**LAPORAN TUGAS BESAR CLO 2**

**CUI AUTO VENDING Using C# with Visual Studio 2022**



Dibuat Oleh:

Zidan Irfan Zaky 1201220003

**SOFTWARE ENGINEERING**

**TELKOM UNIVERSITY SURABAYA**

**2025**

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc198241124)

[DESKRIPSI SINGKAT 3](#_Toc198241125)

[DAFTAR ANGGOTA KELOMPOK 3](#_Toc198241126)

[GITHUB 4](#_Toc198241127)

[IMPLEMENTASI DESIGN BY KONTRAK 5](#_Toc198241128)

[Defensive Programming 7](#_Toc198241129)

[HASIL UNIT TESTING 8](#_Toc198241130)

[MsTest : 8](#_Toc198241131)

[xUnit 9](#_Toc198241132)

[HASIL PERFOMANCE TESTING 11](#_Toc198241133)

# 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:



# 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>

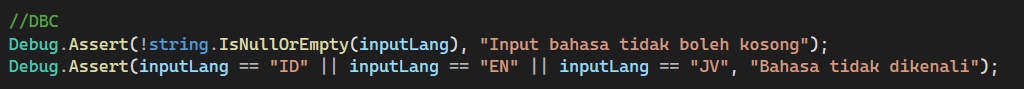
Dengan bukti:

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Konten yang dihasilkan AI mungkin salah.

# IMPLEMENTASI DESIGN BY KONTRAK

Salah Satu Bentuk Implementasi Design By Kontrak Ialah dengan Mengadakan Debug.Assert untuk melakukan Pengecekan Terhadap yang kita minta dengan Minimal 2 Parameter, Yang Parameter pertama diisi dengan Kondisi Yang harus dipenuhi sebelum Assert, kemudian parameter ke 2 pesan error yang ingin diberikan



Tetapi Debug ini Hanya akan Tampil Ketika Mode Debug Saja Dengan Tampilan Error Sebagaimana Berikut :

---- DEBUG ASSERTION FAILED ----

---- Assert Short Message ----

Bahasa tidak dikenali

---- Assert Long Message ----

Bukti Contoh Hasil Screenshot :

A screenshot of a computer

AI-generated content may be incorrect.

Dan Seluruh Konsep Design By Contract Aku Implementasikan Pada Class GenericRepositry yang Memiliki Cukup banyak Fungsi Seperti :

* Fungsi Tambah untuk Tambah Produk, Diberikan Debug Assert untuk Mengecek Apakah Data yang diisikan Memang ada atau null
* Fungsi LihatSemua, Diberikan Assert untuk Melakukan Pengecekan yang harusnya Items list yang ada di class ini itu tidak Null
* Fungsi Detil, Diberikan Asasert untuk Id yang diberikan pada parameter fungsi ini itu sama dengan salah satu ID Dari list item Yang ada
* Fungsi Hapus, Diberikan assert untuk Melakukan Pengecekan Pertama ID nya harus sama dengan salah satu item, kemudian dicek pada akhirnya jumlah item harusnya berkurang 1
* Fungsi Clear, diberikan assert untuk melakukan check Dimana seharusnya setelah clear seluruh item yang ad aitu sudah tidak ada dengan Panjang array listnya itu 0
* Fungsi hitungTotal diberikan assert untuk menegecek parameter fungsi kalkulasi ini yang terdiri dari generic dan Decimal ini tidak null

Bukti Implementasi Dibawah :

A screen shot of a computer program

AI-generated content may be incorrect.

## Defensive Programming

Saya juga Mengimplementasikan Untuk Defensive Programming Yakni dengan mengadakannya Kondisi Else di berbagai macam kondisi yang memungkinkan untuk memberikan Error atau pengembalian yang tidak di Handle Seperti :

A screen shot of a computer program

AI-generated content may be incorrect.

Ini Contoh Defensive Programming pada Input pemilihan Bahasa, apakah Bahasa Itu terdaftar atau tidak Dilakukan cek pada If diatas, sehingga Mengimplementasikan Defensive Programming

# HASIL UNIT TESTING

## MsTest :

***GenericRepositoryTests***

* A screenshot of a computer program

  AI-generated content may be incorrect.GenericRepo\_Tambah()
* GenericRepo\_Detil()
* GenericRepo\_Hapus()
* GenericRepo\_Clear()
* GenericRepo\_HitungTotal()

***VendingManagerTests***

* VendingManager\_TambahProduk()
* VendingManager\_DetilProduk()
* VendingManager\_DeleteProduk()
* VendingManager\_CheckoutProduk()
* VendingManager\_HitungTotal()

***LanguageConfigTests***

* LanguageConfig\_GetMessage()
* RuntimeLanguage\_Load()

***ProductTransactionTests***

* Product\_AddProduct()
* Transaction\_SimpanData()

14 Passed, 0 Warning, 0 Failed

Pengujian unit menggunakan MsTest dilakukan untuk memastikan setiap komponen dari sistem vending machine bekerja dengan baik dan sesuai harapan. Pada bagian GenericRepositoryTests, dilakukan validasi terhadap fungsionalitas dasar repository generik seperti penambahan data melalui GenericRepo\_Tambah, pengambilan detail dengan GenericRepo\_Detil, penghapusan item lewat GenericRepo\_Hapus, pengosongan koleksi lewat GenericRepo\_Clear, serta perhitungan total harga dalam GenericRepo\_HitungTotal.

Selanjutnya, VendingManagerTests menguji logika manajer vending machine. Fungsionalitas yang diuji meliputi penambahan produk (VendingManager\_TambahProduk), melihat detail produk (VendingManager\_DetilProduk), menghapus produk (VendingManager\_DeleteProduk), melakukan checkout terhadap seluruh produk (VendingManager\_CheckoutProduk), dan menghitung total biaya transaksi (VendingManager\_HitungTotal).

Pengujian konfigurasi bahasa dilakukan pada LanguageConfigTests, di mana LanguageConfig\_GetMessage memastikan bahwa sistem bisa mengambil pesan berdasarkan kode bahasa yang dipilih, dan RuntimeLanguage\_Load menguji proses pembacaan konfigurasi bahasa dari file JSON eksternal.

Terakhir, bagian ProductTransactionTests memastikan bahwa objek Product dan Transaction dapat menyimpan serta merepresentasikan data dengan benar. Fungsi Product\_AddProduct mengecek inisialisasi properti produk, sementara Transaction\_SimpanData memverifikasi bahwa transaksi berhasil mencatat ID produk, nama, kuantitas, total harga, dan waktu pembelian dengan tepat.

## xUnit

**GenericRepositoryTests**

* + GenericRepo\_Tambah()
  + GenericRepo\_Detil()
  + A screenshot of a computer

    AI-generated content may be incorrect.GenericRepo\_Hapus()
  + GenericRepo\_Clear()
  + GenericRepo\_HitungTotal()

**VendingManagerTests**

* + VendingManager\_TambahProduk()
  + VendingManager\_DetilProduk()
  + VendingManager\_DeleteProduk()
  + VendingManager\_CheckoutProduk()
  + VendingManager\_HitungTotal()

**LanguageConfigTests**

* + LanguageConfig\_GetMessage()
  + RuntimeLanguage\_Load()

**ProductTransactionTests**

* + Product\_AddProduct()
  + Transaction\_SimpanData()

14 Passed, 0 Warning, 0 Failed

Pengujian unit menggunakan **xUnit** dilakukan untuk memastikan bahwa setiap komponen dari sistem vending machine bekerja dengan baik dan sesuai dengan fungsionalitas yang diharapkan.

Pada bagian **GenericRepositoryTests**, dilakukan validasi terhadap operasi dasar dari repository generik. GenericRepo\_Tambah menguji proses penambahan item ke repository, GenericRepo\_Detil memastikan pengambilan detail item berdasarkan ID berhasil, GenericRepo\_Hapus memverifikasi bahwa item bisa dihapus, GenericRepo\_Clear memastikan seluruh data bisa dikosongkan, dan GenericRepo\_HitungTotal menguji fungsi agregasi untuk menghitung total harga produk.

Bagian **VendingManagerTests** fokus pada logika bisnis utama vending machine. Fungsi VendingManager\_TambahProduk memastikan produk dapat ditambahkan, VendingManager\_DetilProduk mengambil detail produk dari sistem, VendingManager\_DeleteProduk menghapus produk, VendingManager\_CheckoutProduk melakukan pengosongan semua produk (checkout), dan VendingManager\_HitungTotal menghitung total harga seluruh produk yang tersimpan.

Untuk mendukung fitur multi-bahasa, pengujian pada **LanguageConfigTests** dilakukan. Fungsi LanguageConfig\_GetMessage menguji apakah pesan bisa diambil sesuai kode bahasa, dan RuntimeLanguage\_Load memverifikasi pemuatan file konfigurasi JSON eksternal agar sistem dapat dinamis dalam menampilkan pesan berdasarkan pilihan bahasa.

Terakhir, pada bagian **ProductTransactionTests**, dilakukan pengujian terhadap struktur data entitas utama. Fungsi Product\_AddProduct mengecek apakah objek Product dapat dibentuk dengan benar, dan Transaction\_SimpanData memastikan objek Transaction menyimpan data transaksi seperti ID produk, nama, jumlah, total harga, dan waktu pembelian secara akurat.

# HASIL PERFOMANCE TESTING

A screenshot of a computer

AI-generated content may be incorrect.

Pada Hasil Diatas Menampilkan CPU Usage dengan partisi 98.5% Kernel dan Lain2 Senbesar 1.5%, dikarenakan pada Fungsi diatas mula-mula melakukan Loading Core Library Aplikasi dan Menjalankan Fungsi Program.Main sehingga CPU Usage paling banyak Oleh Kernel

A screenshot of a computer program

AI-generated content may be incorrect.

Pada Performance test Di Line 30, CPU Usage menampilkan 27.7% Json, 69.6% Kernel dan Other 2.7%. Dikarenakan Pada Line 30 Memanggil Fungsi Load Yang Menjalankan Fungsi Untuk Melakukan Deserialisasi Pada Json menjadi Data yang dapat diolah diaplikasi sehingga adanya lonjakan di CPU Untuk JSON.

A screenshot of a computer program

AI-generated content may be incorrect.

Pada Performance Testing Untuk Fungsi Menampilkan Detil Produk, Menamplikan CPU Usage Oleh kernel Yang cukup besar dikarenakan Mengambil Get Data pada Aplikasi Auto Vending yang Merupakan BackEnd, sehingga Kebanyakan CPU pada Saat ini Oleh Kernel

A screenshot of a computer program

AI-generated content may be incorrect.

Pada Performance testing ini, menampilkan hamper 100% CPU digunakan Oleh kernel karena pada line code 137 – 144 Sedang melakukan penyimpanan data Untuk Disimpan pada Object Transaksi.

A screenshot of a computer

AI-generated content may be incorrect.

Setelah itu 1 Flow Aplikasi Telah di jalankan Dan Hasil Performance Test Pun Sebagaimana yang diatas, dengan akhirnya aplikasi secara otomatis menjalankan ClearScreen sehingga Tampilan Akhirnya Langsung halaman Kosong Kembali ke Main Menu, Sehingga CPU hamper 100% DIgunakan Oleh kernel