Kelompok 2
Praktikum Pemrograman C

Integrating **System and Calibration** for Your **Medical Devices Check Up**

CalibraMed

“The lack of, or inappropriate, calibration and maintenance of medical devices can seriously jeopardize their safety and performance.”

World Health Organization - 2000

MEET TEAM

Andrea Nathania Justendy

Biomedical Engineering

2306213496

University of Indonesia

Chico Joshua Agung

Biomedical Engineering

2306224865

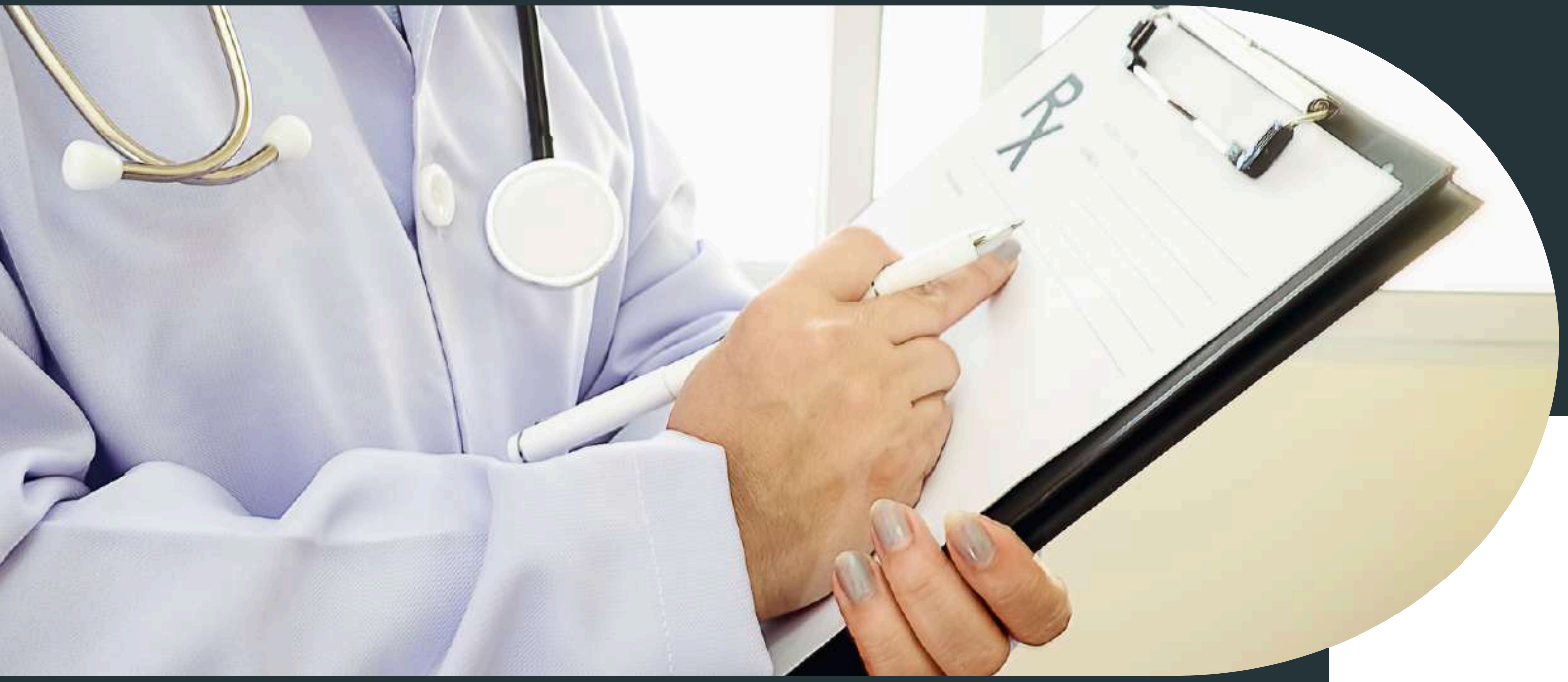
University of Indonesia

Siti Fauzia Putri W

Biomedical Engineering

2306242962

University of Indonesia



Introduction

Saat ini, banyak rumah sakit di Indonesia masih belum memiliki sistem manajemen terintegrasi yang secara khusus mendata dan memantau kondisi alat-alat kesehatannya. Akibatnya, proses pelacakan informasi penting mengenai alat kesehatan menjadi tidak efisien, tidak terdokumentasi dengan baik, dan rawan terjadi kesalahan pencatatan atau kehilangan data.

Borcelle Hospital
Nurturing Care at Every Stage

Tujuan

Tujuan program ini adalah mengembangkan sistem manajemen alat kesehatan rumah sakit yang terintegrasi, efisien, dan mudah digunakan oleh tenaga kesehatan dan teknisi, dengan menyediakan database terpusat untuk mencatat, memantau, dan melacak riwayat alat kesehatan.

Program ini memiliki kemampuan:

- Pencatatan Informasi Alat Kesehatan
- Pelacakan Status Alat
- Riwayat Penggunaan
- Akses Multi-Peran
- Pelacakan Riwayat Pasien Terkait Alat
- Pencarian dan Filter Data

Limitasi

Pada program **CalibraMed: Integrating System and Calibration for Your Medical Devices Check Up** yang telah dibuat terdapat beberapa limitasi yang ada terhadap kondisi dan kerja alat terhadap program yang telah dibuat.

User Interface

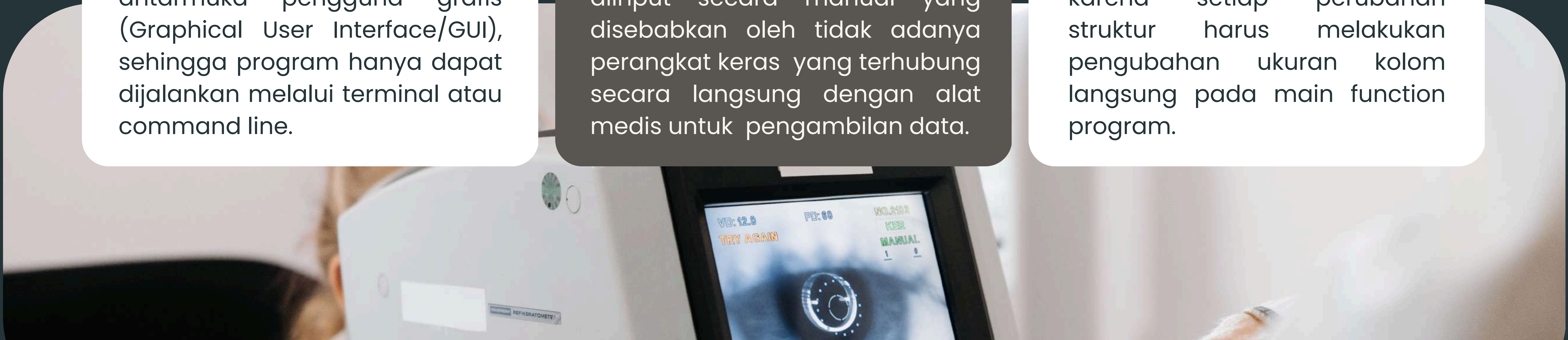
Keterbatasan dari program ini terletak pada tidak tersedianya antarmuka pengguna grafis (Graphical User Interface/GUI), sehingga program hanya dapat dijalankan melalui terminal atau command line.

Manual Input

Program memiliki keterbatasan karena seluruh data harus diinput secara manual yang disebabkan oleh tidak adanya perangkat keras yang terhubung secara langsung dengan alat medis untuk pengambilan data.

Database

Penggunaan & penambahan pada database masih terbatas karena setiap perubahan struktur harus melakukan pengubahan ukuran kolom langsung pada main function program.



So, *who* is this program for?

Biomedical Engineers & Hospital Technicians

With this program, engineers are able to **keep track of medical equipment** throughout a hospital with just a few clicks of their keyboard.

- View **details** of existing medical equipment
- **Add new** medical equipment
- **View and update** medical equipment status

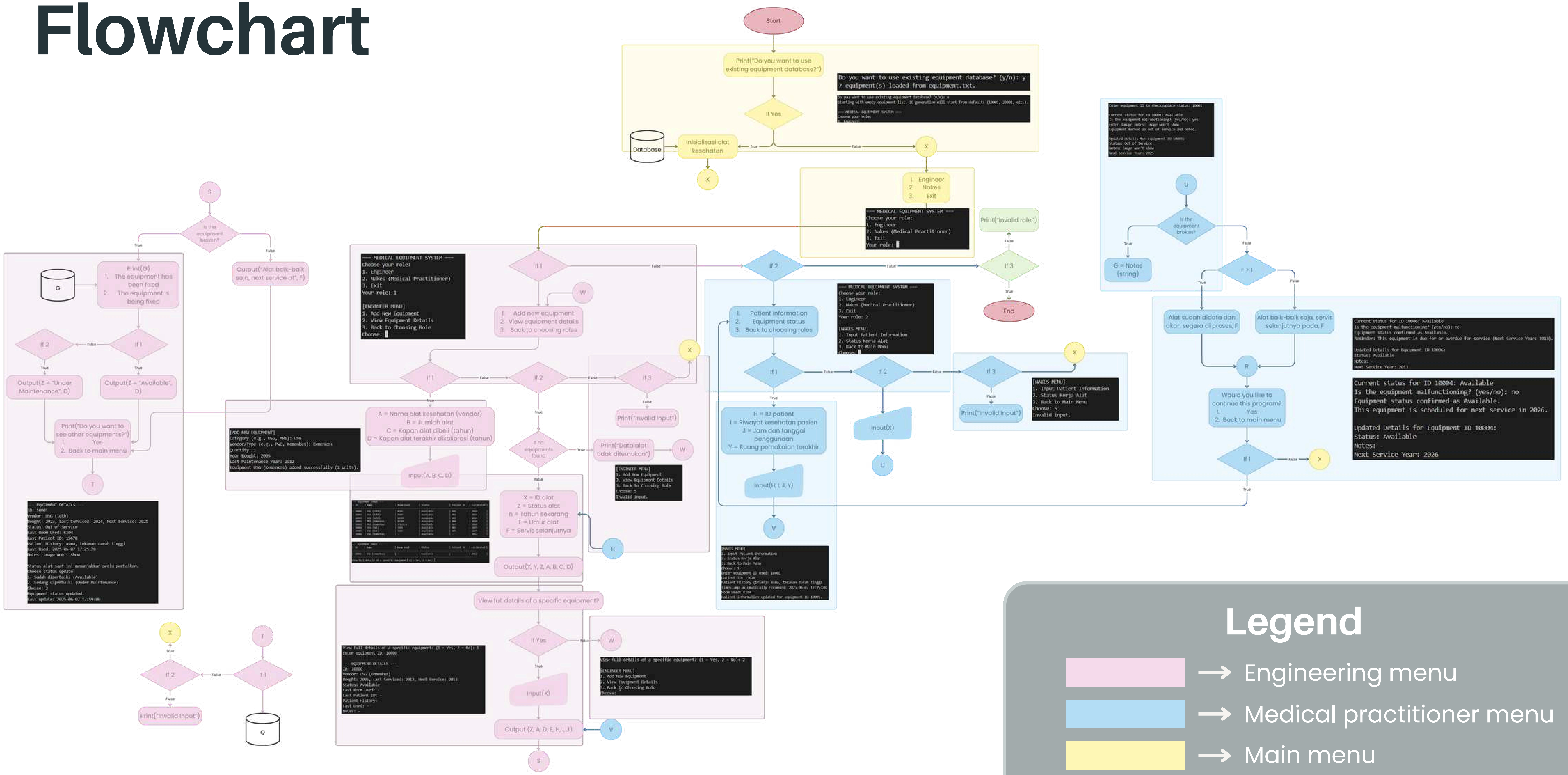
Medical Practitioners & Healthcare Workers

With this program, healthcare workers can **update equipment use history** and **file reports on broken equipment** easily.

- Add **equipment use history**
- **Report broken equipment**
- Add **notes** for engineers **regarding broken equipment**

Hey, I think our **MRI device is broken**.
Sure thing! We'll get it **fixed** right away!

Flowchart


Start
Print("Do you want to use existing equipment database?")
If Yes
True
False
Database
Inisialisasi alat kesehatan
X
1. Engineer
2. Nakes
3. Exit
Print("Invalid role.")
If 1
If 2
If 3
End
W
1. Add new equipment
2. View equipment details
3. Back to choosing roles
A = Nama alat kesehatan (vendor)
B = Jumlah alat
C = Kapan alat dibeli (tahun)
D = Kapan alat terakhir dikalibrasi (tahun)
Input(A, B, C, D)
If no equipments found
Print("Data alat tidak ditemukan")
Print("Invalid input")
X = ID alat
Z = Status alat
n = Tahun sekarang
E = Umur alat
F = Servis selanjutnya
R
Output(X, Y, Z, A, B, C, D)
View full details of a specific equipment?
If Yes
Input(X)
Output (Z, A, D, E, H, I, J)
V
S
Is the equipment broken?
Print(G)
1. The equipment has been fixed
2. The equipment is being fixed
G
Output("Alat baik-baik saja, next service at", F)
Output(Z = "Under Maintenance", D)
Output(Z = "Available", D)
Print("Do you want to see other equipments?")
1. Yes
2. Back to main menu
T
Q
1. Patient information
2. Equipment status
3. Back to choosing roles
H = ID patient
I = Riwayat kesehatan pasien
J = Jam dan tanggal penggunaan
Y = Ruang pemakaian terakhir
Input(H, I, J, Y)
Input(X)
U
Is the equipment broken?
G = Notes (string)
F > 1
Alat sudah didata dan akan segera di proses, F
Alat baik-baik saja, servis selanjutnya pada, F
Would you like to continue this program?
1. Yes
2. Back to main menu

```
Do you want to use existing equipment database? (y/n): y
7 equipment(s) loaded from equipment.txt.
```

```
Do you want to use existing equipment database? (y/n): n
Starting with empty equipment list. ID generation will start from defaults (10001, 20001, etc.).

=== MEDICAL EQUIPMENT SYSTEM ===
Choose your role:
1. Engineer
```

```
=== MEDICAL EQUIPMENT SYSTEM ===
Choose your role:
1. Engineer
2. Nakes (Medical Practitioner)
3. Exit
Your role: █
```

```
=== MEDICAL EQUIPMENT SYSTEM ===
Choose your role:
1. Engineer
2. Nakes (Medical Practitioner)
3. Exit
Your role: 1

[ENGINEER MENU]
1. Add New Equipment
2. View Equipment Details
3. Back to Choosing Role
Choose: █
```

```
[ADD NEW EQUIPMENT]
Category (e.g., USG, MRI): USG
Vendor/Type (e.g., PWC, Komenkes): Komenkes
Quantity: 1
Year Bought: 2005
Last Maintenance Year: 2012
Equipment USG (Komenkes) added successfully (1 units).
```

```
[ENGINEER MENU]
1. Add New Equipment
2. View Equipment Details
3. Back to Choosing Role
Choose: 5
Invalid input.
```

```
View full details of a specific equipment? (1 = Yes, 2 = No): 1
Enter equipment ID: 10006

--- EQUIPMENT DETAILS ---
ID: 10006
Vendor: USG (Komenkes)
Bought: 2005, Last Serviced: 2012, Next Service: 2013
Status: Available
Last Room Used: -
Last Patient ID: -
Patient History: -
Last Used: -
Notes: -
```

```
View full details of a specific equipment? (1 = Yes, 2 = No): 2

[ENGINEER MENU]
1. Add New Equipment
2. View Equipment Details
3. Back to Choosing Role
Choose: █
```

```
--- EQUIPMENT DETAILS ---
ID: 10001
Vendor: USG (Lefth)
Bought: 2023, Last Serviced: 2024, Next Service: 2025
Status: Out of Service
Last Room Used: K304
Last Patient ID: 15678
Patient History: asma, tekanan darah tinggi
Last Used: 2025-06-07 17:25:28
Notes: image won't show

Status alat saat ini menunjukkan perlu perbaikan.
Choose status update:
1. Sudah diperbaiki (Available)
2. Sedang diperbaiki (Under Maintenance)
Choice: 2
Equipment status updated.
Last update: 2025-06-07 17:59:08
```

```
=== MEDICAL EQUIPMENT SYSTEM ===
Choose your role:
1. Engineer
2. Nakes (Medical Practitioner)
3. Exit
Your role: 2

[NAKES MENU]
1. Input Patient Information
2. Status Kerja Alat
3. Back to Main Menu
Choose: █
```

```
[NAKES MENU]
1. Input Patient Information
2. Status Kerja Alat
3. Back to Main Menu
Choose: 5
Invalid input.
```

```
[NAKES MENU]
1. Input Patient Information
2. Status Kerja Alat
3. Back to Main Menu
Choose: 1
Enter equipment ID used: 10001
Patient ID: 15678
Patient History (brief): asma, tekanan darah tinggi
Timestamp automatically recorded: 2025-06-07 17:25:28
Room Used: K304
Patient information updated for equipment ID 10001.
```

```
Enter equipment ID to check/update status: 10001

Current status for ID 10001: Available
Is the equipment malfunctioning? (yes/no): yes
Entry damage notes: image won't show
Equipment marked as out of service and noted.

Updated details for equipment ID 10001:
Status: Out of Service
Notes: image won't show
Next Service Year: 2025
```

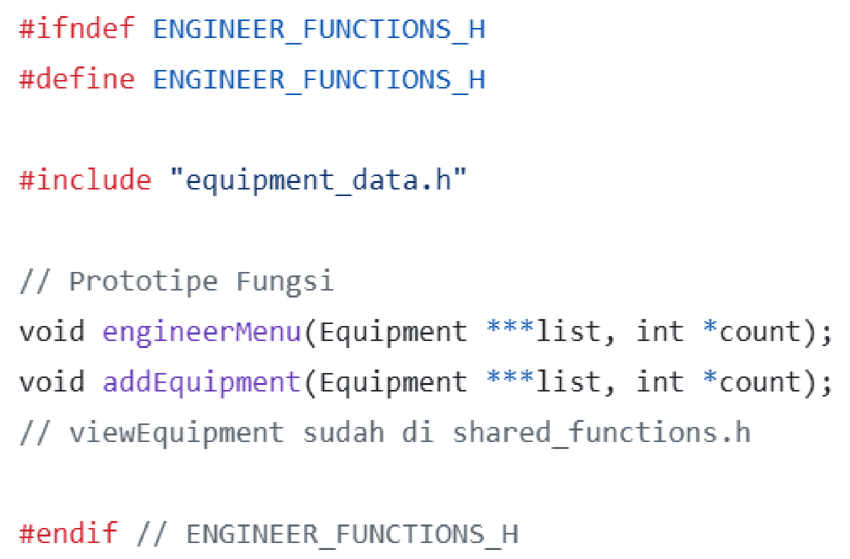
```
Current status for ID 10006: Available
Is the equipment malfunctioning? (yes/no): no
Equipment status confirmed as Available.
Reminder: This equipment is due for or overdue for service (Next Service Year: 2013).

Updated Details for Equipment ID 10006:
Status: Available
Notes: -
Next Service Year: 2013
```

```
Current status for ID 10004: Available
Is the equipment malfunctioning? (yes/no): no
Equipment status confirmed as Available.
This equipment is scheduled for next service in 2026.

Updated Details for Equipment ID 10004:
Status: Available
Notes: -
Next Service Year: 2026
```

Legend

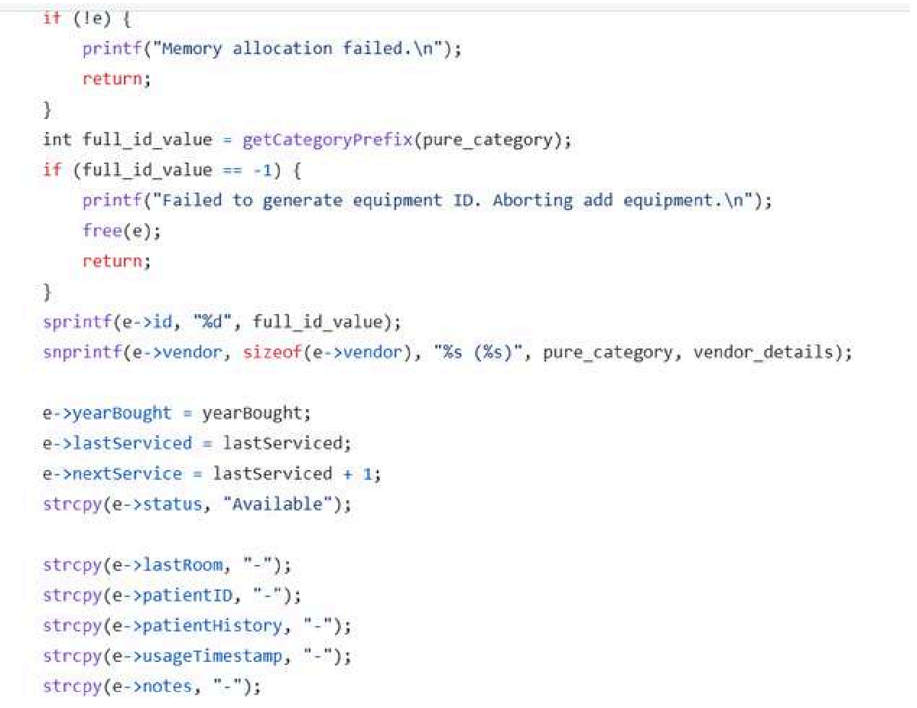
- ▨ → Engineering menu
- ▨ → Medical practitioner menu
- ▨ → Main menu
- ▨ → Exit program

Engineer Function



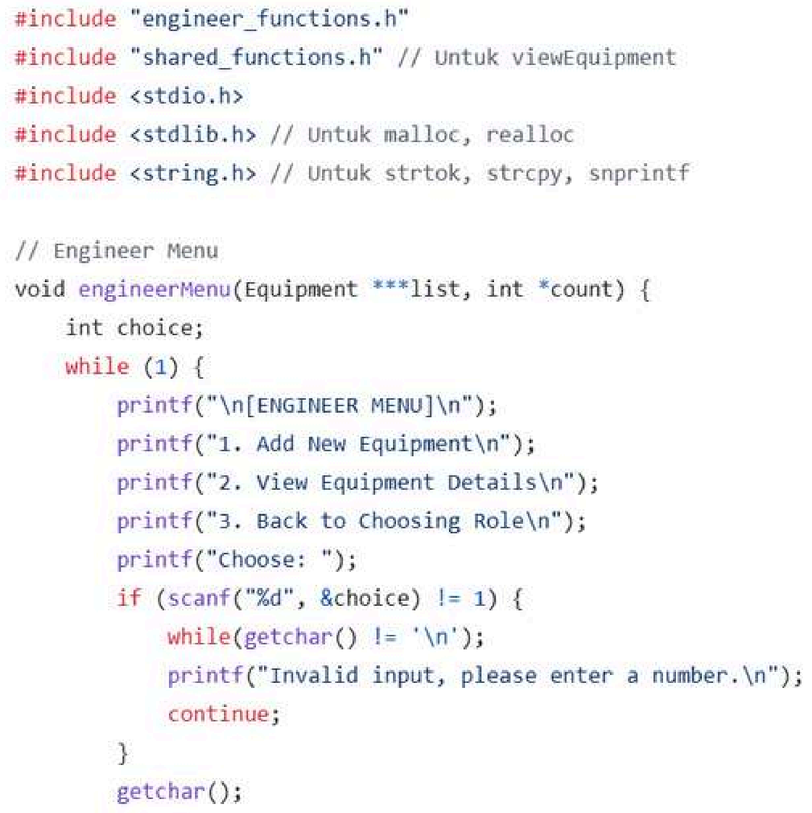
```c
#ifndef ENGINEER_FUNCTIONS_H
#define ENGINEER_FUNCTIONS_H

#include "equipment_data.h"

// Prototipe Fungsi
void engineerMenu(Equipment ***list, int *count);
void addEquipment(Equipment ***list, int *count);
// viewEquipment sudah di shared_functions.h

#endif // ENGINEER_FUNCTIONS_H
```

Call Function

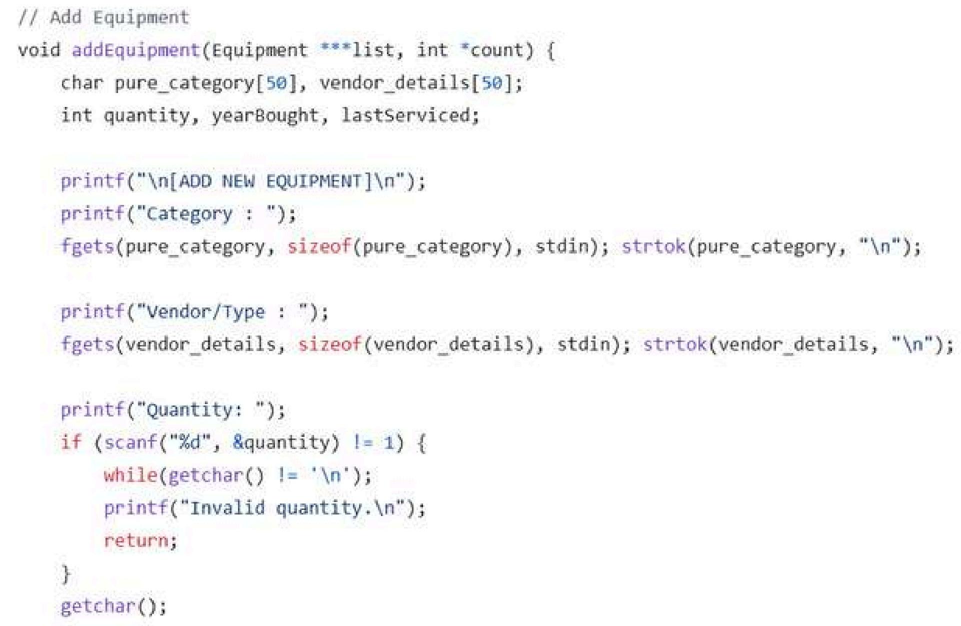


```c
if (!e) {
    printf("Memory allocation failed.\n");
    return;
}
int full_id_value = getCategoryPrefix(pure_category);
if (full_id_value == -1) {
    printf("Failed to generate equipment ID. Aborting add equipment.\n");
    free(e);
    return;
}
sprintf(e->id, "%d", full_id_value);
snprintf(e->vendor, sizeof(e->vendor), "%s (%s)", pure_category, vendor_details);

e->yearBought = yearBought;
e->lastServiced = lastServiced;
e->nextService = lastServiced + 1;
strcpy(e->status, "Available");

strcpy(e->lastRoom, "-");
strcpy(e->patientID, "-");
strcpy(e->patientHistory, "-");
strcpy(e->usageTimestamp, "-");
strcpy(e->notes, "-");
```

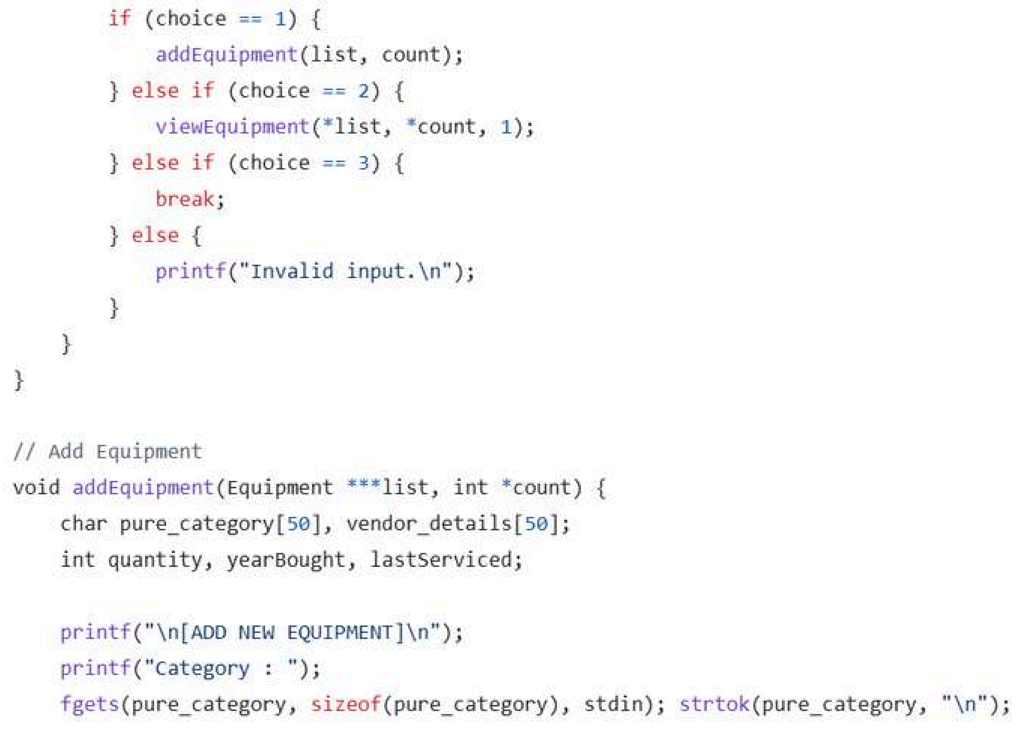
Memory Function Dari Keterangan yang udah di input



```c
#include "engineer_functions.h"
#include "shared_functions.h" // Untuk viewEquipment
#include <stdio.h>
#include <stdlib.h> // Untuk malloc, realloc
#include <string.h> // Untuk strtok, strcpy, snprintf

// Engineer Menu
void engineerMenu(Equipment ***list, int *count) {
    int choice;
    while (1) {
        printf("\n[ENGINEER MENU]\n");
        printf("1. Add New Equipment\n");
        printf("2. View Equipment Details\n");
        printf("3. Back to Choosing Role\n");
        printf("Choose: ");
        if (scanf("%d", &choice) != 1) {
            while(getchar() != '\n');
            printf("Invalid input, please enter a number.\n");
            continue;
        }
        getchar();
```

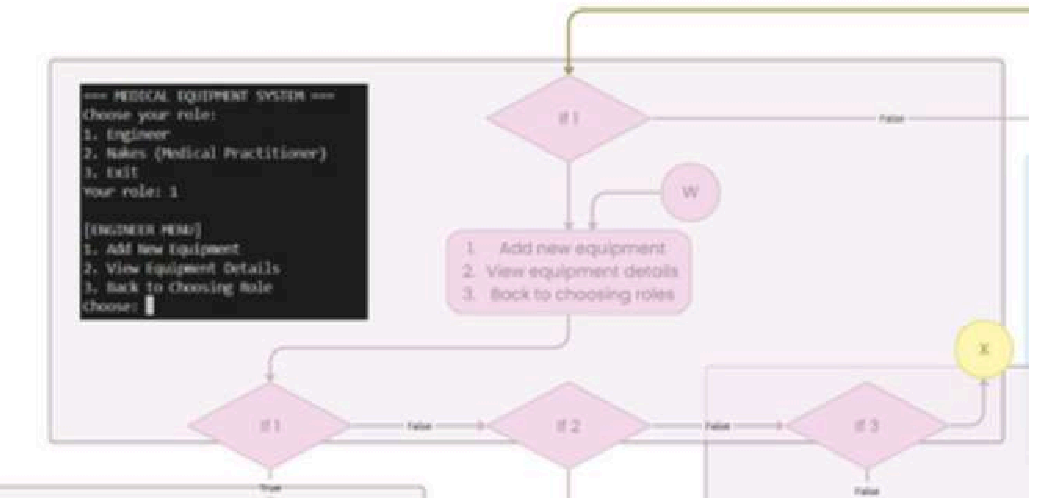


```c
// Add Equipment
void addEquipment(Equipment ***list, int *count) {
    char pure_category[50], vendor_details[50];
    int quantity, yearBought, lastServiced;

    printf("\n[ADD NEW EQUIPMENT]\n");
    printf("Category : ");
    fgets(pure_category, sizeof(pure_category), stdin); strtok(pure_category, "\n");

    printf("Vendor/Type : ");
    fgets(vendor_details, sizeof(vendor_details), stdin); strtok(vendor_details, "\n");

    printf("Quantity: ");
    if (scanf("%d", &quantity) != 1) {
        while(getchar() != '\n');
        printf("Invalid quantity.\n");
        return;
    }
    getchar();
```

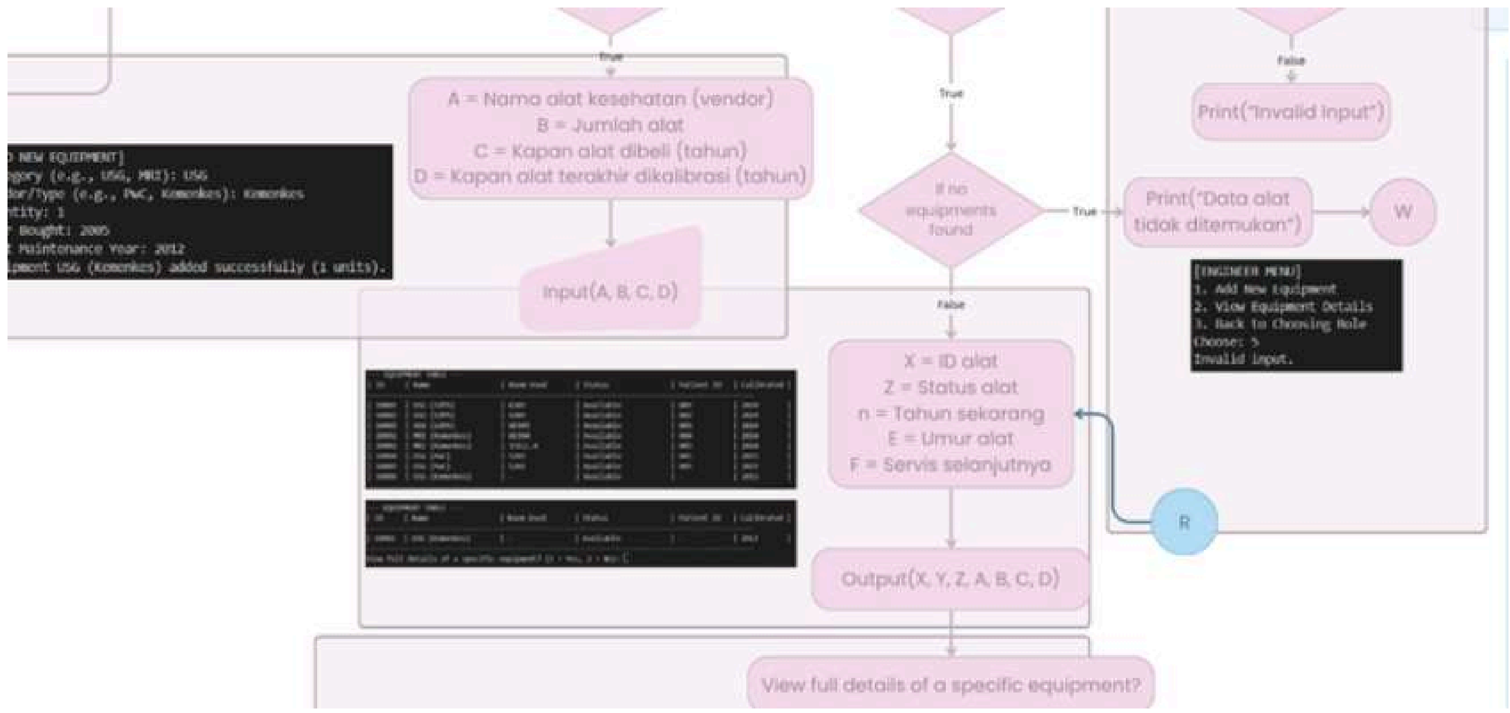


```c
        if (choice == 1) {
            addEquipment(list, count);
        } else if (choice == 2) {
            viewEquipment(*list, *count, 1);
        } else if (choice == 3) {
            break;
        } else {
            printf("Invalid input.\n");
        }
    }
}

// Add Equipment
void addEquipment(Equipment ***list, int *count) {
    char pure_category[50], vendor_details[50];
    int quantity, yearBought, lastServiced;

    printf("\n[ADD NEW EQUIPMENT]\n");
    printf("Category : ");
    fgets(pure_category, sizeof(pure_category), stdin); strtok(pure_category, "\n");
```

Engineer Function



```
=== MEDICAL EQUIPMENT SYSTEM ===
Choose your role:
1. Engineer
2. Nakes (Medical Practitioner)
3. Exit
Your role: 1

[ENGINEER MENU]
1. Add New Equipment
2. View Equipment Details
3. Back to Choosing Role
Choose: 
        // Add to list
        Equipment **temp_list = realloc(*list, (*count + 1) * sizeof(Equipment*));
        if (!temp_list) {
            printf("Memory reallocation failed.\n");
            free(e);
            return;
        }
        *list = temp_list;
        (*list)[*count] = e;
        (*count)++;
    }
    printf("Equipment %s (%s) added successfully (%d units).\n", pure_category, vendor_details, quantity);
}
```



True
A = Nama alat kesehatan (vendor)
B = Jumlah alat
C = Kapan alat dibeli (tahun)
D = Kapan alat terakhir dikalibrasi (tahun)
Input(A, B, C, D)
True
If no equipments found
True
Print("Data alat tidak ditemukan")
W
False
X = ID alat
Z = Status alat
n = Tahun sekarang
E = Umur alat
F = Servis selanjutnya
Output(X, Y, Z, A, B, C, D)
View full details of a specific equipment?
False
Print("Invalid input")
R

```
D NEW EQUIPMENT]
gory (e.g., USG, MRI): USG
dor/Type (e.g., PWC, Kemenkes): Kemenkes
ntity: 1
r Bought: 2005
t Maintenance Year: 2012
ipment USG (Kemenkes) added successfully (1 units).
```

```
[ENGINEER MENU]
1. Add New Equipment
2. View Equipment Details
3. Back to Choosing Role
Choose: 5
Invalid input.
```

Engineer Function

Medical Practitioner Program

This program has **3 main aspects**,

nakes_functions.c

Or in other words, the **source file**. It contains all the relevant functions used in this part of the program.

nakes_functions.o

The **header file** is used to **declare** the three functions from nakes_functions.c, so other parts of the **program can call them**.

nakes_functions.h

This file is more or less the **compiled version** of the source file. It appears automatically to run as part of the full app.

Why do **we need it**?

There are 3 functions used in this aspect of the program,

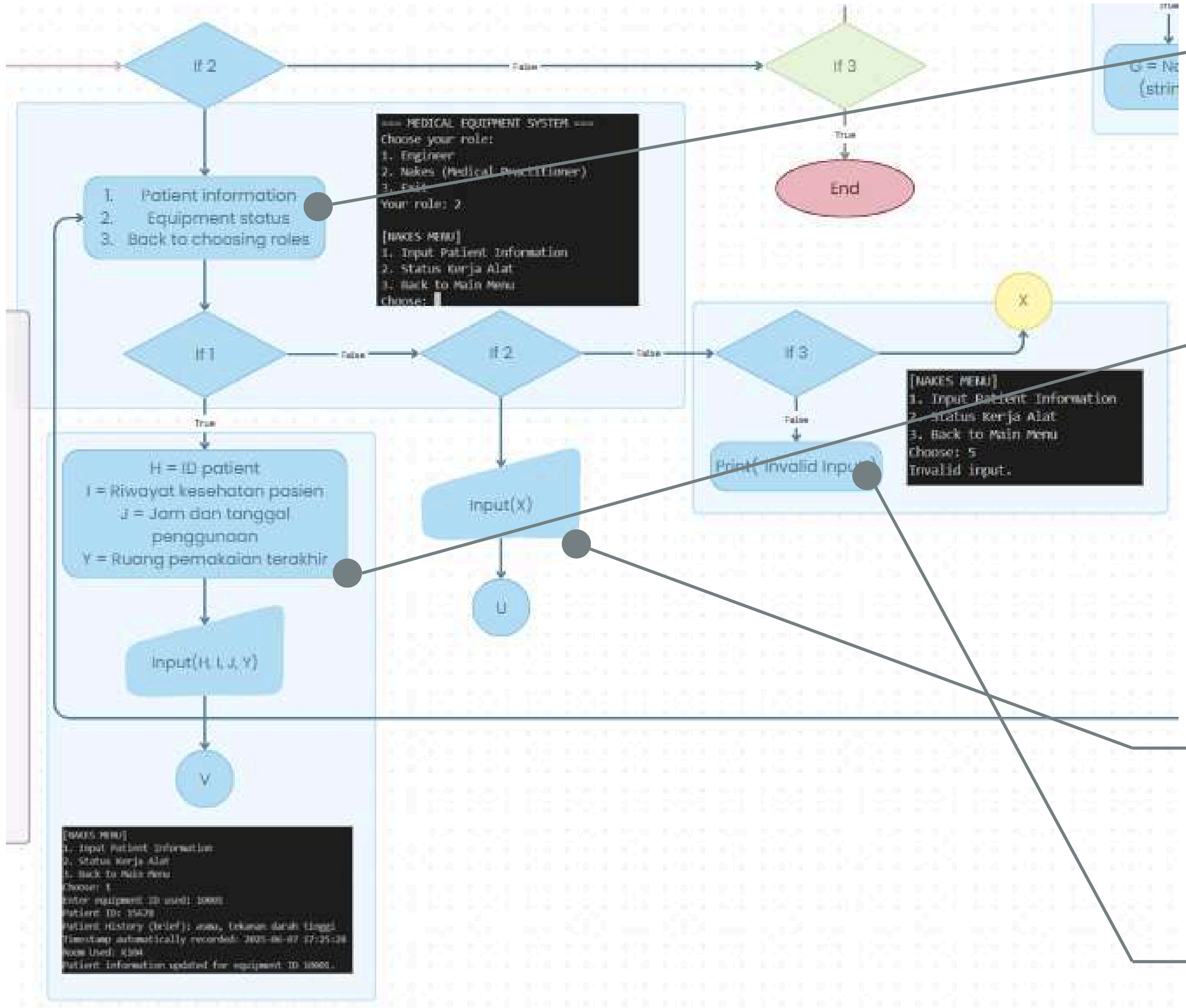
1. **Medical Practitioner Menu**

2. **Input Patient Information**

3. **Equipment Information**

Medical Practitioner Program

How does it **work?**



```
=== MEDICAL EQUIPMENT SYSTEM ===
Choose your role:
1. Engineer
2. Nakes (Medical Practitioner)
3. Exit
Your role: 2

[NAKES MENU]
1. Input Patient Information
2. Status Kerja Alat
3. Back to Main Menu
Choose: 
```

```
[NAKES MENU]
1. Input Patient Information
2. Status Kerja Alat
3. Back to Main Menu
Choose: 5
Invalid input.
```

```
[NAKES MENU]
1. Input Patient Information
2. Status Kerja Alat
3. Back to Main Menu
Choose: 1
Enter equipment ID used: 10001
Patient ID: 15678
Patient History (brief): asma, tekanan darah tinggi
Timestamp automatically recorded: 2025-06-07 17:25:28
Room Used: R104
Patient Information updated for equipment ID 10001.
```

Menu

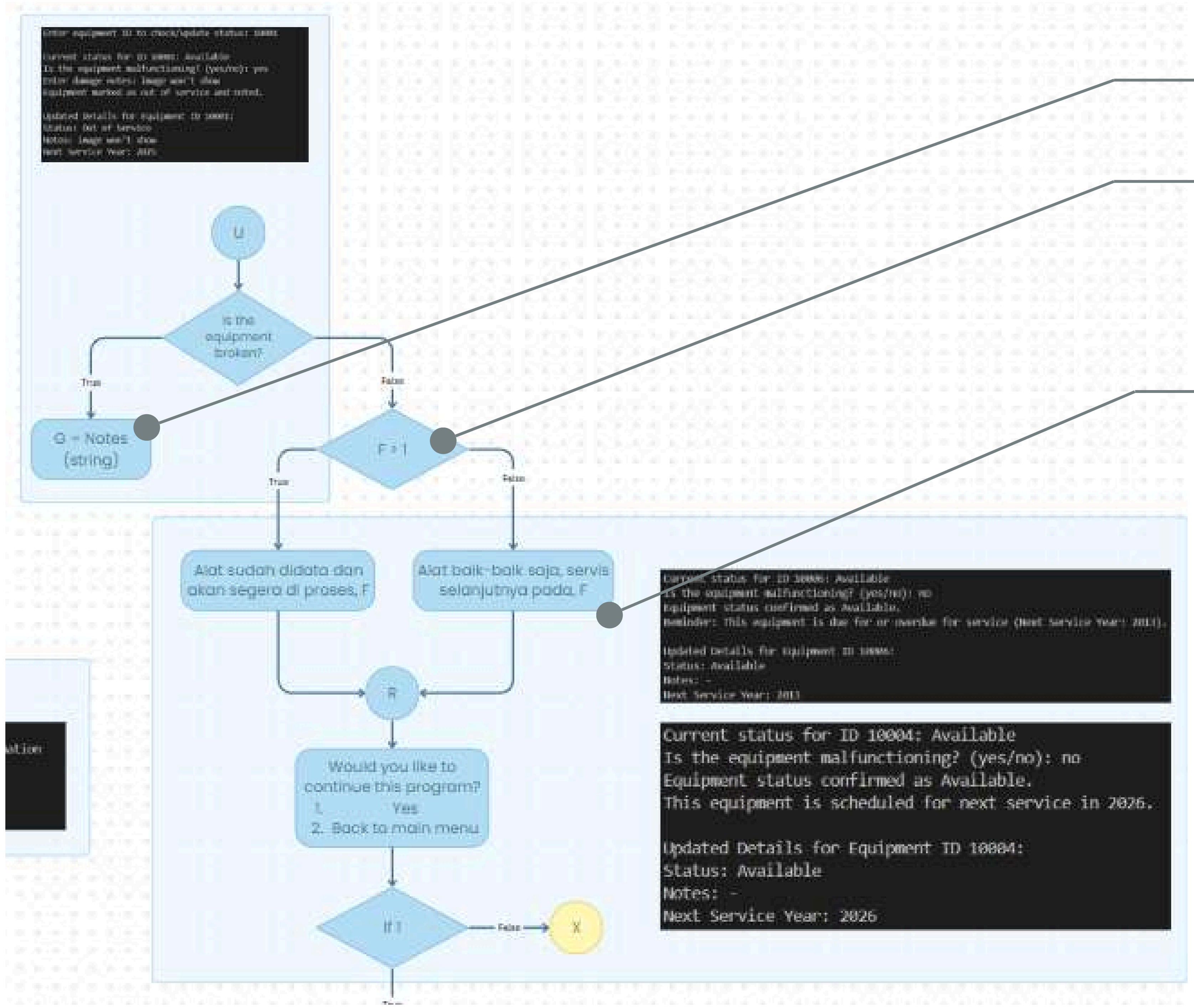
Input equipment history

Report equipment status

Error handling

Medical Practitioner Program

How does it **work?**



```
Enter equipment ID to check/update status: 10001

Current status for ID 10001: Available
Is the equipment malfunctioning? (yes/no): yes
Enter damage notes: image won't show
Equipment marked as out of service and noted.

Updated Details for Equipment ID 10001:
Status: Out of Service
Notes: image won't show
Next Service Year: 2025
```

```
Current status for ID 10006: Available
Is the equipment malfunctioning? (yes/no): no
Equipment status confirmed as Available.
Reminder: This equipment is due for or overdue for service (Next Service Year: 2013).

Updated Details for Equipment ID 10006:
Status: Available
Notes: -
Next Service Year: 2013
```

```
Current status for ID 10004: Available
Is the equipment malfunctioning? (yes/no): no
Equipment status confirmed as Available.
This equipment is scheduled for next service in 2026.

Updated Details for Equipment ID 10004:
Status: Available
Notes: -
Next Service Year: 2026
```

U
Is the equipment broken?
True
False
G = Notes (string)
F > 1
True
False
Alat sudah didata dan akan segera di proses, F
Alat baik-baik saja, servis selanjutnya pada, F
R
Would you like to continue this program?
1. Yes
2. Back to main menu
If 1
False
X
True
ation

Broken equipment report

Check equipment status

Status update based on last callibration

Medical Practitioner Program

nakes_function.c

Libraries

```
#include "nakes_functions.h"
#include "shared_functions.h" // Untuk viewEquipment
#include "utils.h"      // Untuk getCurrentDateTime
#include <stdio.h>
#include <stdlib.h> // Untuk time_t, localtime, tm
#include <string.h> // Untuk strcmp, strcpy, strtok
#include <time.h> // Untuk time, localtime
```

nakesMenu

```
void nakesMenu(Equipment **list, int count) {
 int choice;
 while (1) {
  printf("\n[NAKES MENU]\n");
  printf("1. Input Patient Information\n");
  ...
  if (scanf("%d", &choice) != 1) {
   while(getchar() != '\n');
   printf("Invalid input, please enter a number.\n");
   ...
}
  if (choice == 1) {
   inputPatientInformation(list, count);
  } else if (choice == 2) {
   viewEquipment(list, count, 0);
   checkEquipmentStatus(list, count, 0);
  } else if (choice == 3) {
   break;
  } else {
   printf("Invalid input.\n");
  }
 }
}
```

inputPatientInformation

```
void inputPatientInformation(Equipment **list, int count) {
 if (count == 0) {
  printf("No equipment available to assign patient information.\n");
  return;
 }
 ...
 printf("Enter equipment ID used: ");
 fgets(equipmentID, sizeof(equipmentID), stdin); strtok(equipmentID, "\n");
 Equipment *found = NULL;
 for (int i = 0; i < count; i++) {
  if (strcmp(list[i]->id, equipmentID) == 0) {
   found = list[i];
   break;
  }
 }
 if (!found) {
  printf("Equipment not found.\n");
  return;
 }
 printf("Patient ID: ");
 fgets(found->patientID, sizeof(found->patientID), stdin); strtok(found->patientID, "\n");
 printf("Patient History (brief): ");
 ...
 printf("Patient information updated for equipment ID %s.\n", found->id);
}
```

Medical Practitioner Program

nakes_function.c

inputPatientInformation

```
void checkEquipmentStatus(Equipment **list, int count, int isEngineer) {
  if (count == 0) {
    printf("No equipment available to check status.\n");
    return;
  }
  char equipmentID[20];
  printf("Enter equipment ID to check/update status: ");
  fgets(equipmentID, sizeof(equipmentID), stdin); strtok(equipmentID, "\n");

  for (int i = 0; i < count; i++) {
    Equipment *found = list[i];
    if (strcmp(equipmentID, found->id) == 0) {
      printf("\nCurrent status for ID %s: %s\n", found->id, found->status);
      char broken[10];
      printf("Is the equipment malfunctioning? (yes/no): ");
      fgets(broken, sizeof(broken), stdin); strtok(broken, "\n");

      time_t t = time(NULL);
      struct tm tm_info = *localtime(&t);
      int currentYear = tm_info.tm_year + 1900;

      if (strcmp(broken, "yes") == 0) {
        printf("Enter damage notes: ");
        fgets(found->notes, sizeof(found->notes), stdin); strtok(found->notes, "\n");
        strcpy(found->status, "Out of Service");
```

```
found->nextService = found->lastServiced + 1;

        printf("Equipment marked as out of service and noted.\n");
      } else if (strcmp(broken, "no") == 0) {
        if (strcmp(found->status, "Out of Service") == 0 || strcmp(found->status, "Under
Maintenance") == 0) {
           printf("Equipment status is '%s'. Please contact an engineer for status update
if it's now working.\n", found->status);
        } else {
          strcpy(found->status, "Available");
          strcpy(found->notes, "-");
          printf("Equipment status confirmed as Available.\n");
        }
        if (currentYear >= found->nextService) {
           printf("Reminder: This equipment is due for or overdue for service (Next
Service Year: %d).\n", found->nextService);
        } else {
           printf("This equipment is scheduled for next service in %d.\n", found-
>nextService);
        }
      } else {
        printf("Invalid input for malfunctioning status. No changes made.\n");
      }
      printf("\nUpdated Details for Equipment ID %s:\n", found->id);
      printf("Status: %s\n", found->status);
```

```
 printf("Notes: %s\n", found->notes);
 printf("Next Service Year: %d\n", found->nextService);
 return;
 }
 }
 printf("Equipment not found.\n");
}
```

Medical Practitioner Program

nakes_function.h

```
#ifndef NAKES_FUNCTIONS_H
#define NAKES_FUNCTIONS_H

#include "equipment_data.h" // Butuh definisi Equipment

// Prototipe Fungsi
void nakesMenu(Equipment **list, int count);
void inputPatientInformation(Equipment **list, int count);
void checkEquipmentStatus(Equipment **list, int count, int isEngineer);
// viewEquipment sudah di shared_functions.h

#endif // NAKES_FUNCTIONS_H
```

Equipment

This program has **3 main aspects**,

equipment_data.c

Or in other words, the **source file**. It contains all the relevant functions used in this part of the program.

equipment_data.o

The **header file** is used to **declare** the three functions from nakes_functions.c, so other parts of the **program can call them**.

equipment_data.h

This file is more or less the **compiled version** of the source file. It appears automatically to run as part of the full app.

Why do **we need it**?

The **equipment_data** module is like a **utility hub** for **viewing category ID and for looping ID in equipment**. It's meant to be used by adding equipment ID.

EquipmentCategoryID

idGenerators[100]

Equipment

equipment_data.c

Libraries

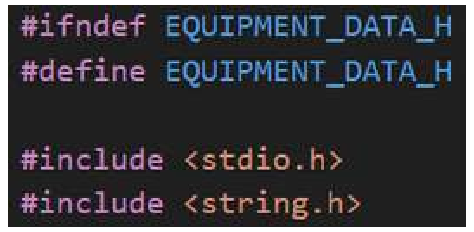
```c
#include "equipment_data.h"
#include <string.h> // Included via equipment_data.h
#include <stdio.h> // For fprintf, stderr if errors occur
```

Equipment Category

```c
EquipmentCategoryID idGenerators[100];
int categoryCount = 0;

int getCategoryPrefix(const char *pureCategoryName) {
  for (int i = 0; i < categoryCount; i++) {
    if (strcmp(idGenerators[i].category, pureCategoryName) == 0) {
      idGenerators[i].currentCount++;
      return idGenerators[i].baseID + idGenerators[i].currentCount;
    }
  }
  if (categoryCount < 100) {
    strcpy(idGenerators[categoryCount].category, pureCategoryName);
    idGenerators[categoryCount].baseID = (categoryCount + 1) * 10000;
    idGenerators[categoryCount].currentCount = 0;
    categoryCount++;
    idGenerators[categoryCount - 1].currentCount++;
    return idGenerators[categoryCount - 1].baseID + idGenerators[categoryCount - 1].currentCount;
  } else {
    fprintf(stderr, "Error: Maximum number of categories reached. Cannot add new category %s.\n", pureCategoryName);
    return -1;
  }
}
```

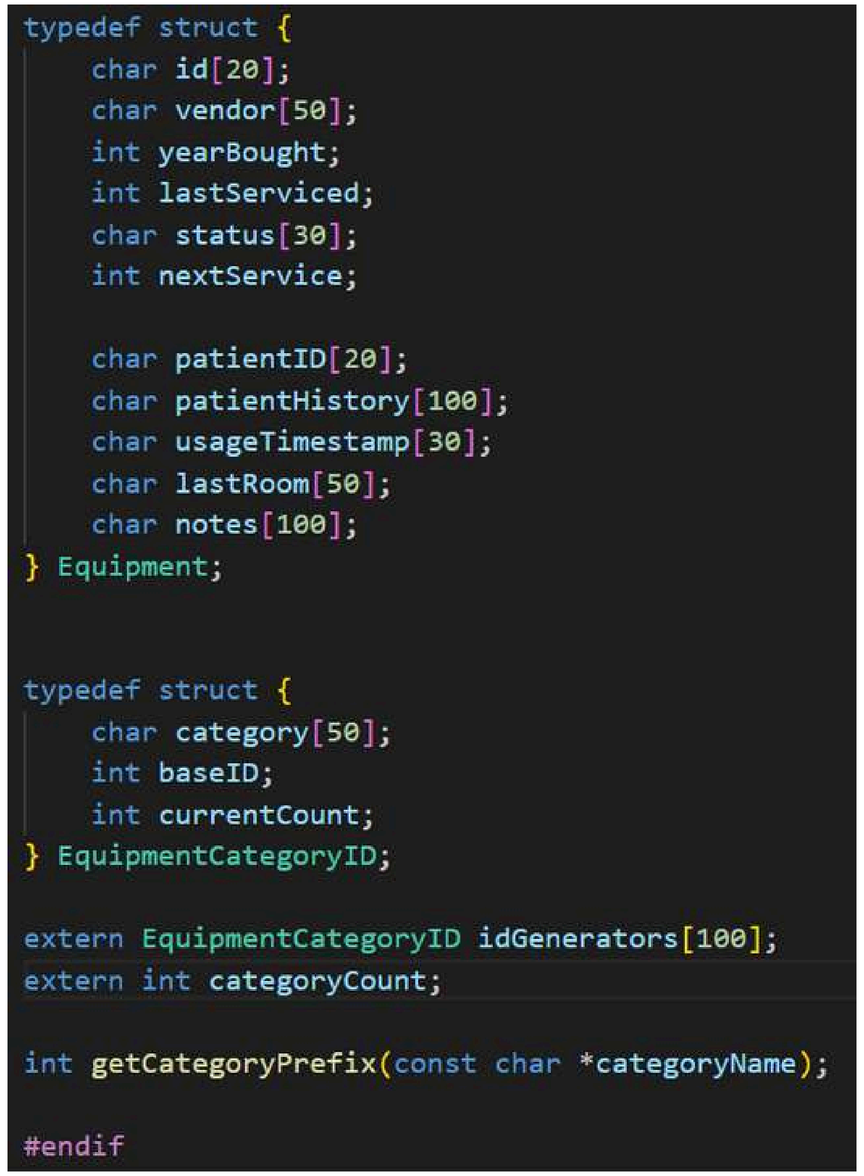
equipment_data.h

Libraries



```c
#ifndef EQUIPMENT_DATA_H
#define EQUIPMENT_DATA_H

#include <stdio.h>
#include <string.h>
```

Equipment Category



```c
typedef struct {
    char id[20];
    char vendor[50];
    int yearBought;
    int lastServiced;
    char status[30];
    int nextService;

    char patientID[20];
    char patientHistory[100];
    char usageTimestamp[30];
    char lastRoom[50];
    char notes[100];
} Equipment;


typedef struct {
    char category[50];
    int baseID;
    int currentCount;
} EquipmentCategoryID;

extern EquipmentCategoryID idGenerators[100];
extern int categoryCount;

int getCategoryPrefix(const char *categoryName);

#endif
```

Shared Functions

This program has **3 main aspects**,

shared_functions.c

Or in other words, the **source file**. It contains all the relevant functions used in this part of the program.

shared_functions.h

The **header file** is used to **declare** the three functions from nakes_functions.c, so other parts of the **program can call them**.

shared_functions.o

This file is more or less the **compiled version** of the source file. It appears automatically to run as part of the full app.

Why do **we need it**?

The **shared_functions** module is like a **utility hub** for **viewing and interacting with medical equipment data**. It's meant to be used by both medical staff (nakes) and engineers, hence the name "**shared**". It only consists of one function,

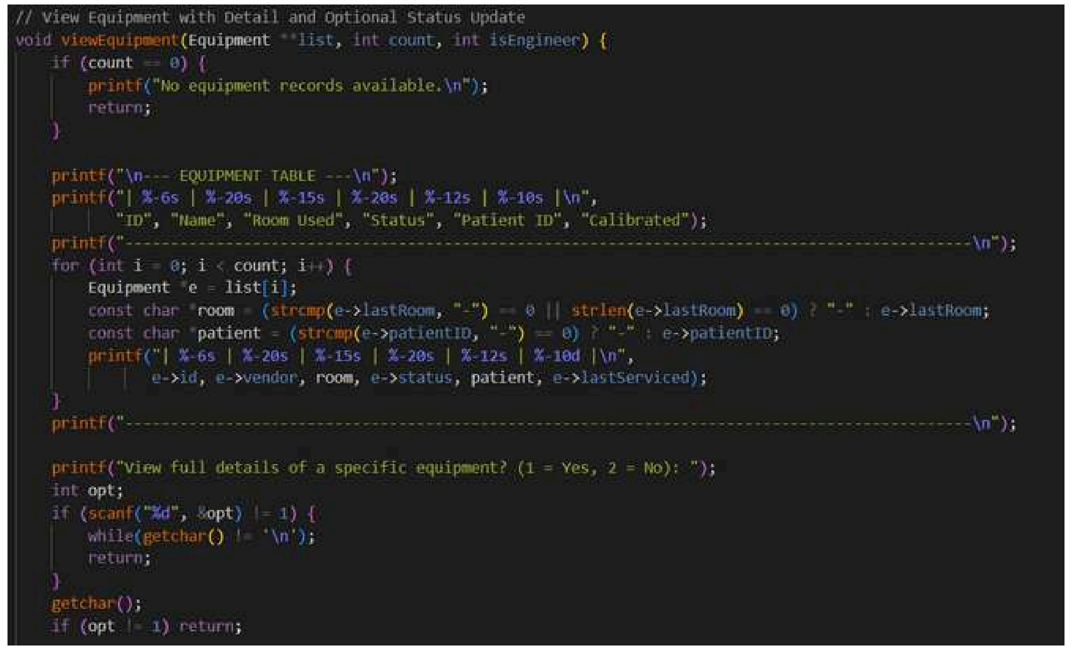
viewEquipment Function

Shared Functions

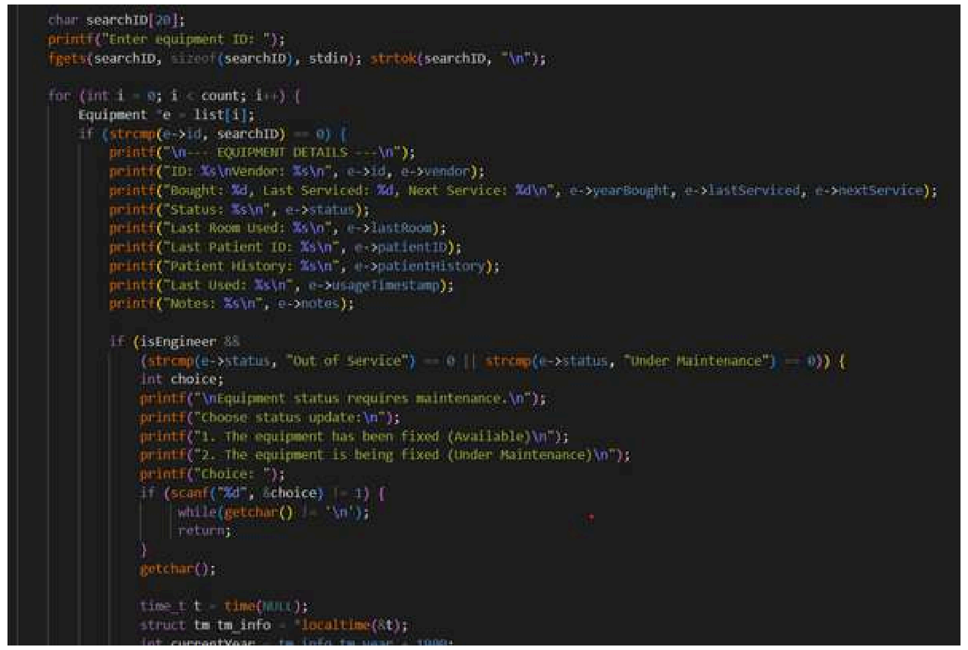
shared_functions.c

viewEquipment

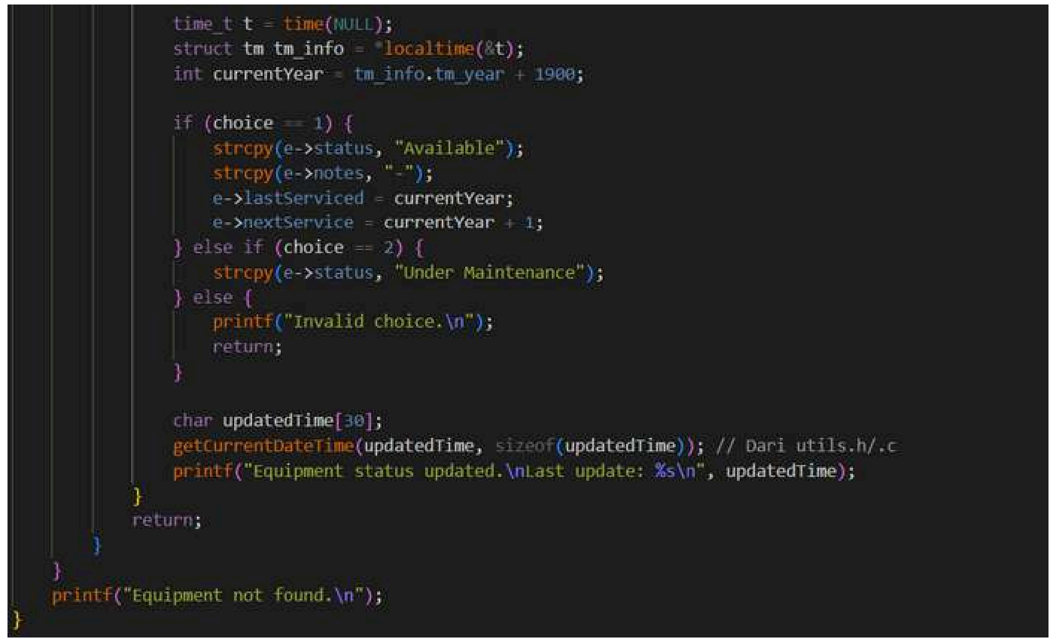
It handles presenting equipment in a clear and readable table format, to allow interactive exploration of individual equipment records, and to give engineers access to update status (like fixing broken equipment).



```c
// View Equipment with Detail and Optional Status Update
void viewEquipment(Equipment **list, int count, int isEngineer) {
    if (count == 0) {
        printf("No equipment records available.\n");
        return;
    }

    printf("\n--- EQUIPMENT TABLE ---\n");
    printf("| %-6s | %-20s | %-15s | %-20s | %-12s | %-10s |\n",
           "ID", "Name", "Room Used", "Status", "Patient ID", "Calibrated");
    printf("---------------------------------------------------------------------------------------------\n");
    for (int i = 0; i < count; i++) {
        Equipment *e = list[i];
        const char *room = (strcmp(e->lastRoom, "-") == 0 || strlen(e->lastRoom) == 0) ? "-" : e->lastRoom;
        const char *patient = (strcmp(e->patientID, "-") == 0) ? "-" : e->patientID;
        printf("| %-6s | %-20s | %-15s | %-20s | %-12s | %-10d |\n",
               e->id, e->vendor, room, e->status, patient, e->lastServiced);
    }
    printf("---------------------------------------------------------------------------------------------\n");

    printf("View full details of a specific equipment? (1 = Yes, 2 = No): ");
    int opt;
    if (scanf("%d", &opt) != 1) {
        while(getchar() != '\n');
        return;
    }
    getchar();
    if (opt != 1) return;
```

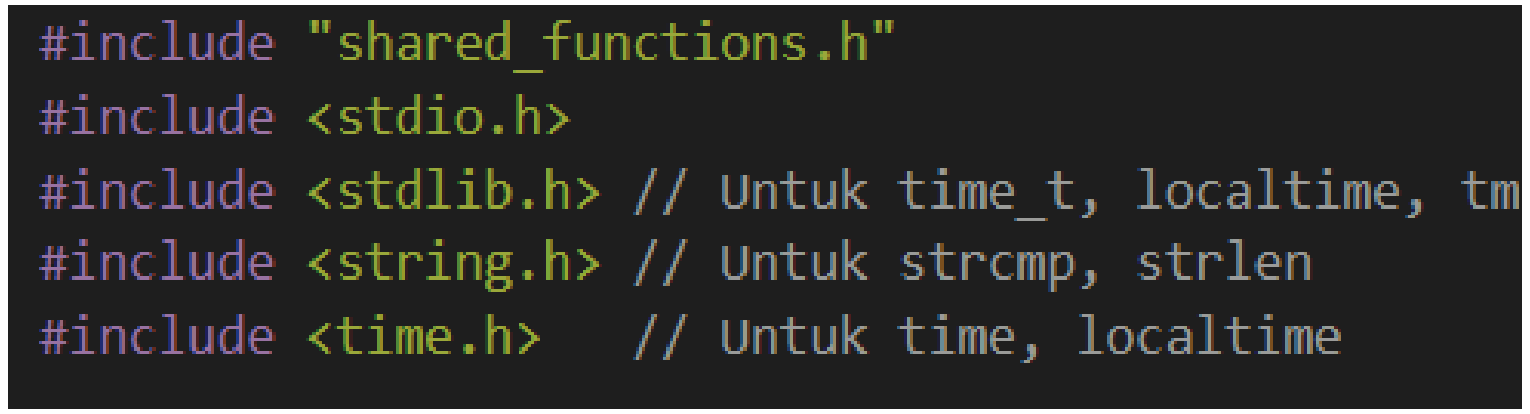


```c
    char searchID[20];
    printf("Enter equipment ID: ");
    fgets(searchID, sizeof(searchID), stdin); strtok(searchID, "\n");

    for (int i = 0; i < count; i++) {
        Equipment *e = list[i];
        if (strcmp(e->id, searchID) == 0) {
            printf("\n--- EQUIPMENT DETAILS ---\n");
            printf("ID: %s\nVendor: %s\n", e->id, e->vendor);
            printf("Bought: %d, Last Serviced: %d, Next Service: %d\n", e->yearBought, e->lastServiced, e->nextService);
            printf("Status: %s\n", e->status);
            printf("Last Room Used: %s\n", e->lastRoom);
            printf("Last Patient ID: %s\n", e->patientID);
            printf("Patient History: %s\n", e->patientHistory);
            printf("Last Used: %s\n", e->usageTimestamp);
            printf("Notes: %s\n", e->notes);

            if (isEngineer &&
                (strcmp(e->status, "Out of Service") == 0 || strcmp(e->status, "Under Maintenance") == 0)) {
                int choice;
                printf("\nEquipment status requires maintenance.\n");
                printf("Choose status update:\n");
                printf("1. The equipment has been fixed (Available)\n");
                printf("2. The equipment is being fixed (Under Maintenance)\n");
                printf("Choice: ");
                if (scanf("%d", &choice) != 1) {
                    while(getchar() != '\n');
                    return;
                }
                getchar();

                time_t t = time(NULL);
                struct tm tm_info = *localtime(&t);
                int currentYear = tm_info.tm_year + 1900;
```



```c
                time_t t = time(NULL);
                struct tm tm_info = *localtime(&t);
                int currentYear = tm_info.tm_year + 1900;

                if (choice == 1) {
                    strcpy(e->status, "Available");
                    strcpy(e->notes, "-");
                    e->lastServiced = currentYear;
                    e->nextService = currentYear + 1;
                } else if (choice == 2) {
                    strcpy(e->status, "Under Maintenance");
                } else {
                    printf("Invalid choice.\n");
                    return;
                }

                char updatedTime[30];
                getCurrentDateTime(updatedTime, sizeof(updatedTime)); // Dari utils.h/.c
                printf("Equipment status updated.\nLast update: %s\n", updatedTime);
            }
            return;
        }
    }
    printf("Equipment not found.\n");
}
```

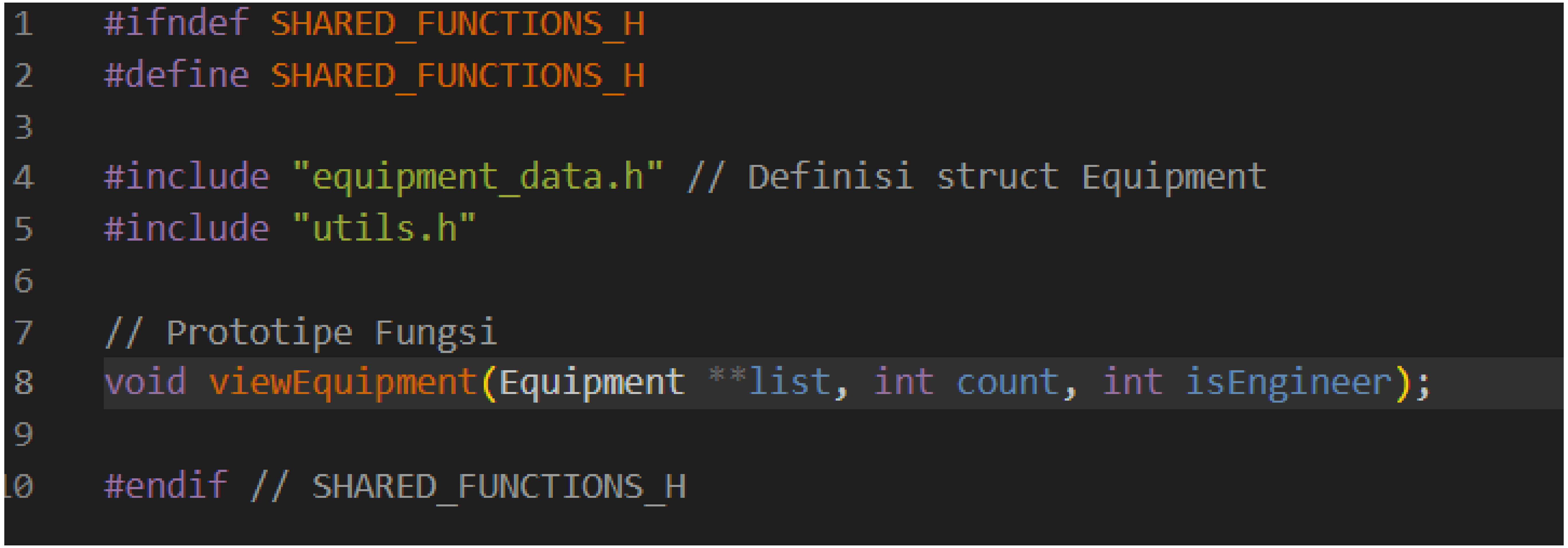
Libraries used



```c
#include "shared_functions.h"
#include <stdio.h>
#include <stdlib.h> // Untuk time_t, localtime, tm
#include <string.h> // Untuk strcmp, strlen
#include <time.h>   // Untuk time, localtime
```

Shared Functions

shared_functions.h



```c
#ifndef SHARED_FUNCTIONS_H
#define SHARED_FUNCTIONS_H

#include "equipment_data.h" // Definisi struct Equipment
#include "utils.h"

// Prototipe Fungsi
void viewEquipment(Equipment **list, int count, int isEngineer);

#endif // SHARED_FUNCTIONS_H
```

Utility

This program has **3 main aspects**,

utils.c

Or in other words, the **source file**. It contains all the relevant functions used in this part of the program.

utils.h

The **header file** is used to **declare** the three functions from nakes_functions.c, so other parts of the **program can call them**.

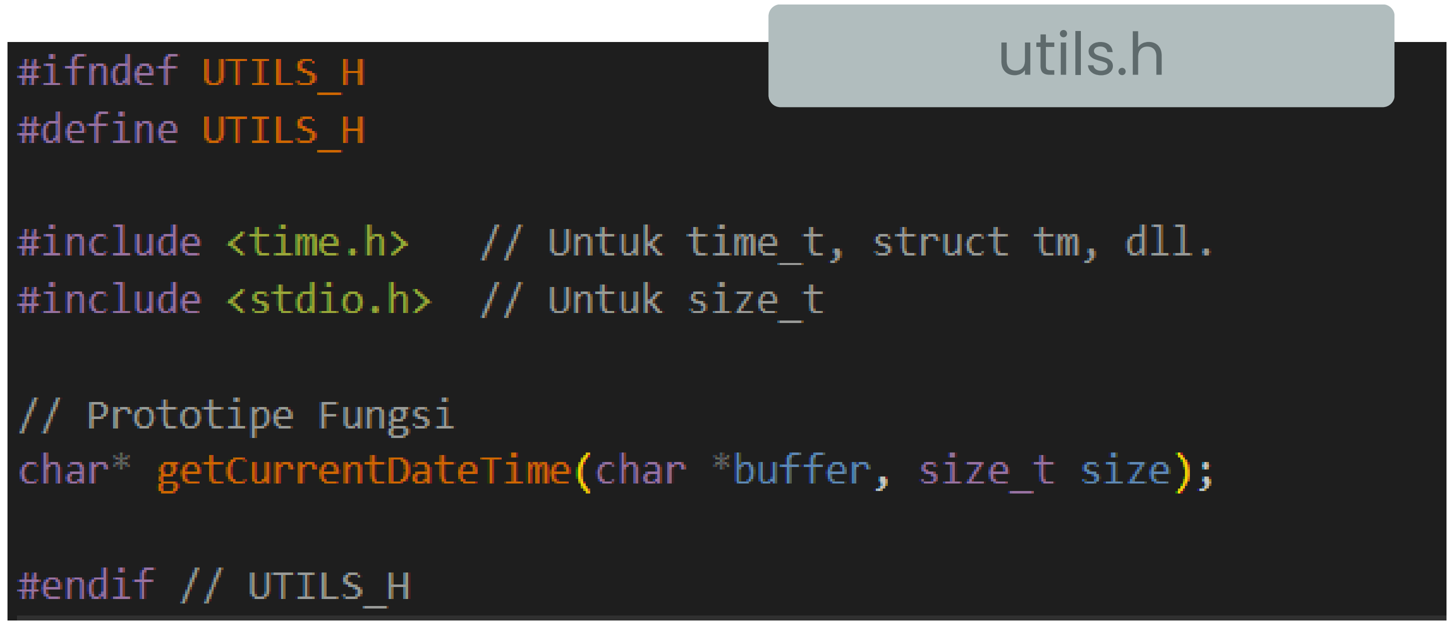
utils.o

This file is more or less the **compiled version** of the source file. It appears automatically to run as part of the full app.

Why do **we need it**?

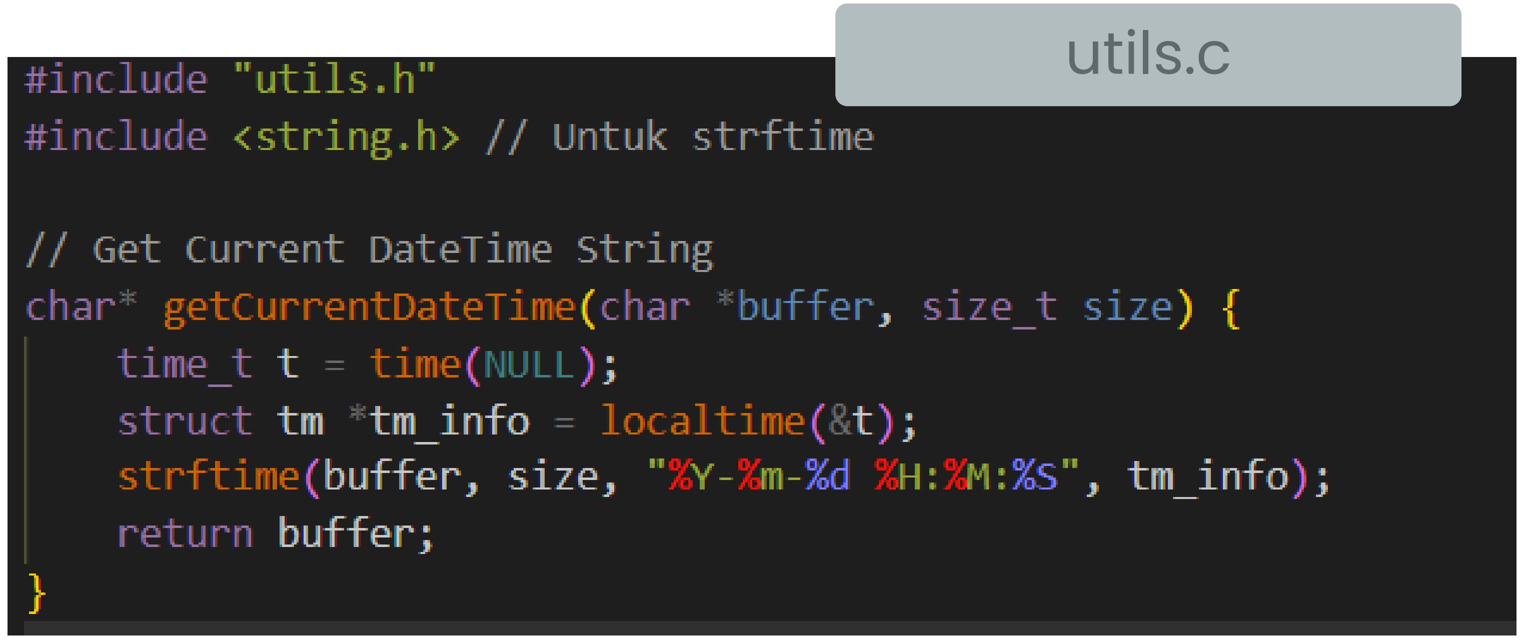
Utility provides a function used to get the current date and time. We separated it to get a more clean and modular program.

utils.h



```c
#ifndef UTILS_H
#define UTILS_H

#include <time.h>   // Untuk time_t, struct tm, dll.
#include <stdio.h>  // Untuk size_t

// Prototipe Fungsi
char* getCurrentDateTime(char *buffer, size_t size);

#endif // UTILS_H
```

utils.c

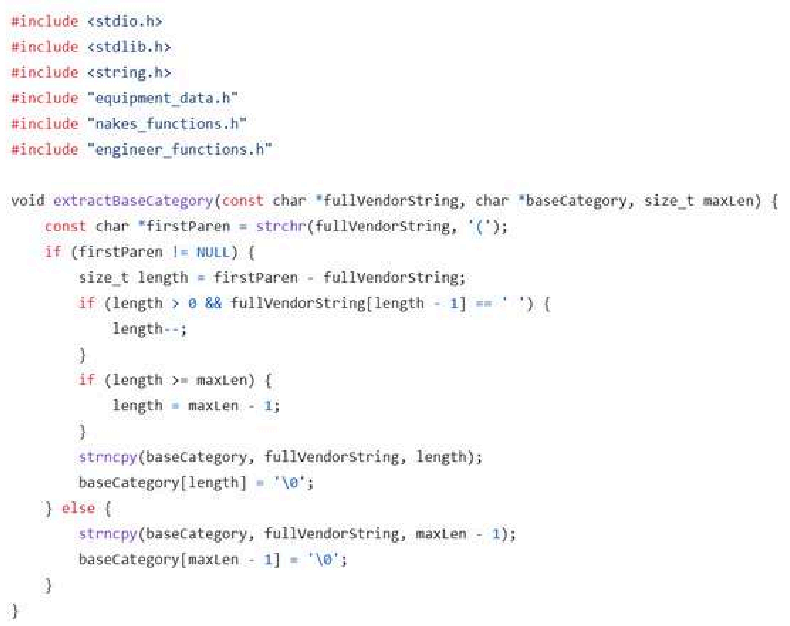


```c
#include "utils.h"
#include <string.h> // Untuk strftime

// Get Current DateTime String
char* getCurrentDateTime(char *buffer, size_t size) {
    time_t t = time(NULL);
    struct tm *tm_info = localtime(&t);
    strftime(buffer, size, "%Y-%m-%d %H:%M:%S", tm_info);
    return buffer;
}
```

MAIN FUNCTION

Main.c

Kategori Function Alat

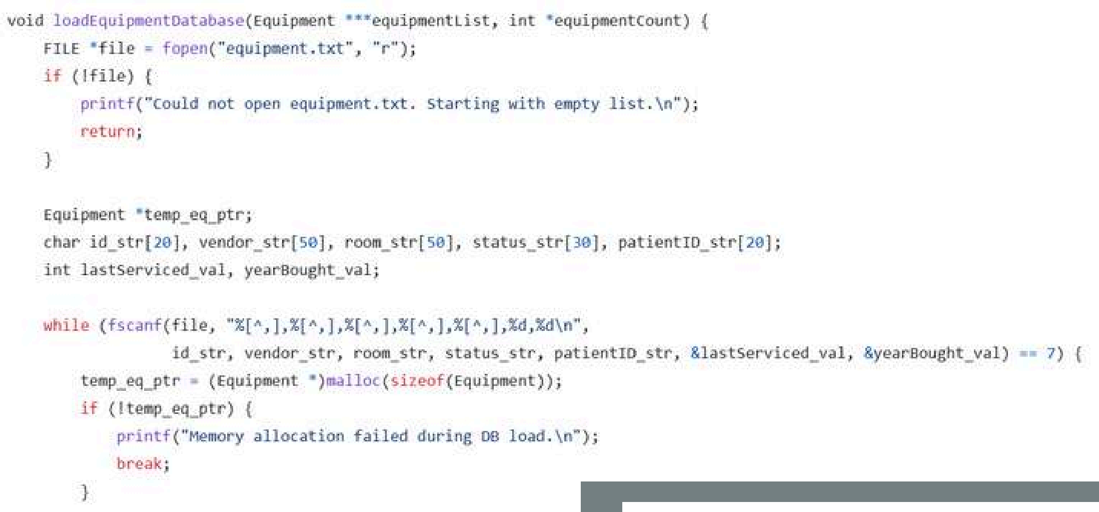


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "equipment_data.h"
#include "nakes_functions.h"
#include "engineer_functions.h"

void extractBaseCategory(const char *fullVendorString, char *baseCategory, size_t maxLen) {
    const char *firstParen = strchr(fullVendorString, '(');
    if (firstParen != NULL) {
        size_t length = firstParen - fullVendorString;
        if (length > 0 && fullVendorString[length - 1] == ' ') {
            length--;
        }
        if (length >= maxLen) {
            length = maxLen - 1;
        }
        strncpy(baseCategory, fullVendorString, length);
        baseCategory[length] = '\0';
    } else {
        strncpy(baseCategory, fullVendorString, maxLen - 1);
        baseCategory[maxLen - 1] = '\0';
    }
}
```

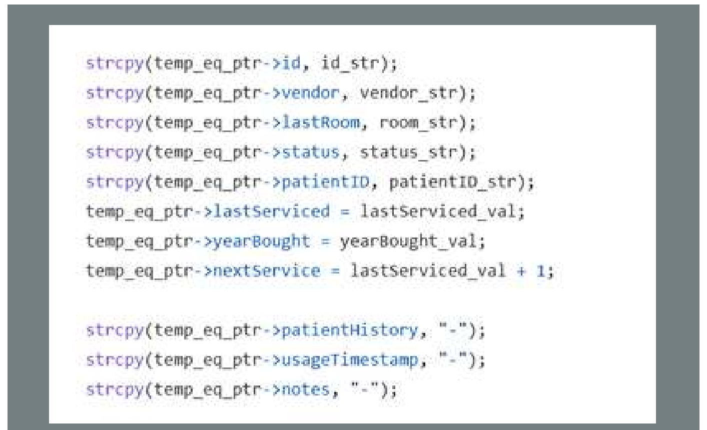
Kategori Function Alat

```c
int main() {
    Equipment **equipmentList = NULL;
    int equipmentCount = 0;
    int role;

    char useDatabase;
    printf("Do you want to use existing equipment database? (y/n): ");
    scanf(" %c", &useDatabase);
    getchar();

    if (useDatabase == 'y' || useDatabase == 'Y') {
        loadEquipmentDatabase(&equipmentList, &equipmentCount);
    } else {
        printf("Starting with empty equipment list. ID generation will start from defaults (10001, 20001, etc.).\n");
    }
```

Function Jika pakai database



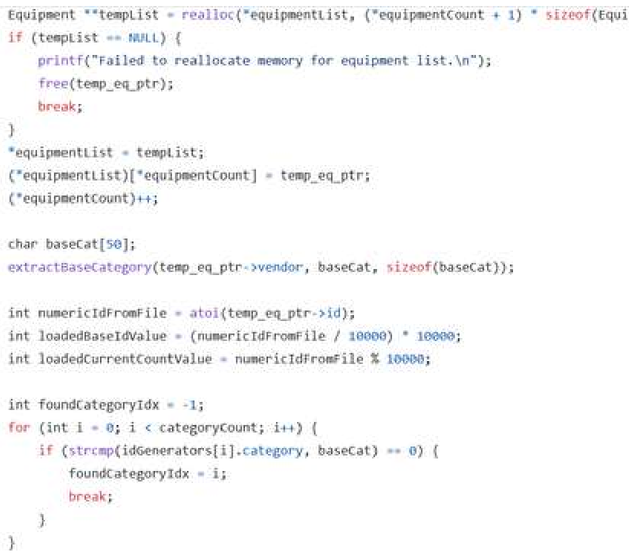
```c
void loadEquipmentDatabase(Equipment ***equipmentList, int *equipmentCount) {
    FILE *file = fopen("equipment.txt", "r");
    if (!file) {
        printf("Could not open equipment.txt. Starting with empty list.\n");
        return;
    }

    Equipment *temp_eq_ptr;
    char id_str[20], vendor_str[50], room_str[50], status_str[30], patientID_str[20];
    int lastServiced_val, yearBought_val;

    while (fscanf(file, "%[^,],%[^,],%[^,],%[^,],%[^,],%d,%d\n",
                  id_str, vendor_str, room_str, status_str, patientID_str, &lastServiced_val, &yearBought_val) == 7) {
        temp_eq_ptr = (Equipment *)malloc(sizeof(Equipment));
        if (!temp_eq_ptr) {
            printf("Memory allocation failed during DB load.\n");
            break;
        }
```



```c
strcpy(temp_eq_ptr->id, id_str);
strcpy(temp_eq_ptr->vendor, vendor_str);
strcpy(temp_eq_ptr->lastRoom, room_str);
strcpy(temp_eq_ptr->status, status_str);
strcpy(temp_eq_ptr->patientID, patientID_str);
temp_eq_ptr->lastServiced = lastServiced_val;
temp_eq_ptr->yearBought = yearBought_val;
temp_eq_ptr->nextService = lastServiced_val + 1;

strcpy(temp_eq_ptr->patientHistory, "-");
strcpy(temp_eq_ptr->usageTimestamp, "-");
strcpy(temp_eq_ptr->notes, "-");
```

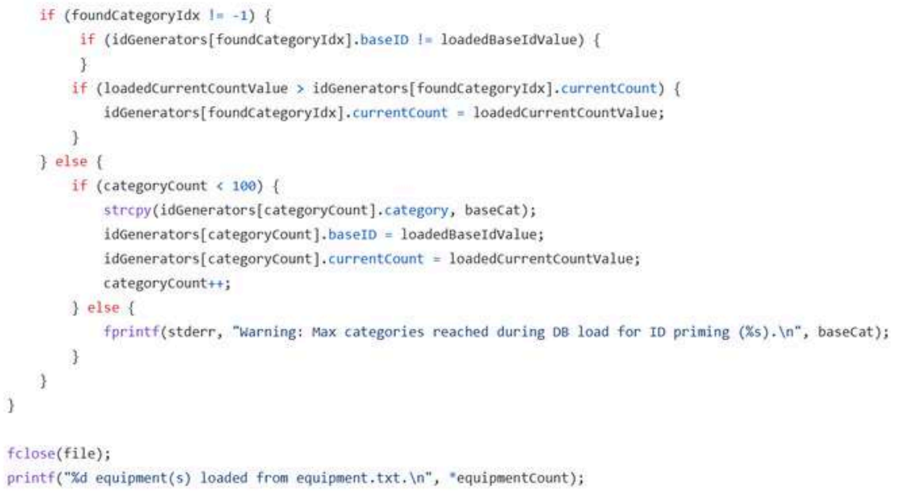
Pointer function di section lain

Main.c



```c
Equipment **tempList = realloc(*equipmentList, (*equipmentCount + 1) * sizeof(Equipment *));
if (tempList == NULL) {
    printf("Failed to reallocate memory for equipment list.\n");
    free(temp_eq_ptr);
    break;
}
*equipmentList = tempList;
(*equipmentList)[*equipmentCount] = temp_eq_ptr;
(*equipmentCount)++;

char baseCat[50];
extractBaseCategory(temp_eq_ptr->vendor, baseCat, sizeof(baseCat));

int numericIdFromFile = atoi(temp_eq_ptr->id);
int loadedBaseIdValue = (numericIdFromFile / 10000) * 10000;
int loadedCurrentCountValue = numericIdFromFile % 10000;

int foundCategoryIdx = -1;
for (int i = 0; i < categoryCount; i++) {
    if (strcmp(idGenerators[i].category, baseCat) == 0) {
        foundCategoryIdx = i;
        break;
    }
}
```

Untuk manggil Category alat berdasarkan ID



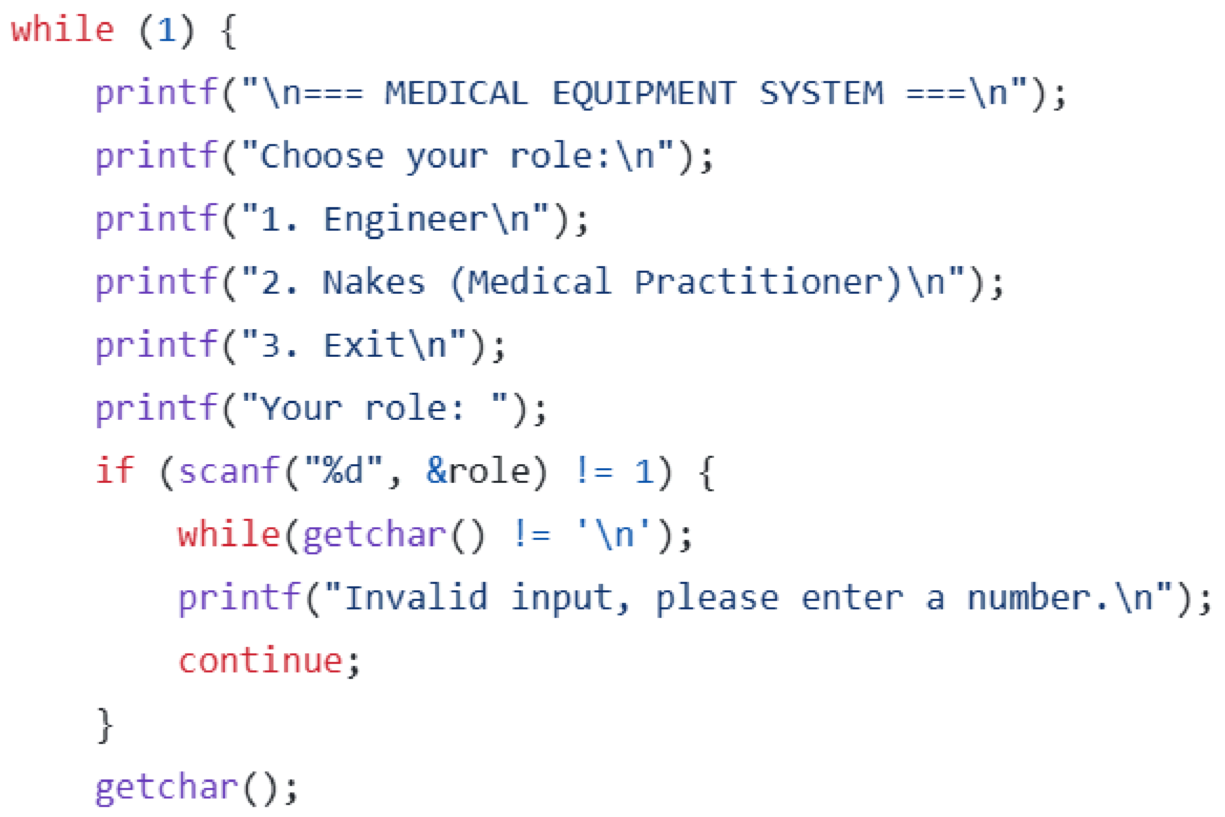
```c
    if (foundCategoryIdx != -1) {
        if (idGenerators[foundCategoryIdx].baseID != loadedBaseIdValue) {
        }
        if (loadedCurrentCountValue > idGenerators[foundCategoryIdx].currentCount) {
            idGenerators[foundCategoryIdx].currentCount = loadedCurrentCountValue;
        }
    } else {
        if (categoryCount < 100) {
            strcpy(idGenerators[categoryCount].category, baseCat);
            idGenerators[categoryCount].baseID = loadedBaseIdValue;
            idGenerators[categoryCount].currentCount = loadedCurrentCountValue;
            categoryCount++;
        } else {
            fprintf(stderr, "Warning: Max categories reached during DB load for ID priming (%s).\n", baseCat);
        }
    }
}

fclose(file);
printf("%d equipment(s) loaded from equipment.txt.\n", *equipmentCount);
```

MAIN FUNCTION

Main.c

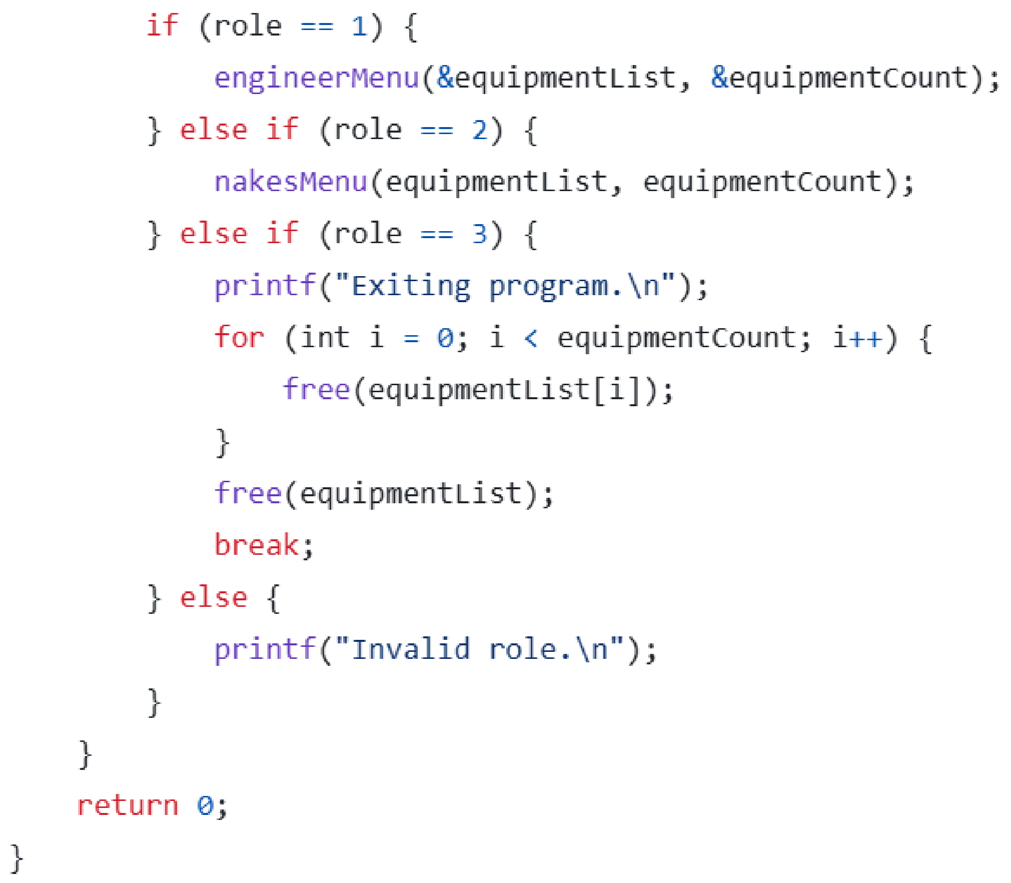
Main.c

Print Menu Utama



```c
while (1) {
    printf("\n=== MEDICAL EQUIPMENT SYSTEM ===\n");
    printf("Choose your role:\n");
    printf("1. Engineer\n");
    printf("2. Nakes (Medical Practitioner)\n");
    printf("3. Exit\n");
    printf("Your role: ");
    if (scanf("%d", &role) != 1) {
        while(getchar() != '\n');
        printf("Invalid input, please enter a number.\n");
        continue;
    }
    getchar();
```

Print menu utama

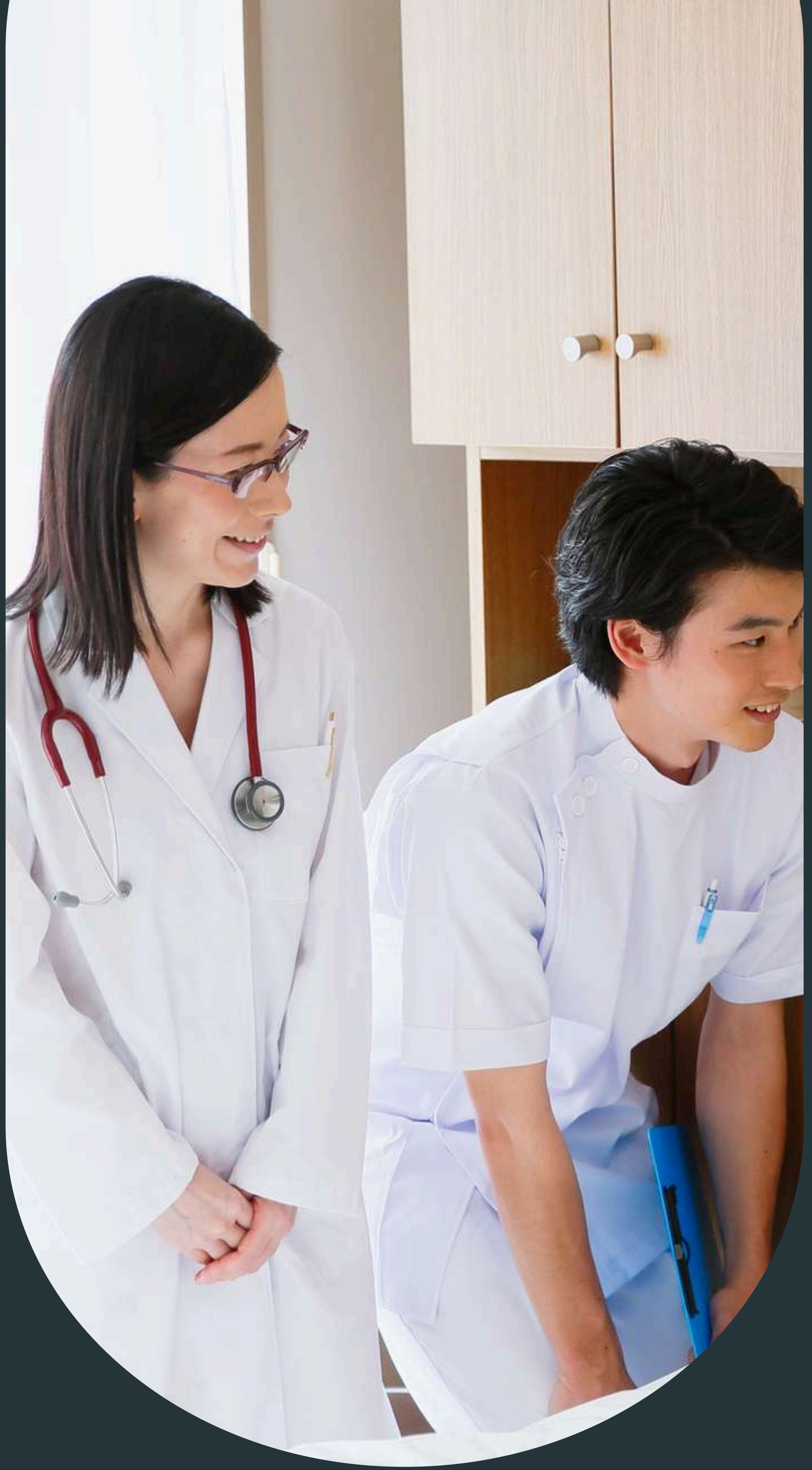


```c
        if (role == 1) {
            engineerMenu(&equipmentList, &equipmentCount);
        } else if (role == 2) {
            nakesMenu(equipmentList, equipmentCount);
        } else if (role == 3) {
            printf("Exiting program.\n");
            for (int i = 0; i < equipmentCount; i++) {
                free(equipmentList[i]);
            }
            free(equipmentList);
            break;
        } else {
            printf("Invalid role.\n");
        }
    }
    return 0;
}
```

Pemilihan Menu (Engineer, Nakes, Exit)

Conclusion

Program sistem manajemen alat kesehatan terintegrasi membantu rumah sakit dalam mencatat, memantau, dan melacak alat secara efisien. Sistem ini meningkatkan akurasi data, mendukung pengambilan keputusan, serta memudahkan kolaborasi antara tenaga kesehatan dan teknisi.



Building Trust and Awareness in the Community

Thank You