LAPORAN TUGAS BESAR CLO 2

CUI AUTO VENDING Using C# with Visual Studio 2022

Dibuat Oleh:

NAMA : GIOVAN DEO PRATAMA
NIM : 1201220450

SOFTWARE ENGINEERING

TELKOM UIVERSITY

2025

DAFTAR ISI

DESKRIPSI SINGKAT

Auto Vending adalah aplikasi smart vending machine berbasis self-checkout yang memungkinkan pengguna melakukan pemesanan dan pembayaran produk secara mandiri tanpa bantuan operator. Aplikasi ini dikembangkan menggunakan C# dalam Visual Studio 2022 dan dirancang untuk diintegrasikan langsung dengan perangkat vending fisik. Dengan antarmuka yang intuitif dan pengalaman transaksi yang cepat, Auto Vending bertujuan untuk meningkatkan efisiensi layanan penjualan otomatis, serta memberikan fleksibilitas tinggi dalam pengelolaan produk dan konfigurasi sistem.

Fitur-fitur yang tersedia:

- Perubahan Status
- Pemilihan Produk
- Ganti Bahasa
- Manajemen Produk
- Jam operasional
- Convert Mata Uang
- Checkout
- Transaksi

DAFTAR ANGGOTA KELOMPOK

1. Zidan Irfan Zaky 1201220003
2. Farhan Nugraha Sasongko Putra 1201220449
3. Radinka Putra Rahadian 1201220020
4. Giovan Deo Pratama 1201220450
5. Evi Fitriya 1201222005

Dengan Pembagian Tugas sesuai arahan dimana 1 teknik konstruksi dipegang oleh maximal 2 orang:

Nama Anggota / Nama Teknik konstruksi	Zidan Irfan Zaky	Farhan Nugraha Sasongko	Radnka Putra Rahadian	Giovan Deo Pratama	Evi Fitriya	Total Teknik Dipegang
Automata		✓			✓	2
Table Driven		✓	✓			2
Runtime	✓			✓		2
Code Reuse			✓		✓	2
Generics	✓			✓		2
Total Peserta Memegang	2	2	2	2	2	

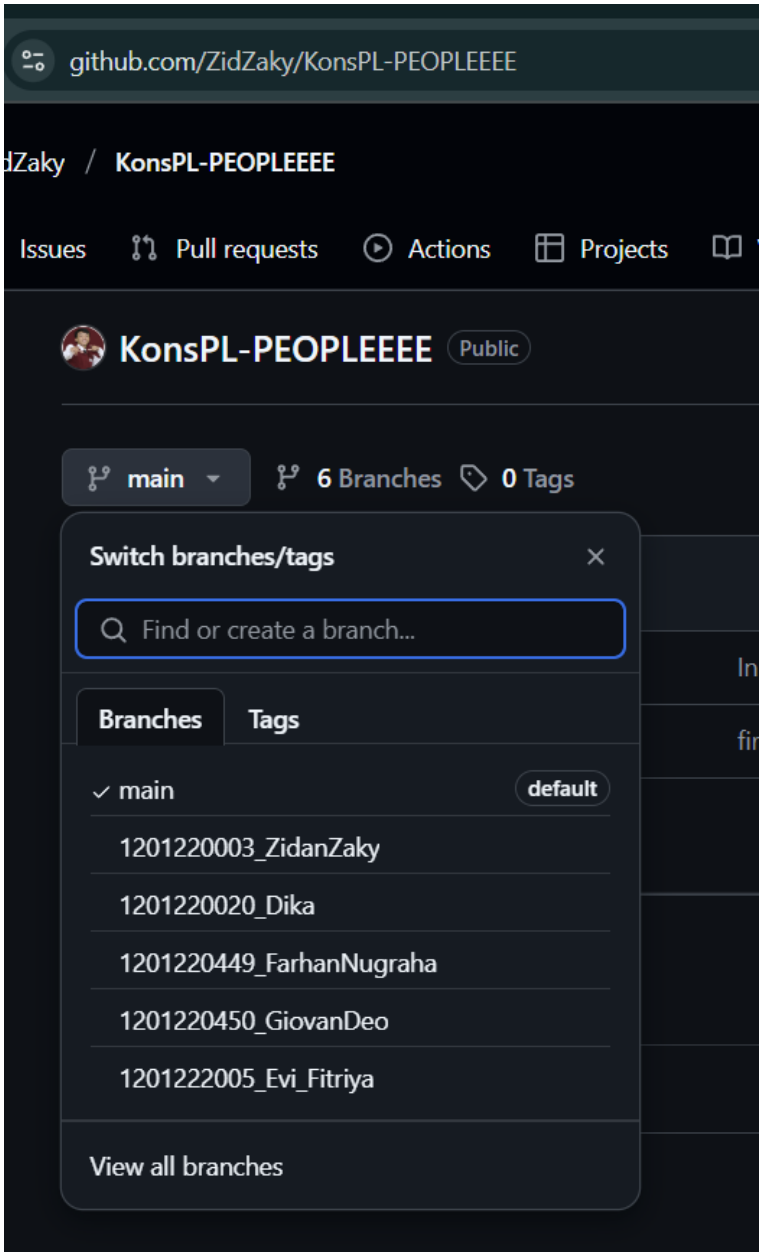
GITHUB

Repositori ini menggunakan sistem version control GitHub untuk mendokumentasikan proses konstruksi proyek secara kolaboratif. Terdapat *branch* pribadi yang digunakan untuk pengembangan fitur atau modul secara terpisah, sebelum digabungkan ke *branch* utama (*main*) melalui proses *commit* dan *merge*. Pendekatan ini memastikan riwayat perubahan tercatat dengan jelas serta mempermudah kolaborasi dan pengelolaan versi.

Berikut Link Github Kami:

https://github.com/ZidZaky/KonsPL-PEOPLEEEE.git

Dengan bukti:



IMPLEMENTASI DESIGN BY KONTRAK

```
Debug.Assert(currencyConfig != null, "currencyConfig should not be null");
Debug.Assert(currencyConfig.Currencies != null, "Currencies dictionary should not be null");
Debug.Assert(currencyConfig.Currencies.Count > 0, "Currencies dictionary should not be empty");

Debug.Assert(jamConfig != null, "jamConfig should not be null");
Debug.Assert(jamConfig.Modes != null, "Modes dictionary should not be null");
Debug.Assert(jamConfig.Modes.Count > 0, "Modes dictionary should not be empty");
```

1. Pada line code pertama memiliki garis besarnya yaitu melakukan pengecekan terlebih dahulu, apakah file currency_config.json berhasil dibaca atau tidak. Jika file tidak terbaca maka program akan berhenti disitu, karena jika tidak berhasil membaca file dan program masih lanjut berjalan maka akan menyebabkan error pada saat penggunaan data ke depannya.

2. Pada line code kedua memiliki sebuah garis besar yaitu memastikan semua daftar mata uang berada di dalam file tersebut bukan kosong ataupun sampai rusak. Apabila jika objek ini null, maka proses membaca file konfigurasi mata uang gagal dan program tidak akan bisa berjalan sebagaimana mestinya.

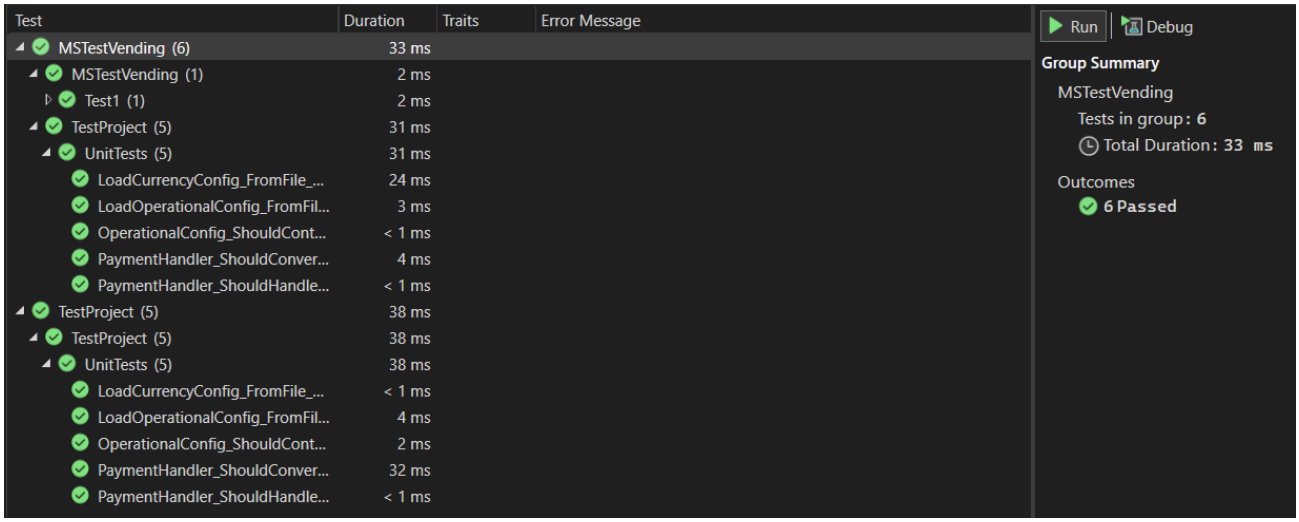
3. Pada line code ketiga bergaris besarkan daftarnya memang sudah tersedia, tetapi akan dilakukan pengecekan juga apakah terdapat isinya atau bahkan kosong, jika kosong maka program akan berhenti atau tidak lanjut. Karena jika isinya kosong, user tidak akan bisa memilih convert mata uang apapun untuk melakukan pembayaran.

4. Line code keempat memiliki garis besar yaitu mengecek pada bagian file operational_config.json bisa dibuka, jika tidak program akan dihentikan. Karena, jika tidak ada file tersebut maka tidak akan bisa menampilkan jam operasional vending machine itu sendiri.

5. Line code kelima bergaris besarkan memeriksa bahwa data jam operasional benar – benar memiliki daftar atau isinya. Jika pada saat akan menampilkan isi dari daftar tersebut kosong, maka bisa menyebabkan error atau tidak memunculkan apapun.

6. Yang terakhir pada line code keenam memiliki garis besar mengecek apakah daftar dari mode jam operasional berisikan sebuah data. Jika tidak berisikan sebuah data, sekalipun file berhasil dibaca program tidak akan memunculkan apa pun itu.

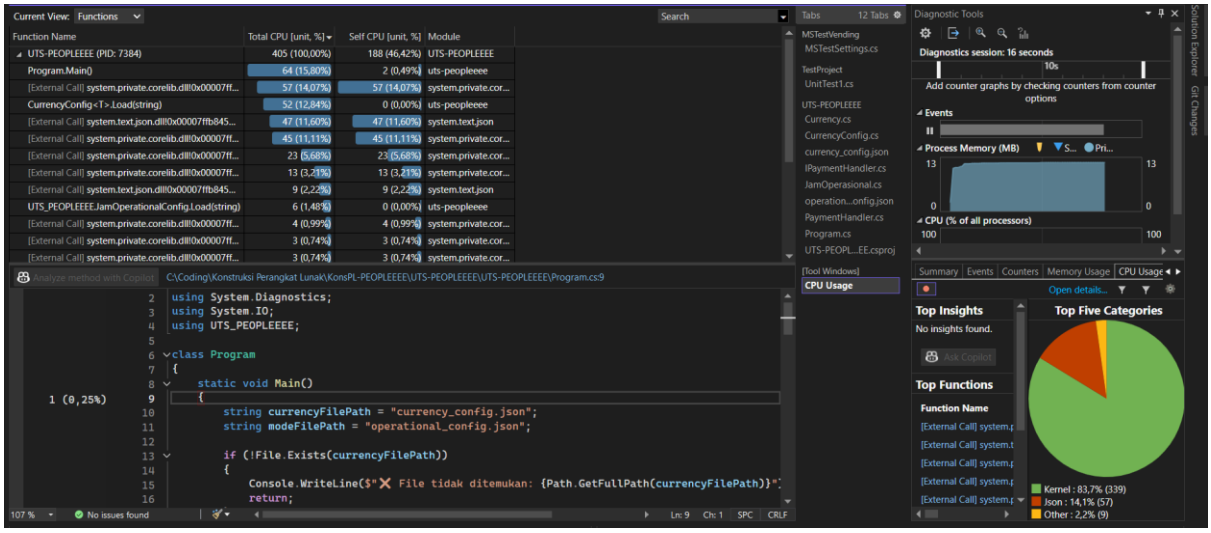
HASIL UNIT TESTING



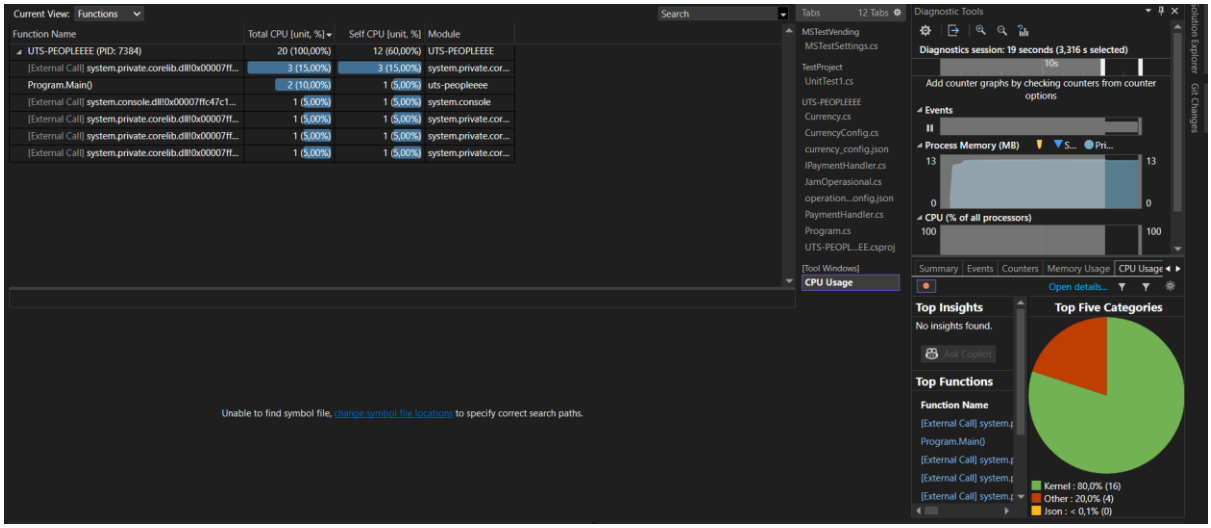
Setiap test method yang dijalankan berasal dari class bernama UnitTests, dan mencakup beberapa pengujian penting seperti membaca konfigurasi mata uang dari file, membaca konfigurasi jam operasional, memverifikasi mode default dalam konfigurasi jam operasional, serta dua pengujian untuk memverifikasi fungsi pembayaran (Pay()) baik untuk mata uang yang valid maupun yang tidak dikenali. Semua test ini berhasil dijalankan dengan baik, tanpa adanya error atau exception yang muncul. Ini menunjukkan bahwa logika aplikasi berjalan sesuai dengan harapan, baik dalam kondisi normal maupun saat dihadapkan pada input yang tidak valid.

Tidak hanya itu, kecepatan waktu eksekusi juga menunjukkan bahwa tidak ada proses yang terlalu berat atau tidak efisien dalam logika yang diuji. Dengan semua test lulus dan hasilnya sesuai yang diharapkan, ini menandakan bahwa kualitas kode dan struktur pengujian dalam proyek Anda sudah cukup baik.

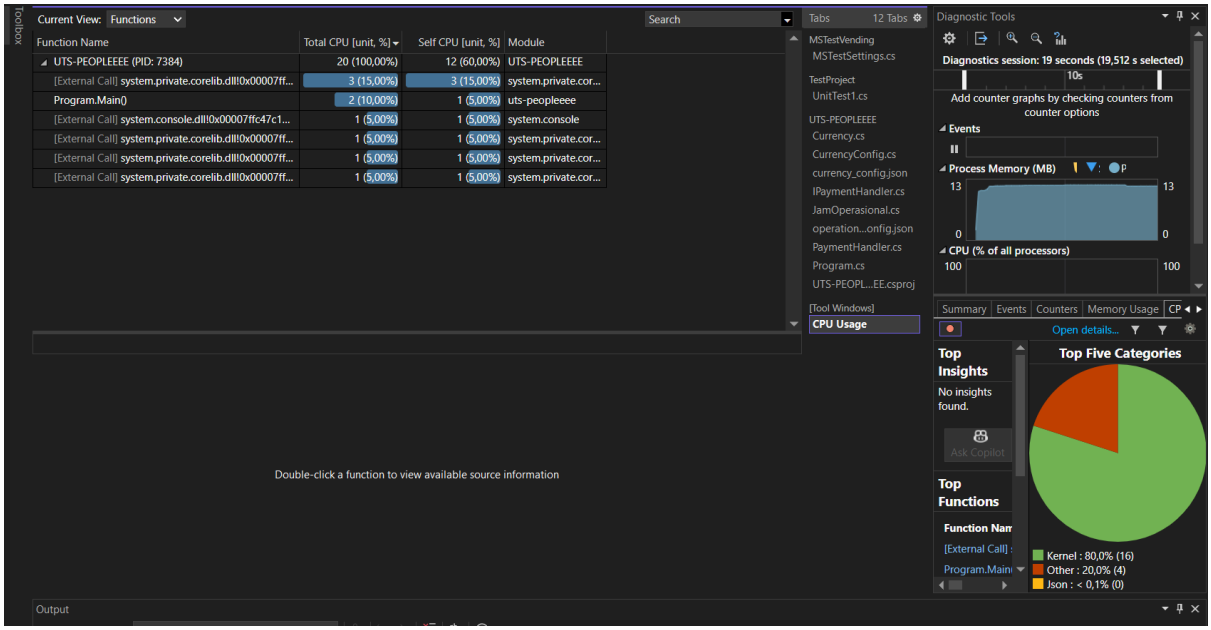
HASIL PERFORMANCE TESTING



Fungsi System.Private.CoreLib!System.Globalization.CalendarData.CreateInvariant() mendominasi dengan penggunaan CPU sebesar 67,41%, diikuti oleh System.Private.CoreLib!System.Globalization.CalendarData.LoadCalendarDataFromSystem() sebesar 22,79%. Ini menunjukkan bahwa sebagian besar waktu eksekusi program dihabiskan dalam proses internal .NET untuk menangani data kalender, bukan di logika utama program.



70,01% pemakaian CPU berasal dari fungsi eksternal atau sistem (External Code), sementara 29,99% digunakan oleh fungsi dalam program Anda (Program.Main). Ini berarti sebagian besar proses masih ditangani oleh pustaka internal .NET, dan hanya sebagian kecil yang berasal dari logika program utama.



Memperlihatkan bahwa 69,81% aktivitas CPU berasal dari kode eksternal (seperti pustaka .NET), sementara 29,61% digunakan oleh fungsi Program.Main milik aplikasi. Ini berarti sebagian besar pemrosesan terjadi di luar kode utama, dan hanya sebagian kecil berasal dari logika program pengguna.