

LAPORAN PRESENTASI

Tugas Besar Algoritma & Pemrograman
Aplikasi Sembako Rumahan

ANGGOTA KELOMPOK:



Raden Aurel Aditya K

103112430267



M. Fachri Auravyano A. S.

103112430180

LATAR BELAKANG

Dalam kehidupan sehari-hari, pengelolaan stok bahan makanan seringkali menjadi tantangan, terutama untuk menghindari pemborosan akibat kedaluwarsa atau kehabisan stok. Pencatatan manual seringkali tidak efisien dan rentan kesalahan. Tugas ini bertujuan untuk menerapkan konsep-konsep dasar algoritma dan pemrograman dalam solusi praktis.

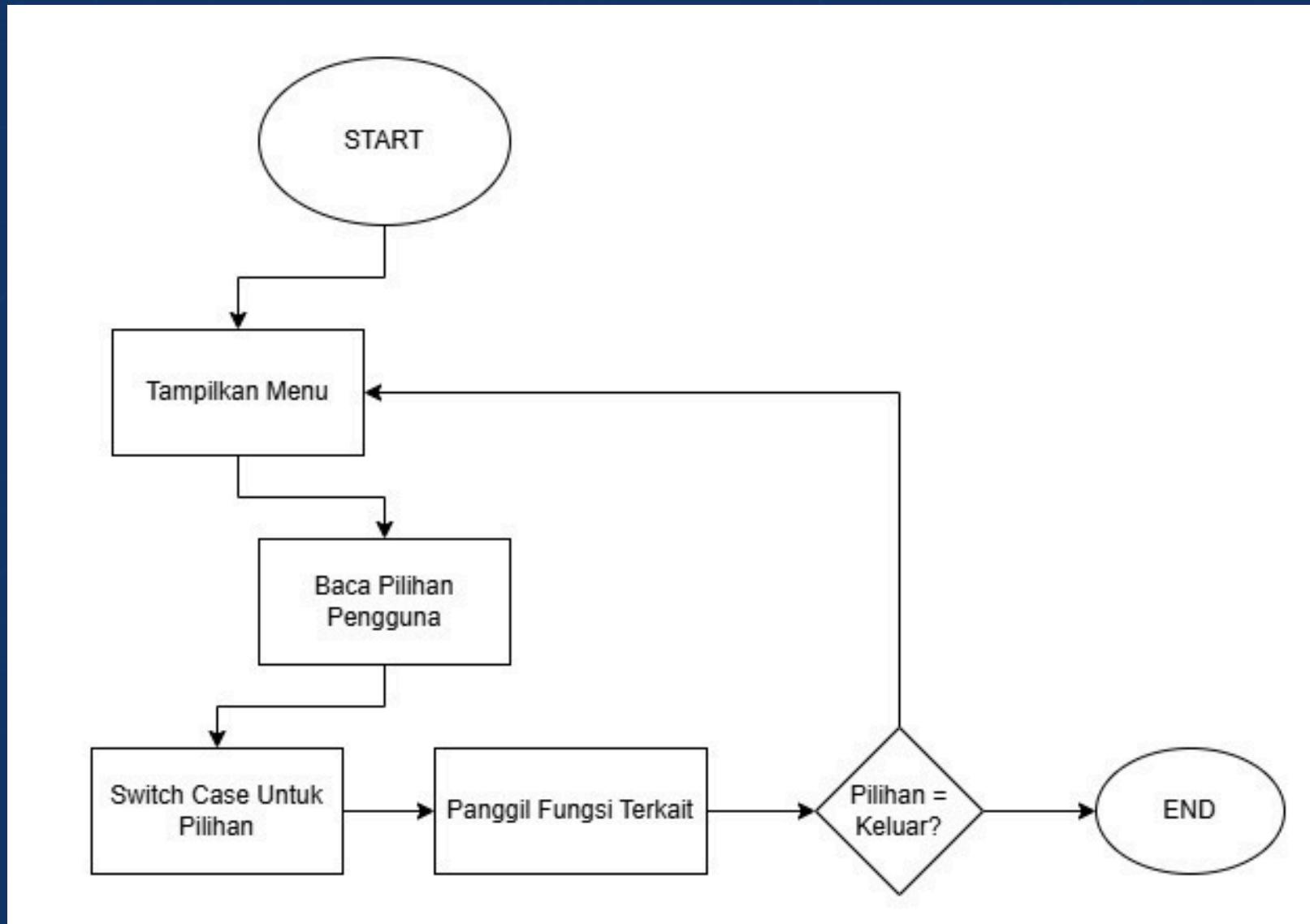
TUJUAN PROYEK

1. Membangun aplikasi sederhana untuk membantu pengguna mengelola stok bahan makanan secara digital.
2. Memfasilitasi pencatatan, pembaruan, penghapusan, pencarian, dan pelaporan stok bahan makanan.
3. Menerapkan algoritma pencarian (Sequential Search, Binary Search) dan pengurutan (Selection Sort, Insertion Sort) dalam skenario nyata.

RUANG LINGKUP PROYEK

1. Aplikasi ini berjalan di lingkungan command-line interface (CLI) atau terminal.
2. Data disimpan sementara dalam memori (tidak persisten).
3. Fungsionalitas utama mencakup penambahan, pengubahan, penghapusan, pencarian, tampilan data, pengurutan, penandaan penggunaan, peringatan kedaluwarsa, dan laporan stok.

FLOWCHART PROGRAM UTAMA



STRUKTUR DATA YANG DIGUNAKAN

```
package main

import (
    "fmt"
    "strings"
)

const MAKS = 100

type BahanMakanan struct {
    Nama      string
    Jumlah    int
    Tanggal   string
    Digunakan bool
}

var stok [MAKS]BahanMakanan
var jumlahData int
```

Package main merupakan Program Utama, Import fmt & string untuk menggunakan fungsi. struct BahanMakanan untuk merepresentasikan setiap item stok, yang memiliki atribut Nama, Jumlah, Tanggal kedaluwarsa, dan status Digunakan. Semua data bahan makanan disimpan dalam sebuah array stok dengan ukuran maksimum 100, dan variabel jumlahData melacak berapa banyak elemen yang saat ini terisi di array tersebut.

PENJELASAN SOURCE CODE

```
func bacaString(pesan string) string {  
    var input string  
    fmt.Print(pesan)  
    fmt.Scanln(&input)  
    return input  
}  
  
func bacaInteger(pesan string) int {  
    var input int  
    fmt.Print(pesan)  
    fmt.Scanln(&input)  
    return input  
}
```

Input Pengguna (Fungsi bacaString dan bacaInteger)

Fungsi bacaString dan bacaInteger adalah fungsi helper untuk membaca input dari pengguna dengan menampilkan pesan tertentu. Ini membuat kode utama lebih bersih dan mudah dibaca.

PENJELASAN SOURCE CODE

```
func tambahBahan() {
    if jumlahData >= MAKS {
        fmt.Println("🚫 Stok penuh!")
        return
    }
    fmt.Println("\n➕ Tambah Bahan Baru")
    fmt.Println("-----")
    stok[jumlahData].Nama = bacaString("➡ Nama bahan: ")
    stok[jumlahData].Jumlah = bacaInteger("📦 Jumlah: ")
    stok[jumlahData].Tanggal = bacaString("📅 Tanggal kedaluwarsa (YYYY-MM-DD): ")
    stok[jumlahData].Digunakan = false
    jumlahData++
    fmt.Println("✅ Bahan berhasil ditambahkan.")
}
```

Operasi Dasar (Tambah)

Fungsi tambahBahan bertanggung jawab untuk menambahkan item baru ke dalam stok. Sebelum menambahkan, ia memeriksa apakah array sudah penuh. Kemudian, ia meminta input dari pengguna untuk nama, jumlah, dan tanggal kedaluwarsa, lalu menyimpannya ke dalam array stok pada indeks jumlahData dan menginkrementasi jumlahData.

PENJELASAN SOURCE CODE

```
func ubahBahan() {
    fmt.Println("\n📝 Ubah Data Bahan")
    fmt.Println("-----")
    nama := bacaString("🔍 Masukkan nama bahan yang ingin diubah: ")
    idx := sequentialSearch(nama)
    if idx == -1 {
        fmt.Println("✖ Bahan tidak ditemukan.")
        return
    }
    stok[idx].Nama = bacaString("➡ Nama baru: ")
    stok[idx].Jumlah = bacaInteger("📦 Jumlah baru: ")
    stok[idx].Tanggal = bacaString("📅 Tanggal kedaluwarsa baru (YYYY-MM-DD): ")
    fmt.Println("✓ Data berhasil diubah.")
}
```

Operasi Dasar (Ubah)

Fungsi `ubahBahan` berfungsi untuk memodifikasi data bahan makanan yang sudah ada. Proses ini diawali dengan meminta pengguna untuk memasukkan nama bahan yang ingin diubah. Selanjutnya, program akan melakukan pencarian bahan tersebut menggunakan `sequentialSearch`. Apabila bahan ditemukan, program akan meminta input baru untuk nama, jumlah, dan tanggal kedaluwarsa, kemudian memperbarui data pada indeks yang bersangkutan di dalam array `stok`

PENJELASAN SOURCE CODE

```
func hapusBahan() {
    fmt.Println("\n[Hapus Bahan]")
    fmt.Println("-----")
    nama := bacaString("🔍 Masukkan nama bahan yang ingin dihapus: ")
    idx := sequentialSearch(nama)
    if idx == -1 {
        fmt.Println("❌ Bahan tidak ditemukan.")
        return
    }
    for i := idx; i < jumlahData-1; i++ {
        stok[i] = stok[i+1]
    }
    jumlahData--
    fmt.Println("✓ Data berhasil dihapus.")
}
```

Operasi Dasar (Hapus)

Fungsi `hapusBahan` bertanggung jawab untuk menghapus entri bahan makanan dari stok. Pengguna akan diminta untuk memasukkan nama bahan yang ingin dihapus. Setelah bahan berhasil ditemukan melalui `sequentialSearch`, program akan menggeser semua elemen yang berada setelah indeks bahan yang dihapus, mengisi kekosongan tersebut. Terakhir, nilai `jumlahData` akan didekrementasi untuk mencerminkan berkurangnya jumlah item dalam stok

PENJELASAN SOURCE CODE

```
80 func sequentialSearch(nama string) int {
81     nama = strings.ToLower(nama)
82     for i := 0; i < jumlahData; i++ {
83         if strings.ToLower(stok[i].Nama) == nama {
84             return i
85         }
86     }
87     return -1
88 }

89 func binarySearch(nama string) int {
90     urutkanNama()
91     kiri, kanan := 0, jumlahData-1
92     nama = strings.ToLower(nama)
93     for kiri <= kanan {
94         tengah := (kiri + kanan) / 2
95         namaTengah := strings.ToLower(stok[tengah].Nama)
96         if nama == namaTengah {
97             return tengah
98         } else if nama < namaTengah {
99             kanan = tengah - 1
100        } else {
101            kiri = tengah + 1
102        }
103    }
104    return -1
105 }
106 }
```

Algoritma Pencarian (Sequential Search & Binary Search)

Implementasi dua metode pencarian: Sequential Search dan Binary Search. Sequential Search memeriksa setiap elemen satu per satu hingga ditemukan atau seluruh array habis. Sedangkan Binary Search memerlukan data yang sudah terurut. Algoritma ini bekerja dengan membagi array menjadi dua di setiap langkah, secara signifikan mengurangi jumlah perbandingan yang diperlukan. Untuk implementasi Binary Search, kami memastikan data nama sudah terurut terlebih dahulu menggunakan fungsi `urutkanNama()`. Kedua fungsi ini mengembalikan indeks data jika ditemukan, atau -1 jika tidak ditemukan.

PENJELASAN SOURCE CODE

```
func urutkanNama() {  
    for i := 0; i < jumlahData-1; i++ {  
        for j := 0; j < jumlahData-i-1; j++ {  
            if strings.ToLower(stok[j].Nama) > strings.ToLower(stok[j+1].Nama) {  
                stok[j], stok[j+1] = stok[j+1], stok[j]  
            }  
        }  
    }  
}
```

Fungsi urutkanNama()

Fungsi urutkanNama mengimplementasikan algoritma Bubble Sort untuk mengurutkan data bahan makanan berdasarkan namanya secara alfabetis dari A ke Z. Algoritma ini bekerja dengan membandingkan pasangan elemen yang berdekatan dan menukar jika urutannya salah. Proses ini diulang hingga tidak ada lagi penukaran yang diperlukan, menandakan array sudah terurut. Fungsi ini penting untuk memastikan binarySearch dapat bekerja dengan benar, karena Binary Search memerlukan data yang sudah terurut.

PENJELASAN SOURCE CODE

```
func cariBahan() {
    fmt.Println("\n🔍 Pencarian Bahan")
    fmt.Println("-----")
    fmt.Println("[1] Sequential Search")
    fmt.Println("[2] Binary Search")
    pilih := bacaInteger("👉 Pilihan: ")

    nama := bacaString("🔍 Nama bahan yang dicari: ")
    var idx int
    if pilih == 1 {
        idx = sequentialSearch(nama)
    } else if pilih == 2 {
        idx = binarySearch(nama)
    } else {
        fmt.Println("❌ Metode tidak valid.")
        return
    }

    if idx == -1 {
        fmt.Println("❌ Bahan tidak ditemukan.")
    } else {
        fmt.Printf("✅ Ditemukan: %s (%d) - Kedaluwarsa: %s\n", stok[idx].Nama, stok[idx].Jumlah, stok[idx].Tanggal)
    }
}
```

Fungsi cariBahan()

Fungsi cariBahan menyediakan antarmuka bagi pengguna untuk melakukan pencarian data bahan makanan. Pengguna diberikan pilihan antara menggunakan Sequential Search atau Binary Search. Setelah pengguna memilih metode dan memasukkan nama bahan yang dicari, fungsi ini akan memanggil salah satu dari dua fungsi pencarian (sequentialSearch atau binarySearch). Hasil pencarian, baik itu ditemukan atau tidak, akan ditampilkan kepada pengguna beserta detail bahan jika ditemukan.

PENJELASAN SOURCE CODE

```
143 func peringatanKedaluwarsa() {
144     fmt.Println("⚠️ Bahan mendekati tanggal kedaluwarsa:")
145     ada := false
146     for i := 0; i < jumlahData; i++ {
147         if stok[i].Tanggal != "" && strings.HasPrefix(stok[i].Tanggal, "2025-06") {
148             fmt.Printf("⌚ %s (%d) - Kedaluwarsa: %s\n", stok[i].Nama, stok[i].Jumlah, stok[i].Tanggal)
149             ada = true
150         }
151     }
152     if !ada {
153         fmt.Println("✅ Tidak ada bahan yang mendekati kedaluwarsa.")
154     }
155 }
```

Fungsi peringatanKedaluwarsa()

Fungsi peringatanKedaluwarsa berfungsi untuk menampilkan daftar bahan makanan yang tanggal kedaluwarsanya mendekati bulan Juni 2025. Program akan melakukan iterasi melalui setiap item stok dan memeriksa apakah string tanggal kedaluwarsanya dimulai dengan '2025-06'. Jika ditemukan, informasi bahan tersebut akan ditampilkan. Apabila tidak ada bahan yang memenuhi kriteria, pesan 'Tidak ada bahan yang mendekati kedaluwarsa' akan ditampilkan." (Catatan: Kriteria 2025-06 dalam kode ini menunjukkan bahwa fitur ini mungkin dikonfigurasi untuk bulan tertentu, dan perlu diperhatikan jika tanggal saat ini sudah melewati Juni 2025).

PENJELASAN SOURCE CODE

```
157 func urutkanJumlah() {
158     for i := 0; i < jumlahData-1; i++ {
159         idxMin := i
160         for j := i + 1; j < jumlahData; j++ {
161             if stok[j].Jumlah < stok[idxMin].Jumlah {
162                 idxMin = j
163             }
164         }
165         stok[i], stok[idxMin] = stok[idxMin], stok[i]
166     }
167     fmt.Println("✓ Diurutkan berdasarkan jumlah ( 12 Selection Sort).")
168 }
```

Fungsi urutkanJumlah()

Fungsi urutkanJumlah mengimplementasikan algoritma Selection Sort untuk mengurutkan bahan makanan berdasarkan nilai jumlahnya dari yang terkecil hingga terbesar. Algoritma ini bekerja dengan berulang kali mencari elemen minimum dari bagian array yang belum terurut dan menukarannya dengan elemen pertama dari bagian yang belum terurut.

Proses ini memastikan bahwa setiap iterasi menempatkan elemen yang benar pada posisi yang tepat.

PENJELASAN SOURCE CODE

```
170 func urutkanTanggal() {  
171     for i := 1; i < jumlahData; i++ {  
172         key := stok[i]  
173         j := i - 1  
174         for j >= 0 && stok[j].Tanggal > key.Tanggal {  
175             stok[j+1] = stok[j]  
176             j--  
177         }  
178         stok[j+1] = key  
179     }  
180     fmt.Println("✓ Diurutkan berdasarkan tanggal kedaluwarsa (💻 Insertion Sort).")  
181 }
```

Fungsi urutkanTanggal()

Fungsi `urutkanTanggal` mengimplementasikan algoritma Insertion Sort untuk mengurutkan bahan makanan berdasarkan tanggal kedaluwarsanya dari yang terlama hingga terbaru. Algoritma ini bekerja dengan membagi array menjadi dua bagian: bagian yang sudah terurut dan bagian yang belum terurut. Setiap elemen dari bagian yang belum terurut diambil dan disisipkan ke posisi yang tepat dalam bagian yang sudah terurut, menjaga urutan data.

PENJELASAN SOURCE CODE

```
func laporanStok() {
    total := 0
    digunakan := 0
    for i := 0; i < jumlahData; i++ {
        total += stok[i].Jumlah
        if stok[i].Digunakan {
            digunakan += stok[i].Jumlah
        }
    }
    fmt.Println("\n📊 Laporan Stok Bahan")
    fmt.Println("-----")
    fmt.Printf("📦 Total bahan tersedia: %d\n", total)
    fmt.Printf("✓ Total bahan digunakan: %d\n", digunakan)
    fmt.Printf("📁 Jumlah entri: %d\n", jumlahData)
}
```

Fungsi laporanStok()

Fungsi laporanStok berfungsi untuk menyajikan ringkasan statistik mengenai stok bahan makanan. Fungsi ini menghitung total keseluruhan jumlah bahan yang tersedia dan total jumlah bahan yang telah ditandai sebagai 'digunakan' dengan mengiterasi seluruh array stok. Hasil perhitungan tersebut, beserta jumlah entri data yang ada, kemudian ditampilkan kepada pengguna.

PENJELASAN SOURCE CODE

```
func tampilkanData() {
    fmt.Println("\nDaftar Bahan Makanan")
    fmt.Println("-----")
    for i := 0; i < jumlahData; i++ {
        status := "X Belum Digunakan"
        if stok[i].Digunakan {
            status = "✓ Sudah Digunakan"
        }
        fmt.Printf("%d. %s (%d) - Kedaluwarsa: %s [%s]\n", i+1, stok[i].Nama, stok[i].Jumlah, stok[i].Tanggal, status)
    }
}
```

Fungsi tampilanData()

Fungsi tampilanData bertanggung jawab untuk menampilkan seluruh daftar bahan makanan yang tersimpan dalam sistem. Setiap item bahan makanan akan ditampilkan bersama dengan nomor urut, nama, jumlah, tanggal kedaluwarsa, dan status penggunaannya (apakah 'Sudah Digunakan' atau 'Belum Digunakan'). Fungsi ini memastikan pengguna dapat melihat overview lengkap dari stok mereka.

PENJELASAN SOURCE CODE

```
func tandaiDigunakan() {
    nama := bacaString("⚡ Nama bahan yang ingin ditandai digunakan: ")
    idx := sequentialSearch(nama)
    if idx == -1 {
        fmt.Println("✖ Bahan tidak ditemukan.")
        return
    }
    if stok[idx].Jumlah == 0 {
        fmt.Println("⚠ Tidak ada stok tersisa untuk bahan ini.")
        return
    }

    jumlah := bacaInteger("🔢 Jumlah yang ingin ditandai digunakan: ")
    if jumlah <= 0 || jumlah > stok[idx].Jumlah {
        fmt.Println("✖ Jumlah tidak valid atau melebihi stok.")
        return
    }

    stok[idx].Jumlah -= jumlah

    // Tandai sebagai digunakan jika seluruh stok telah digunakan
    if stok[idx].Jumlah == 0 {
        stok[idx].Digunakan = true
    }

    fmt.Printf("✓ %d dari %s ditandai sebagai digunakan.\n", jumlah, stok[idx].Nama)
}
```

Fungsi tandaiDigunakan()

Fungsi `tandaiDigunakan` memungkinkan pengguna untuk menandai sebagian atau seluruh jumlah bahan makanan sebagai telah digunakan. Pengguna diminta untuk memasukkan nama bahan dan jumlah yang ingin ditandai. Program akan mencari bahan tersebut, memvalidasi input jumlah, mengurangi jumlah stok yang tersedia, dan jika stok bahan tersebut habis (`stok[idx].Jumlah == 0`), status `Digunakan` akan diatur menjadi `true`. Ini membantu melacak penggunaan bahan

PENJELASAN SOURCE CODE

```
func menu() {
    for {
        fmt.Println("\n-----")
        fmt.Println("● MANAJEMEN STOK BAHAN MAKANAN ●")
        fmt.Println("-----")
        fmt.Println("1 [+] Tambah Bahan")
        fmt.Println("2 [pen] Ubah Bahan")
        fmt.Println("3 [trash] Hapus Bahan")
        fmt.Println("4 [magnifying glass] Cari Bahan")
        fmt.Println("5 [list] Tampilkan Semua Data")
        fmt.Println("6 [asc/desc] Urutkan Jumlah")
        fmt.Println("7 [calendar] Urutkan Tanggal Kedaluwarsa")
        fmt.Println("8 [lightning bolt] Tandai Digunakan")
        fmt.Println("9 [warning sign] Peringatan Kedaluwarsa")
        fmt.Println("10 [bar chart] Laporan Stok")
        fmt.Println("0 [X] Keluar")
        fmt.Println("-----")
```

```
pilih := bacaInteger("★ Pilih menu: ")
switch pilih {
case 1:
    tambahBahan()
case 2:
    ubahBahan()
case 3:
    hapusBahan()
case 4:
    cariBahan()
case 5:
    tampilanData()
case 6:
    urutkanJumlah()
    tampilanData()
case 7:
    urutkanTanggal()
    tampilanData()
case 8:
    tandaiDigunakan()
case 9:
    peringatanKedaluwarsa()
case 10:
    laporanStok()
case 0:
    fmt.Println("👋 Terima kasih! Keluar dari program.")
    return
default:
    fmt.Println("✖ Menu tidak valid.")}
```

DEMONSTRASI APLIKASI



ANALISIS & KESIMPULAN



- Kelebihan Aplikasi mudah dipakai karena menuanya jelas dan sederhana
- Sudah menerapkan berbagai cara pencarian dan pengurutan data yang kita pelajari
- Membantu kita mengelola stok agar lebih teratur dan mengurangi barang terbuang
- Bisa memberikan laporan ringkasan dan peringatan kedaluwarsa

ANALISIS & KESIMPULAN



- Data hilang kalau programnya ditutup
- Tampilannya masih sederhana di layar hitam, belum seperti aplikasi yang ada gambarnya
- Jumlah bahan yang bisa dicatat terbatas (maksimal 100 jenis).

ANALISIS & KESIMPULAN



Aplikasi Manajemen Stok Bahan Makanan ini berhasil diimplementasikan menggunakan GoLang dan mampu memenuhi tujuan dasar pengelolaan stok. Proyek ini memberikan pemahaman yang mendalam tentang penerapan struktur data array dan algoritma pencarian serta pengurutan dalam konteks aplikasi nyata.

**SEKIAN
TERIMAKASIH**