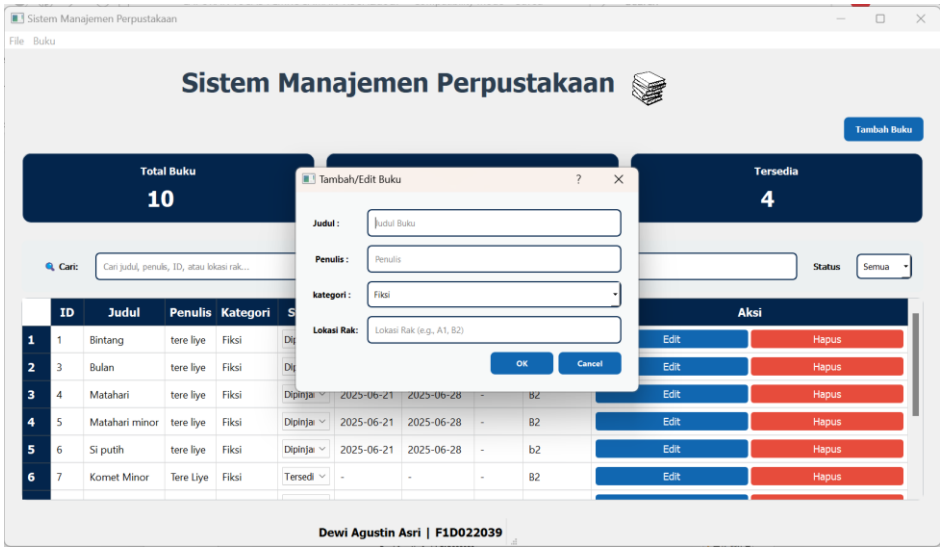
Gambar 2.2 Menu File

Pada **Gambar 2.2** di atas, sistem manajamen perpustakaan memili menu file yang terdiri dari fitur export csv dan keluar. Jika pengguna menekan export csv, data yang telah dimasukkan ke dalam sistem akan disimpan dalam format file CSV(Comma Sperated Values). Hal ini berguna unutk menganlisis data atau sebagai pencadangan data. Sedangkan jika pengguna memilih fitur keluar, jendel sistem akan tertutup dan mengakhiri sesi pengguna.



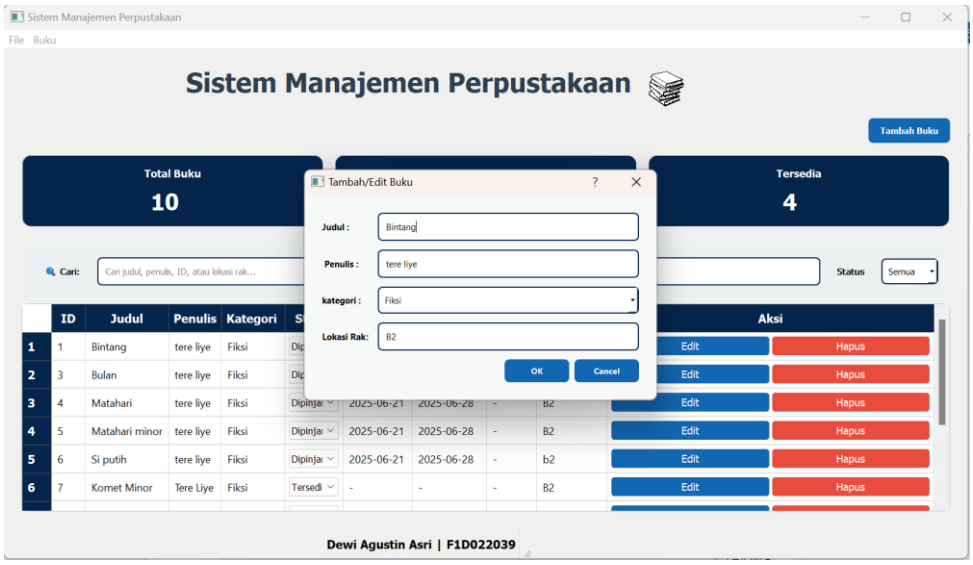
Gambar 2.3 Fitur Menu Buku

Pada **Gambar 2.3** diatas, sistem ini memiliki fitur menu buku yang terdiri dari submenu tambah buku, edit buku, hapus buku, dan ubah status buku. Tambah buku digunakan untuk menambahakn entri buku baru ke dalam system. Edit buku digunakan untuk mengubah detail atau informasi yang buku yang telah terdaftar. Hapus buku memungkian pengguna untuk menghapus buku yang sudah ada didatabase. Dan fitur ubah status buku menungkinkan pengguna mengubah status buku tersedia atau dipinjam.



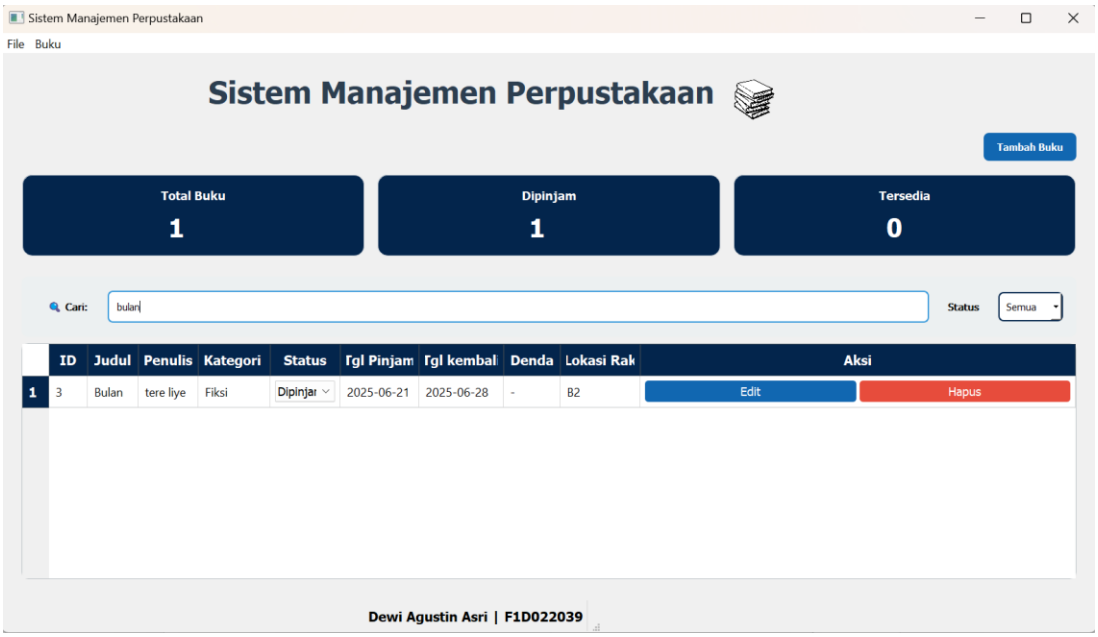
Gambar 2.4 Dialog tambah Buku

ketika pengguna menekan tombol tambah buku, sistem akan menampilkan dialog tambah buku seperti pada Gambar 2.4 di atass. Pengguna diharuskan mengisi data judul buku, penulis buku, kategori buku, dan rak mana tempat buku tersimpan.



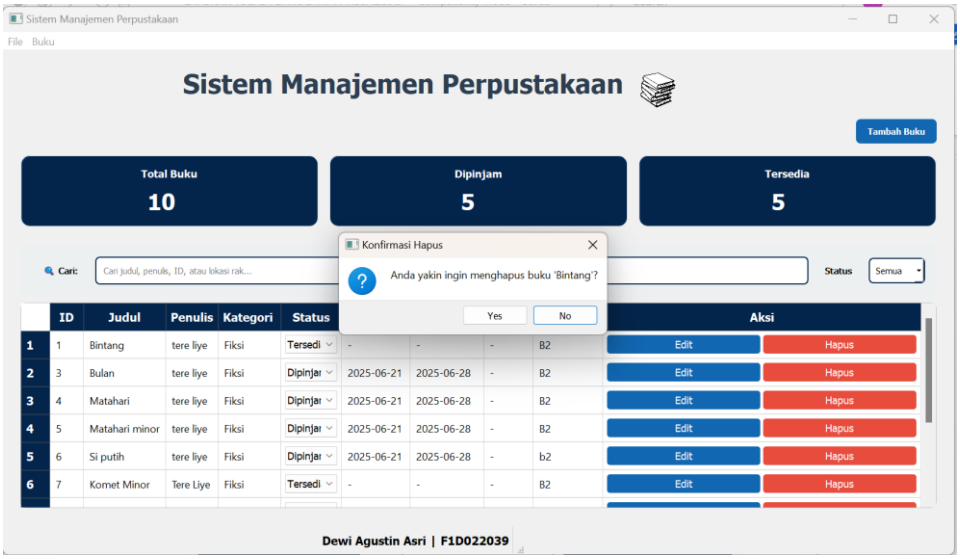
Gambar 2.5 Dialog Edit Buku

Ketika pengguna memekan aksi edit pada data buku,sistem akan menampilkan dialog yang berisi data buku yang dapat diuabah seperti terlihat pada **Gambar 2.5** diatas yang menampilkan dialog data buku dengan judul bintang.



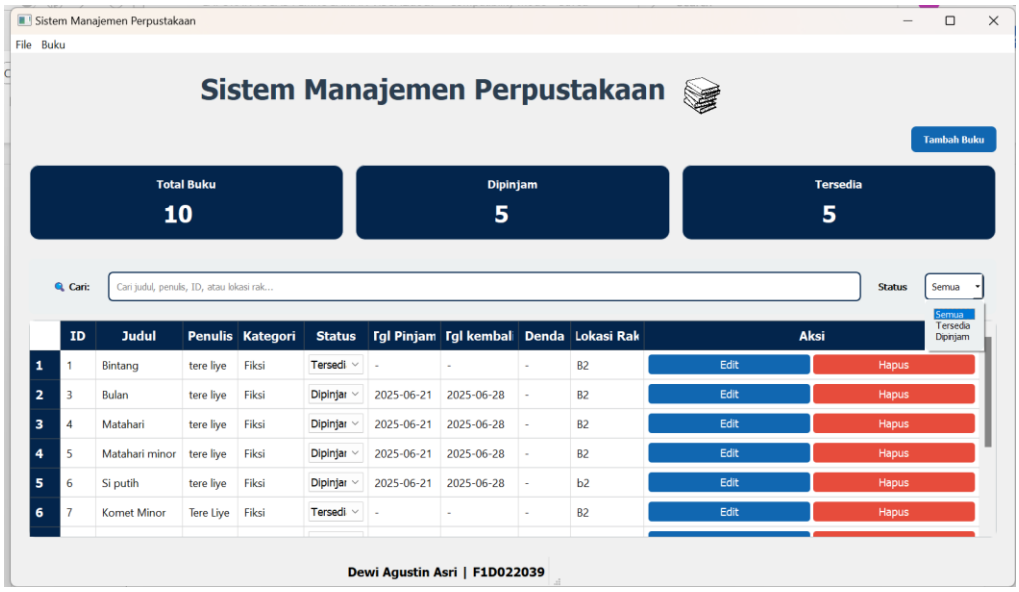
Gambar 2.6 Pencarian Buku

Sistem manjemen perpustakaan ini memiliki fitur pencarian data. Pengguna dapat mencari buku berdasarkan judul, id buku, nama penulis buku dan lokasi rak buku



Gambar 2.7 Hapus Buku

Sistem manajemen perpustakan memiliki fitur hapus data pada setiap baris data bukunya. Ketika pengguna ingin menghapus data buku, data tidak langsung terhapus, sistem telebih dahulu menampilkan konfirmasi apakah pengguna yakin akan menghapus data tersebut atau tidak.



Gambar 2.8 Filter Status

Sistem manajemen perpustakaan ini dapat melakuakn filtering data berdasarkan status data yaitu berdasarkan status buku yang tersedia atau buku yang dipinjam seperti pada tampilan Gambar 2.8 di atas

BAB III PENUTUP

3.1 Kesimpulan

Aplikasi *Sistem Manajemen Perpustakaan* yang dikembangkan dengan PyQt5 dan SQLite berhasil menyediakan solusi sederhana dan efektif untuk mengelola data buku dalam sebuah perpustakaan. Aplikasi ini mencakup fitur-fitur utama seperti penambahan, pengeditan, penghapusan, pencarian, serta pengelompokan status buku menjadi tersedia dan dipinjam. Selain itu, fitur ekspor data ke file CSV, penghitungan otomatis denda keterlambatan, serta tampilan statistik jumlah buku semakin menambah kemudahan dalam pengelolaan koleksi buku. Antarmuka yang interaktif dan responsif menjadikan aplikasi ini mudah digunakan bagi pengguna tanpa latar belakang teknis.

3.2 Saran

Pengembangan aplikasi ini masih dapat ditingkatkan lebih lanjut. Beberapa saran pengembangan ke depan antara lain: (1) menambahkan sistem autentikasi pengguna agar hanya admin yang bisa mengakses fitur manajemen buku; (2) mengembangkan sistem peminjaman anggota secara terpisah dengan integrasi database pengguna; (3) menambahkan fitur backup dan restore data otomatis; serta (4) memperindah antarmuka menggunakan ikon, animasi transisi, atau tema gelap (dark mode). Dengan pengembangan lebih lanjut, aplikasi ini dapat menjadi sistem perpustakaan yang lebih profesional dan siap digunakan di lingkungan sekolah maupun instansi lainnya.