LAPORAN PROYEK MATA KULIAH PEMROGRAMAN DASAR

SEMESTER GANJIL 2024/2025

Ghiyats Inventory

Anggota Tim

Muhammad Ghiyats AR Rahmaniey

22/503896/TK/55089

Departemen Teknik Elektro dan Teknologi Informasi

Fakultas Teknik

Universitas Gadjah Mada

2025

Link Github : https://github.com/Gaetrix/Tugas_Progdas_PascaUTS/tree/main/UAS/uas-progdas-ghiyats

1. DESKRIPSI APLIKASI

Ghiyats Inventory adalah aplikasi inventory tracking yang dibuat dengan tujuan untuk memudahkan dalam melakukan pencatatan barang yang masuk dan dipindahkan secara berkala. Applikasi ini merupakan aplikasi desktop *cross-platform*, atau dengan kata lain dapat memiliki fungsionalitas aplikasi yang dapat bekerja di *MAC OS, Windows* dan *Linux*. Fungsionalitas aplikasi tersebut digunakan untuk menyelesaikan masalah user:

- Kesulitan untuk melakukan pencatatan pada inventory yang masuk atau dipindahkan **(P.1)**
- Catatan yang ditulis dalam kertas masih dapat mudah rusak dan hilang **(P.2)**
- Kesulitan dalam mencari nama barang spesifik karena barang yang tercatat masuk bisa saja dalam jumlah yang sangat banyak dan bervariatif **(P.3)**

Masalah yang dialami oleh user ini akan diselesaikan oleh aplikasi Phone Book App, yang memiliki dapat melakukan:

- Pencatatan data inventory yang mendukung operasi *CRUD* (Create, Read, Update dan Delete) secara dinamis dan dapat disimpan secara lokal [Mengatasi **P.1**]
- Penyimpanan data secara local maupun tidak, feature Export ini dibuat untuk memudahkan user dalam mencari data yang mereka simpan dan gunakan kembali sewaktu waktu ada pelaporan inventori [Mengatasi **P.2**]
- View Data secara optimal dan efisien dengan feature search [Mengatasi **P.3**].

2. ANALISIS KEBUTUHAN

Berdasarkan kebutuhan user pada **P.1** dan **P.2** maka akan dibuat yang dapat mendukung aplikasi untuk melakukan *CRUD* dan *Load* data user sehingga masalah user dapat diselesaikan. Aplikasi akan sering bersinggungan dengan data dan pada kasus kali ini aplikasi akan menyimpan data bertipe *.csv* untuk memudahkan user dalam menyimpan dan mengorganisir data mereka.

2.1. Kebutuhan Fungsional

Tabel 1 Kebutuhan Fungsional

Req-ID	Kebutuhan Fungsional	Deskripsi
F-01	Add new item	Menambahkan data inventory baru ke sistem melalui form input.
F-02	Search Item	Mencari item dengan id ataupun nama dari data yang sudah terrecord
F-03	Edit item data	Memperbarui data inventory yang telah dipilih melalui fitur search.
F-04	Delete item data	Menghapus data ineventory yang dipilih dari fitur search.

F-05	Export data to CSV	Menyimpan seluruh data pelanggan yang ada ke dalam file CSV.
F-06	Dynamic Row-Column alignment	Mengatur ulang ukuran table secara otomatis setelah terdapat data/item baru yang masuk. *(opsional, UX-improvement)*

2.2. Kebutuhan Non Fungsional

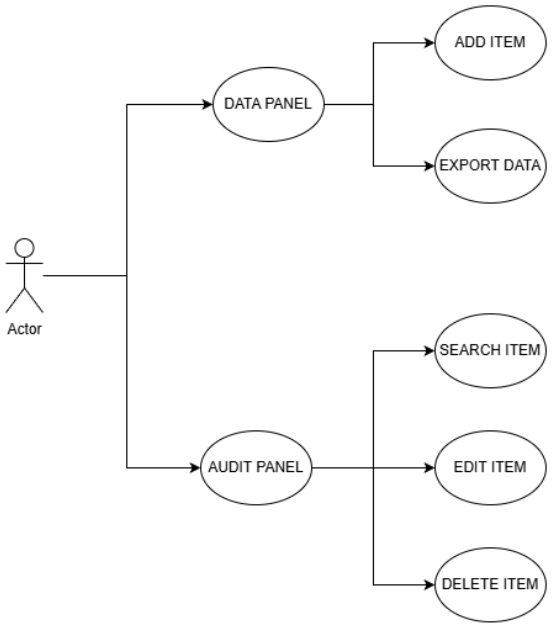
Dalam pengembangan aplikasi ini, tim menggunakan bahasa pemrograman **C++ Versi 17 ke atas** karena kemampuannya dalam menangani performa tinggi dan kontrol memori yang efisien, yang sangat penting untuk aplikasi desktop. Untuk antarmuka pengguna grafis (GUI), dipilih **wxWidgets versi 3.2 ke atas**, yang memungkinkan pengembangan antarmuka lintas platform dengan tampilan native. Sistem build yang digunakan adalah **CMake**, yang memudahkan manajemen proyek dan kompatibel dengan IDE **VSCode**. Untuk perancangan sistem, tim menggunakan **DrawIo** sebagai alat bantu visualisasi dan dokumentasi struktur kelas serta alur interaksi antar komponen.

Tim developer hanya terdiri dari 1 anggota, yaitu **Muhammad Ghiyats AR Rahmaniey** mahasiswa Teknik Elektro semester 7 seorang diri. Saya bertanggung jawab atas perancangan dan implementasi modul **CRUD** menggunakan `wxListGrid`, termasuk fungsi tambah, edit, hapus, dan simpan data, pengembangan modul **pembaca file CSV** menggunakan `wxGrid`, serta integrasi file CSV dengan UI agar pengguna dapat melihat isi file secara tabular. Sehingga perancangan struktur aplikasi dapat dilakukan secara terintegrasi dan modular serta dapat diimplementasikan lewat `wxNotebook`.

Keterbatasan, proyek yang dikembangkan adalah batas waktu pengerjaan yang ketat serta sumber daya terbatas, baik dalam hal waktu coding, pengujian lintas platform, maupun dokumentasi. Selain itu, penggunaan **C++** dan **wxWidgets** memerlukan manajemen memori eksplisit dan debugging yang teliti, yang menjadi tantangan tersendiri. Penggunaan **UML IDE** memerlukan lisensi sehingga tidak memungkinkan membuat class diagram yang maksimal dan mengenerate code berdasarkan **UML IDE** tersebut. Meskipun demikian, dengan pembagian kerja yang jelas dan pemanfaatan tools yang sesuai, tim mampu membangun aplikasi yang modular, responsif, dan mudah dikembangkan lebih lanjut.

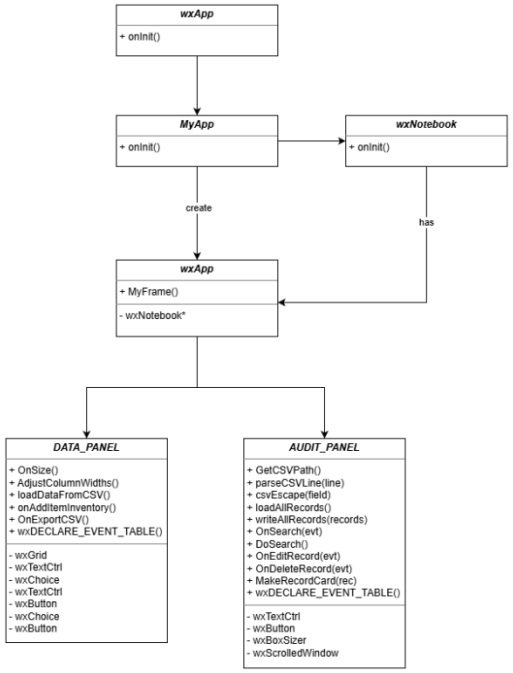
3. PEMODELAN DAN PERANCANGAN SISTEM

3.1. Use Case Diagram



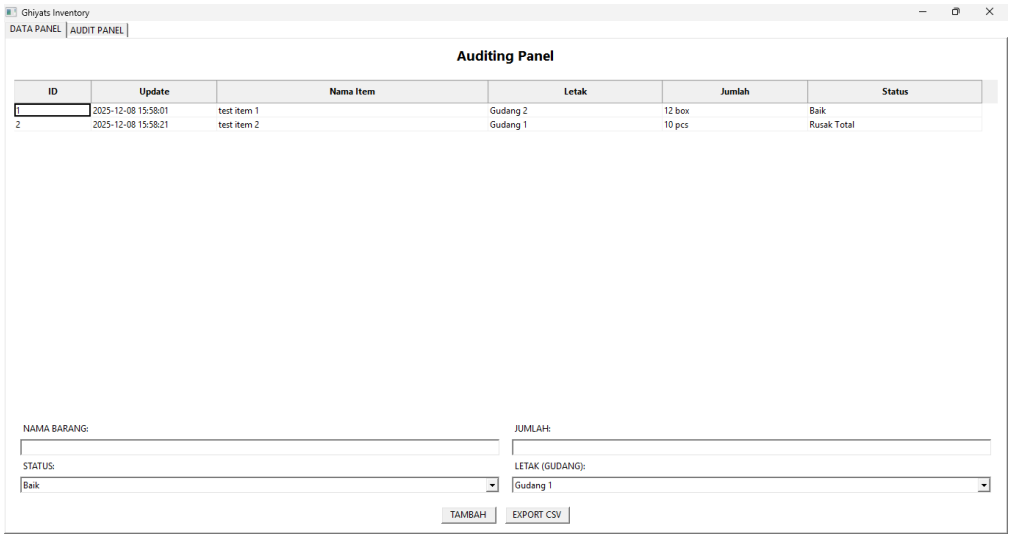
Gambar 1 Use Case Diagram

3.2. Class Diagram



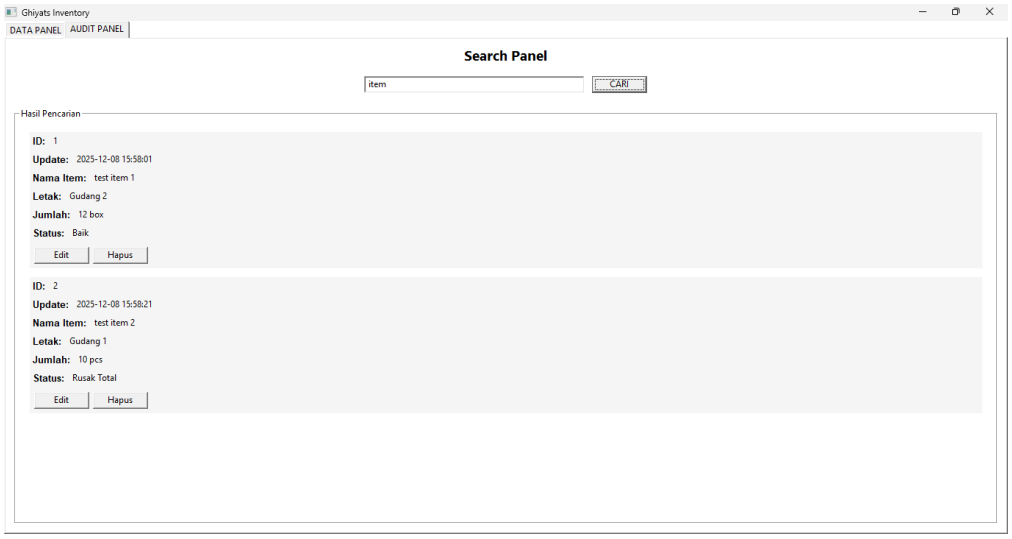
Gambar 2 Class Diagram

3.3. Perancangan User Interface



4. Gambar 3 Tab DATA PANEL

Tab ini dirancang untuk memudahkan pengguna dalam mengelola data inventory secara interaktif. Di bagian bawah terdapat empat input field secara vertikal dan horizontal: Nama barang, Jumlah (beserta satuannya), Letak dan dan Status (kondisi barang). Di bawahnya terdapat dua tombol: Tambah dan Export. Tombol Tambah memiliki fungsi spesifik untuk menambahkan item baru sedangkan Tombol Export memiliki fungsi untuk mengeskport hasil data. Tabel daftar kontak ditampilkan menggunakan wxGridCtrl yang menampilkan semua entri secara real-time. Input divalidasi agar tidak ada data kosong yang disimpan.



Gambar 4 Tab AUDIT PANEL

Pada tab ini, pengguna dapat mencari item yang sudah di masukan kedalam data .csv sebelumnya. Ketika data yang dicari didapat meka pengguna dapat menghapus data tersebut atau bahkan mengedit data teresebut. Hasil pencarian Item akan memasukan semua hasil pencarian dan merender nama, tanggal, jumlah dan letak ke component semacam card. Hal ini dihandle oleh method MakeRecordCard. Tujuan dari hal ini adalah memudahkan pengguna dalam melakukan validasi dan check data yang akan dihapus ataupun di edit. Pengguna tidak mungin dapat menghapus data yang keberadaannya null atau nihil. Dengan metode ini pengguna mendapatkan pengalaman auditing yang baik.

4.1. Transformasi Class Diagram Menjadi Kode Program

Pada *Gambar 2* terdapat implementasi OOP dengan pilar inheritance, abstraction, encapsulation dan polymorphism. OOP digunakan untuk membangun aplikasi yang modular dan mudah untuk dimaintenence atau bahkan untuk dikembangkan lebih lanjut lagi. Pada project kali saya mengembangkan aplikasi dengan library wx dari C++ untuk mengembangkan aplikasi berbasis antar muka.

Inheritance digunakan untuk membangun antarmuka utama (frame) dan panel-panel aplikasi dengan cara menurunkan kelas-kelas dari komponen wxWidgets:

- MyFrame mewarisi dari wxFrame, menjadi jendela utama aplikasi.

Gambar 5 Inheritance wxFrame ke MyFrame

- DATA_PANEL dan AUDIT_PANEL mewarisi dari wxPanel, yang memungkinkan mereka digunakan sebagai tab (halaman) di dalam wxWindow.

```
#ifndef DATA_PANEL_H
#define DATA_PANEL_H

class DATA_PANEL : public wxPanel
{
public:
    DATA_PANEL(wxWindow *parent);

private:
    ....
};

#endif // DATA_PANEL_H
```

Gambar 6 Inheritance wxPanel ke DATA_PANEL

```
#ifndef AUDIT_PANEL_H
#define AUDIT_PANEL_H

class AUDIT_PANEL : public wxPanel
{
public:
    AUDIT_PANEL(wxWindow* parent);

private:
    ....
};

#endif // AUDIT_PANEL_H
```

Gambar 7 Inheritance wxPanel ke AUDIT_PANEL

Dengan pewarisan ini, masing-masing panel mendapatkan semua fitur dasar dari wxPanel, dan dapat menambahkan fungsionalitas sesuai kebutuhan spesifik.

Encapsulation diterapkan dengan menyembunyikan detail implementasi dan menjaga data internal tetap terlindungi dari luar kelas:

Gambar 8 Implementasi Private pada pilar Encapsulation

- Objek-objek GUI seperti amtCtrl, grid, statusCtrl, dan wxDECLARE_EVENT_TABLE adalah anggota privat dari kelas panel, tidak diakses langsung dari luar.
- Fungsi seperti OnSearch () dan OnEditRecord () bersifat internal untuk AUDIT_PANEL, dan tidak diekspos ke luar.

Abstraksi dicapai dengan memisahkan antarmuka pengguna dari logika internal:

Gambar 9 Implementasi Acstraction Pada Method DATA_PANEL & AUDIT_PANEL

- Pengguna cukup menekan tombol dan melihat hasil (misalnya menambah atau mengeksport data), tanpa mengetahui cara penyimpanan, pemrosesan teks, atau pembentukan grid/list.
- Kelas DATA_PANEL dan AUDIT_PANEL menyederhanakan interaksi pengguna ke dalam fungsi spesifik seperti OnSearch, OnEditRecord, OnDeleteRecord, dan onAddItemInventory.

Polimorfisme tidak terlalu diimplementasikan dalam project ini, namun konsepnya tetap terpakai melalui pewarisan *wxWidgets*:

```
MyFrame::MyFrame()
    : wxFrame(NULL, wxID_ANY, "Ghiyats Inventory", wxDefaultPosition,
wxSize(980, 600)) {

    SetBackgroundColour(*wxWHITE);

    wxNotebook* notebook = new wxNotebook(this, wxID_ANY);

    notebook->AddPage(new DATA_PANEL(notebook), "DATA PANEL");
    notebook->AddPage(new AUDIT_PANEL(notebook), "AUDIT PANEL");
}
```

Gambar 10 Implementasi Polymorphism Pada Pemanngilan AddPage dengan CRUD_PANEL & LOAD_PANEL

- Semua komponen panel (wxPanel) bisa diperlakukan sebagai objek dari superclass wxWindow oleh wxWidgets.
- wxNotebook->AddPage() menerima argumen bertipe wxWindow*, sehingga dapat menampung objek dari kelas turunan manapun seperti DATA_PANEL atau AUDIT_PANEL.

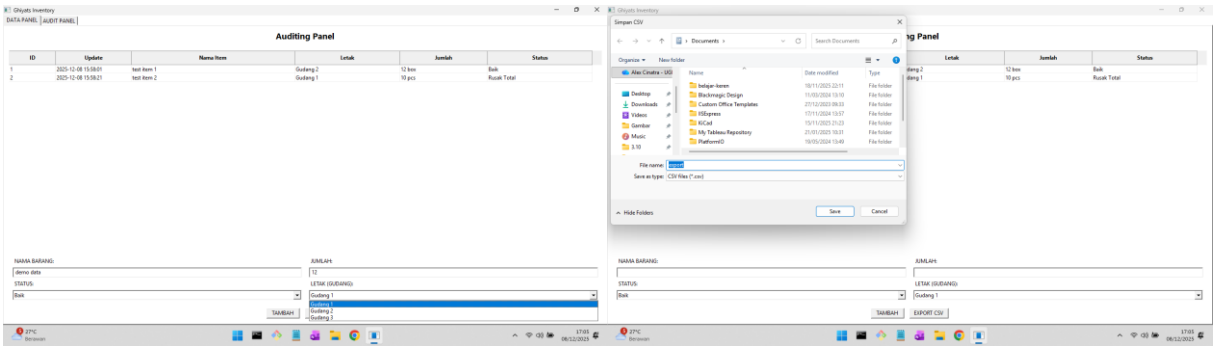
wxNotebook hanya tahu bahwa itu adalah wxWindow, namun tetap bisa menangani berbagai jenis panel, merupakan implementasi dari runtime polymorphism.

5. IMPLEMENTASI APLIKASI

Pada project kali ini berhasil mengembangkan sesuai dengan perancangan awal, sesuai dengan Class Diagram pada *Gambar 2*. Meskipun demikian masih terdapat penyesuaian pada diagram karena tidak ada implementasi untuk penyesuaian load file data yang lebih komperhensif ketika di tab AUDIT_PANEL terjadi perubahan data maka di tab DATA_PANEL tidak terjadi perubahan data simultan dan masalah yang dihadapi adalah saya hanya dapat menggunakan data loader bawaan dari library wx untuk melakukan data load untuk melakukan auditing data. Untuk kedepannya load data dapat lebih dioptimalkan dengan melakukan buffering data terlebih dahulu

khususnya di DATA_PANEL.h sehingga ketika data besar diload aplikasi tidak mengalami crash dan gagal untuk berjalan.

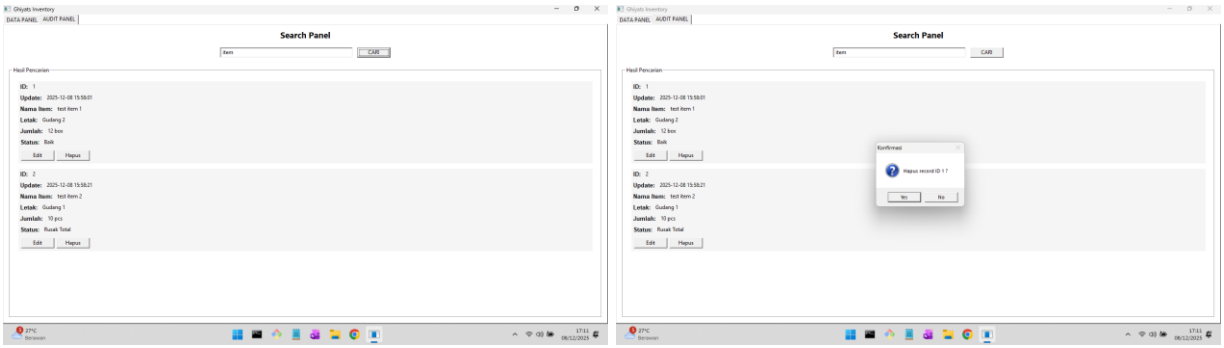
5.1. DATA PANEL



Gambar 11 Add Data (kiri) & Export To csv File (kanan)

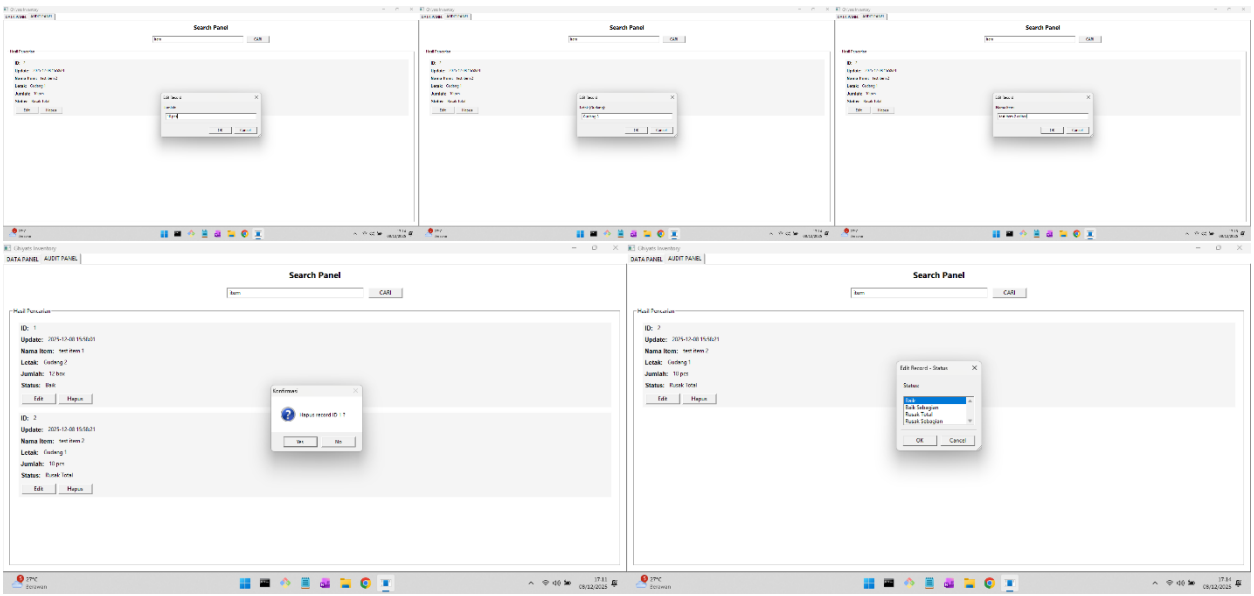
Pada bagian ini user akan disediakan tombol "Tambah" untuk memuat menambahkan data ke file *.csv*. Hal ini menyelesaikan masalah user untuk melakukan pencatatan data dengan lebih baik dan terstruktur dengan tamplate yang jelas (**P.1**). Selain itu terdapat tombol "Ekspor", ketika tombol ini ditriger maka akan muncul menu untk memilih file dan file akan tereksport seperti pada Gambar 11. Hal ini dapat menyelesaikan masalah user untuk melakukan penyimpanan data (**P.2**).

AUDIT PANEL



Gambar 12 Seacrh Item (Tracking Item) (kiri) & Delete Item (kanan)

Fitur ini dikembangkan untuk menyelesaikan **(P.3)** dimana user merasa kesulitan untuk melakukan tracking data yang telah disimpan. Fitur ini khusus dibuat untuk view data. Dan juga melakukan auditing data yang sudah dpiastikan ada. Pada Gambar 12, ditunjukan bahwa setelah terverifikasi bahwa data ada maka data dapat dihapus.



Gambar 13 Edit Item (sequence dari nama sampai kondisi item)

Sedangkan pada Gambar.13 data dapat diedit secara sequence namun id dari data tetap tidak dapat diedit. Hal ini disebabkan karena id adalah data identifier unik yang terbentuk ketika pengguna memasukan atau membuat data.