FITNITE: FITNESS, NUTRITION, AND INTAKE TRACKER ENGINE

KELOMPOK 10

Presented by :

Kaizan Rafi Rayhaqi (2306251544)

Nazwatu Sya'adah (2306169015)

Raudhah (2306169002)

codec.c > showExerciseHistory(const char *)

```
1   #include <stdio.h>
2   #include <string.h>
3   #
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27  void snowWeeklySummary(const char *email, int targetKalori);
28
29  int main() {
```

LATAR BELAKANG

- SULITNYA MELACAK ASUPAN KALORI DAN NUTRISI HARIAN SECARA AKURAT.
- KURANGNYA PEMAHAMAN TENTANG KEBUTUHAN GIZI INDIVIDU.
- KETERBATASAN AKSES TERHADAP DATA NUTRISI MAKANAN

Program FitNite hadir sebagai solusi untuk mengatasi berbagai tantangan di atas. Dengan mengintegrasikan fitur pelacakan kebugaran, nutrisi, dan asupan harian, FitNite memungkinkan pengguna untuk mencatat makanan yang dikonsumsi, memantau aktivitas fisik, serta mendapatkan gambaran detail mengenai asupan kalori dan makronutrien mereka.

UNIVERSITAS INDONESIA
FAKULTAS TEKNIK

TUJUAN UTAMA

Tujuan utama FitNite adalah menjadi sebuah platform digital yang lengkap dan mudah digunakan bagi siapa saja yang ingin mengelola kesehatan dan kebugaran mereka.

KEMAMPUAN FITNITE :

- Pelacakan Makanan: Mencatat detail makanan, dilengkapi database nutrisi, & hitung kalori/makronutrien otomatis.
- Pemantauan Aktivitas: Mencatat jenis & durasi aktivitas fisik, serta perkiraan kalori terbakar.
- Manajemen Profil: Pengaturan profil pribadi & penetapan tujuan kebugaran dengan rekomendasi kalori harian.

LIMITASI FITNITE

DATABASE MAKANAN & AKTIVITAS AWAL TERBATAS

PADA TAHAP AWAL, DATABASE MAKANAN & AKTIVITAS MUNGKIN BELUM SELENGKAP APLIKASI BESAR. BEBERAPA MAKANAN ATAU MEREK PERLU DIMASUKKAN MANUAL OLEH PENGGUNA.

ESTIMASI KALORI AKTIVITAS

PEMBAKARAN KALORI DARI AKTIVITAS FISIK DIDASARKAN PADA ESTIMASI UMUM, BELUM MEMPERHITUNGKAN FAKTOR INDIVIDUAL YANG SANGAT SPESIFIK.

BUKAN PENGGANTI PROFESIONAL MEDIS

FITNITE ADALAH ALAT BANTU PELACAK, BUKAN PENGGANTI SARAN DARI AHLI GIZI ATAU PROFESIONAL MEDIS.

SIAPA PENGGUNA FITNITE?

Pengguna utama dari program FitNite adalah individu yang peduli akan kesehatan dan kebugaran pribadinya.

Melacak dan mengelola asupan makanan harian: Baik untuk tujuan diet, pembentukan otot, atau sekadar menjaga pola makan sehat.

Memantau aktivitas fisik: Untuk memastikan mereka memenuhi target olahraga atau melihat progres kebugaran.

Mencapai tujuan kesehatan spesifik: Seperti menurunkan berat badan, menaikkan massa otot, atau menjaga berat badan ideal.

LIBRARY

- **main.c**

```c
#include <stdio.h>
#include "auth.h"
#include "utils.h"
```

- **utils.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include "utils.h"
```

- **exercise.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "exercise.h"
#include "utils.h"
```

- **auth.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "auth.h"
#include "dashboard.h"
```

- **dashboard.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "dashboard.h"
#include "exercise.h"
#include "meal.h"
#include "utils.h"
```

- **meal.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "meal.h"
#include "utils.h"
```

FLOWCHART (main.c)

Tujuan: Memungkinkan user mengakses sistem pelacak kesehatan dan kebugaran diri dengan akun pribadi


```c
int main() {
    int choice;

    // Membuat folder/direktori yang dibutuhkan jika belum ada
a   createDirectories();

    printf("=========================================\n");
    printf("           WELCOME TO FitNite            \n");
    printf("Your ultimate buddy for a healthier life!\n");
    printf("=========================================\n\n");

    printf("What would you like to do today?\n");
    printf("1. Create a new account\n");
    printf("2. Log in to your account\n\n");
    printf("Your choice (1/2): ");
    scanf("%d", &choice);
    getchar(); // Menghapus karakter newline dari buffer

    // Mengecek pilihan user
    switch(choice) {
b       case 1:
            createAccount();
            break;
c       case 2:
            login();
            break;
        default:
            printf("Oops! That's not a valid choice. Please try again.\n");
    }

    return 0;
}
```

FLOWCHART (auth.c)

```c
struct User {
    char email[50];
    char password[11];
};
```

Menyimpan data pengguna.

D

Tujuan: Menghitung jumlah kalori yang dibutuhkan seseorang per hari, berdasarkan

- **Jenis kelamin** (gender)
- **Umur**, **tinggi**, dan **berat badan**
- **Tingkat aktivitas fisik** (aktivitasLevel)
- **Tujuan pribadi** (misalnya ingin kurus, gemuk, tambah tinggi, atau membentuk otot)

a **Hitung BMR (Basal Metabolic Rate)**
Kalori dasar yang dibutuhkan tubuh saat istirahat, dihitung berbeda untuk pria dan wanita.

b **Tentukan Faktor Aktivitas**
Semakin aktif seseorang, semakin banyak kalori yang dibutuhkan.

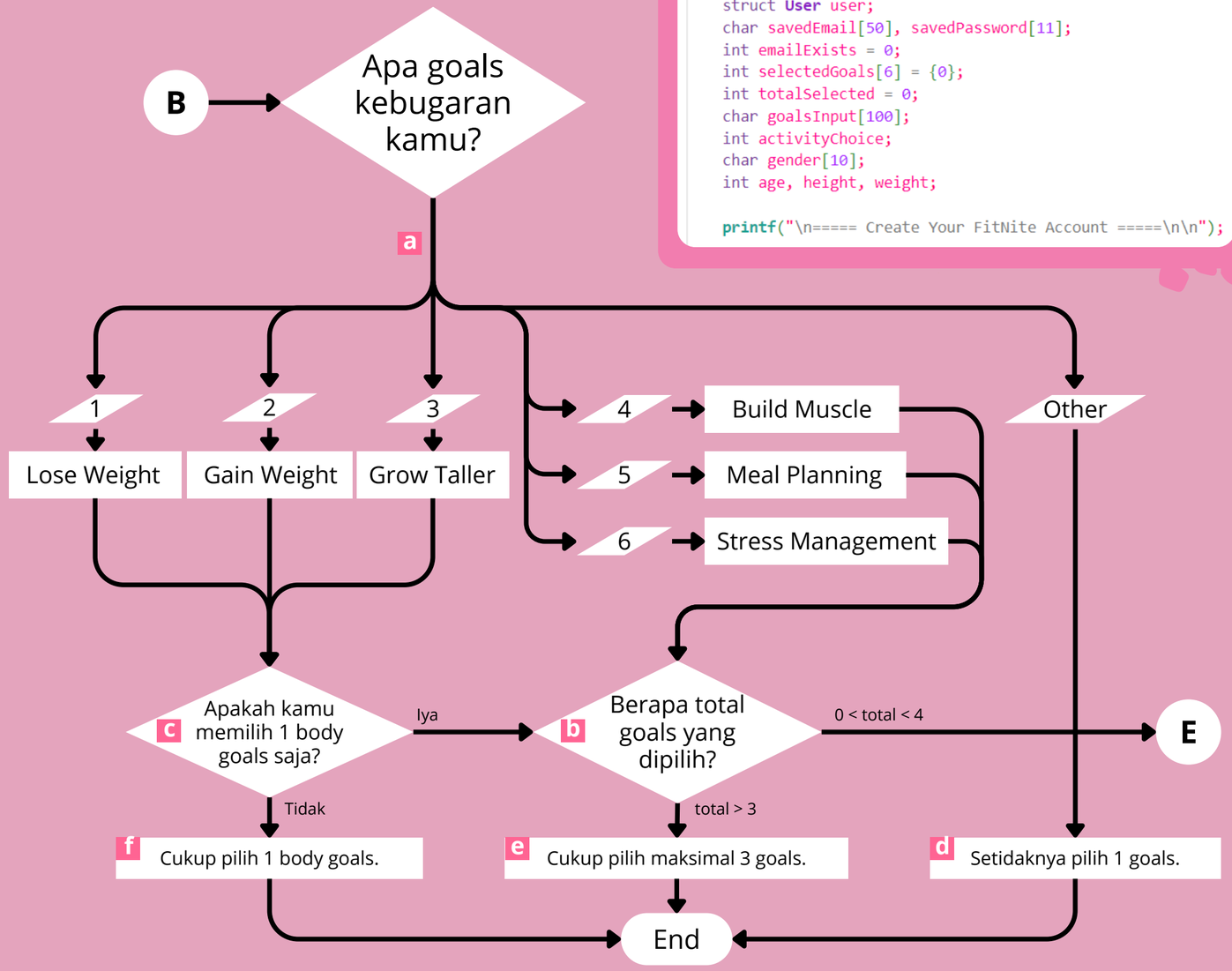
c **Sesuaikan Kalori Berdasarkan Tujuan**

```c
int hitungKaloriTarget(char *gender, int umur, int tinggi, int berat,
    int aktivitasLevel, int *goals) {
    double bmr, faktorAktivitas;

    // Rumus BMR untuk pria dan wanita berbeda
a   if (strcmp(gender, "Male") == 0) {
        bmr = 10 * berat + 6.25 * tinggi - 5 * umur + 5;
    } else {
        bmr = 10 * berat + 6.25 * tinggi - 5 * umur - 161;
    }

    // Menentukan faktor aktivitas berdasarkan input
b   switch (aktivitasLevel) {
        case 1: faktorAktivitas = 1.2; break;
        case 2: faktorAktivitas = 1.375; break;
        case 3: faktorAktivitas = 1.55; break;
        case 4: faktorAktivitas = 1.725; break;
        default: faktorAktivitas = 1.2;
    }

    double kebutuhanKalori = bmr * faktorAktivitas;

    // Menyesuaikan kalori berdasarkan goals
c   if (goals[0]) kebutuhanKalori -= 500;        // Lose weight
    else if (goals[1]) kebutuhanKalori += 500;   // Gain weight
    else if (goals[2]) kebutuhanKalori += 250;   // Grow taller
    else if (goals[3]) kebutuhanKalori += 300;   // Build muscle

    return (int)kebutuhanKalori;
}
```

FLOWCHART (auth.c)

Tujuan: Meminta input untuk membuat akun baru.

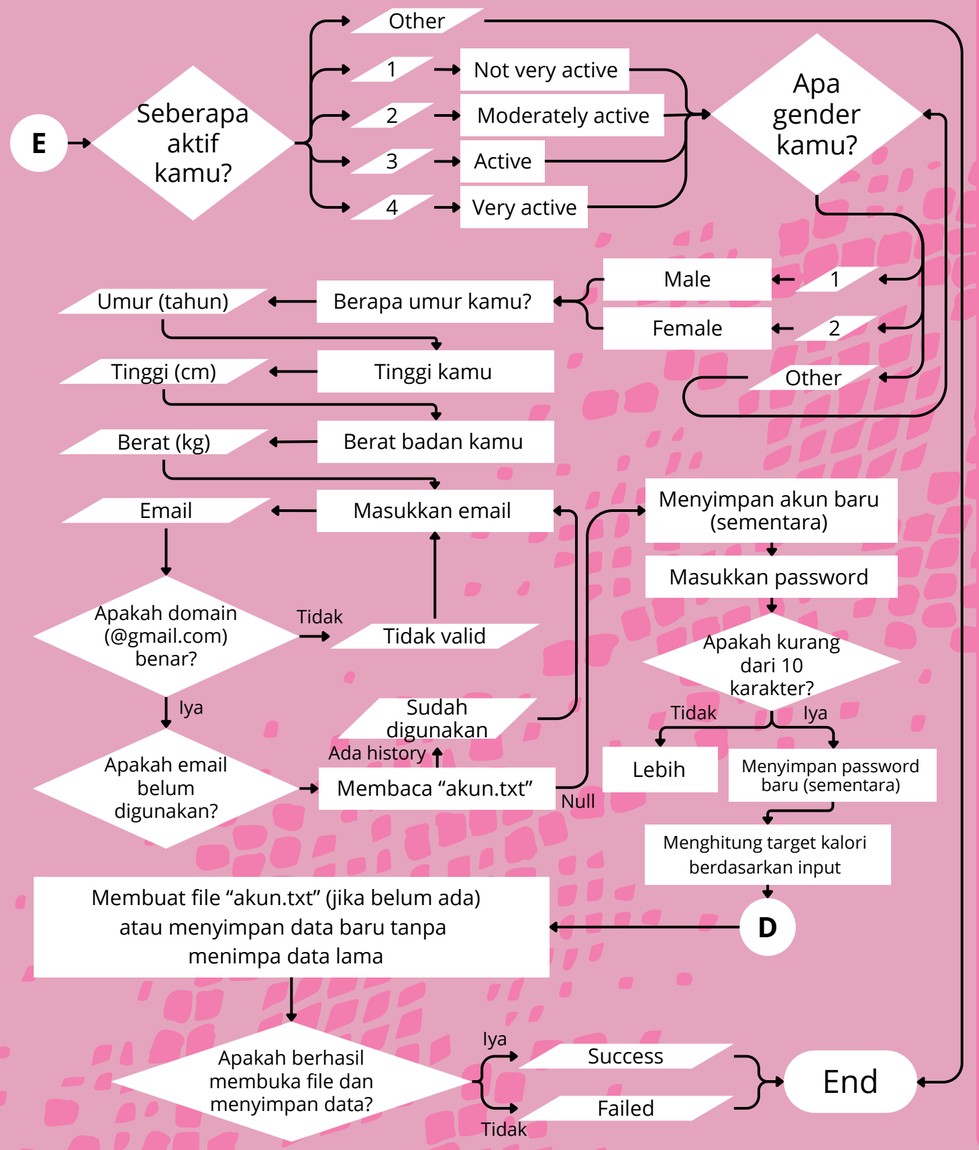
```c
void createAccount() {
    struct User user;
    char savedEmail[50], savedPassword[11];
    int emailExists = 0;
    int selectedGoals[6] = {0};
    int totalSelected = 0;
    char goalsInput[100];
    int activityChoice;
    char gender[10];
    int age, height, weight;

    printf("\n===== Create Your FitNite Account =====\n\n");
```




```c
// ===== Goals =====
const char *goalsList[] = {
    "Lose Weight",
    "Gain Weight",
    "Grow Taller",
    "Build Muscle",
    "Meal Planning",
    "Stress Management\n"
};

printf("What are your fitness goals? (Max 3 choices, e.g., 1 4 6):\n");
a for (int i = 0; i < 6; i++) {
    printf("%d. %s\n", i + 1, goalsList[i]);
}

printf("Your choice: ");
b fgets(goalsInput, sizeof(goalsInput), stdin);
// Parsing input goals
char *token = strtok(goalsInput, " ");
while (token != NULL && totalSelected < 6) {
    int index = atoi(token);
    if (index >= 1 && index <= 6) {
        if (!selectedGoals[index - 1]) {
            selectedGoals[index - 1] = 1;
            totalSelected++;
        }
    }
    token = strtok(NULL, " ");
}

// Validasi jumlah dan jenis goals
c int bodyGoals = selectedGoals[0] + selectedGoals[1] + selectedGoals[2];

d if (totalSelected == 0) {
    printf("You must select at least one goal.\n");
    return;
}
e if (totalSelected > 3) {
    printf("You can only choose up to 3 goals.\n");
    return;
}
f if (bodyGoals > 1) {
    printf("You can only select one body transformation goal");
    printf("(lose/gain weight/height).\n");
    return;
}
```

FLOWCHART (auth.c)

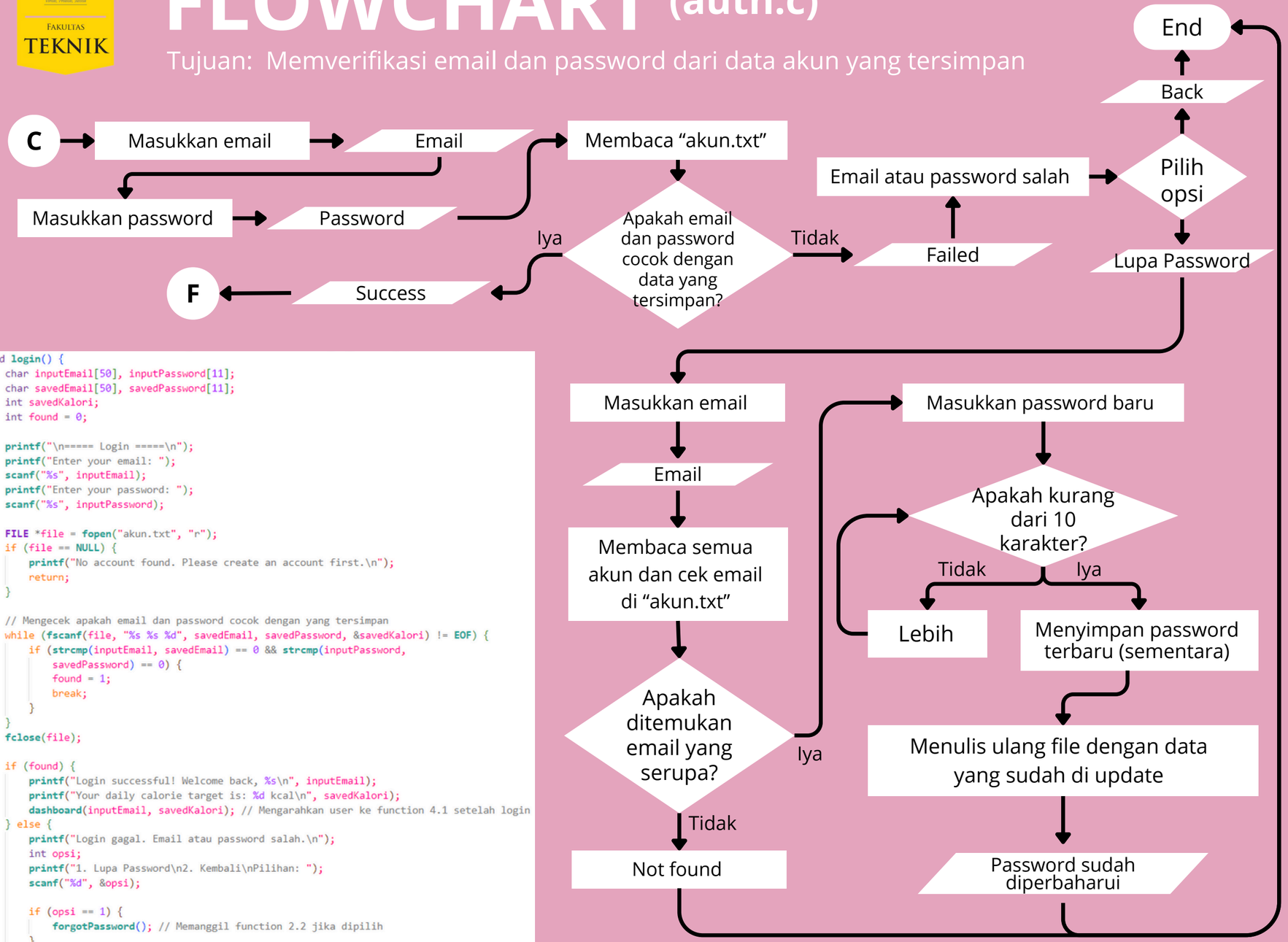
```c
// ===== Activity Level =====
printf("\nHow active are you?\n");
printf("1. Not very active\n2. Moderately active\n3. Active\n");
printf("4. Very active\n\n");
printf("Your choice (1/2/3/4): ");
scanf("%d", &activityChoice);
getchar(); // Menghapus karakter newline dari buffer

if (activityChoice < 1 || activityChoice > 4) {
    printf("Invalid selection. Please choose between 1-4.\n");
    return;
}

// ===== Data Diri =====
do {
    printf("\nWhats is your gender?\n");
    printf("1. Male\n2. Female\n\n");
    printf("Your choice (1/2): ");
    int genderChoice;
    scanf("%d", &genderChoice);
    getchar();

    if (genderChoice == 1) {
        strcpy(gender, "Male");
        break;
    } else if (genderChoice == 2) {
        strcpy(gender, "Female");
        break;
    } else {
        printf("Invalid input. Please enter 1 or 2.\n");
    }
} while(1);

printf("\nHow old are you? (years): ");
scanf("%d", &age);

printf("\nYour height (cm): ");
scanf("%d", &height);

printf("\nYour weight (kg): ");
scanf("%d", &weight);
getchar(); // Menghapus karakter newline dari buffer
```




```c
// ===== Email dan Password =====
FILE *file;

do {
    emailExists = 0;

    printf("\nEnter your email (@gmail.com): ");
    scanf("%s", user.email);

    // Memvalidasi domain email
    if (strstr(user.email, "@gmail.com") == NULL) {
        printf("Email must be a @gmail.com address.\n");
        continue;
    }

    // Mengecek apakah email sudah digunakan
    file = fopen("akun.txt", "r");
    if (file != NULL) {
        while (fscanf(file, "%s %s", savedEmail, savedPassword) != EOF) {
            if (strcmp(user.email, savedEmail) == 0) {
                emailExists = 1;
                printf("Oh noo! This email is already registered.");
                printf("Try another one.\n");
                break;
            }
        }
        fclose(file);
    }

} while (strstr(user.email, "@gmail.com") == NULL || emailExists);

do {
    printf("Set your password (max 10 characters): ");
    scanf("%s", user.password);
    if (strlen(user.password) > 10) {
        printf("Password is too long! Maximum 10 characters allowed.\n");
    }
} while (strlen(user.password) > 10);

// ===== Hitung target kalori =====
int targetKalori = hitungKaloriTarget(gender, age, height, weight,
    activityChoice, selectedGoals);
printf("\nYour daily calorie target is: %d kcal\n", targetKalori);

// Menyimpan data akun ke file
file = fopen("akun.txt", "a"); // Mode append agar tidak menimpa data lama
if (file != NULL) {
    fprintf(file, "%s %s %d\n", user.email, user.password, targetKalori);
    fclose(file);
    printf("Your account has been successfully created and saved!\n");
} else {
    printf("Failed to save account. Please try again later.\n");
}
}
```

FLOWCHART (auth.c)

Tujuan: Memverifikasi email dan password dari data akun yang tersimpan

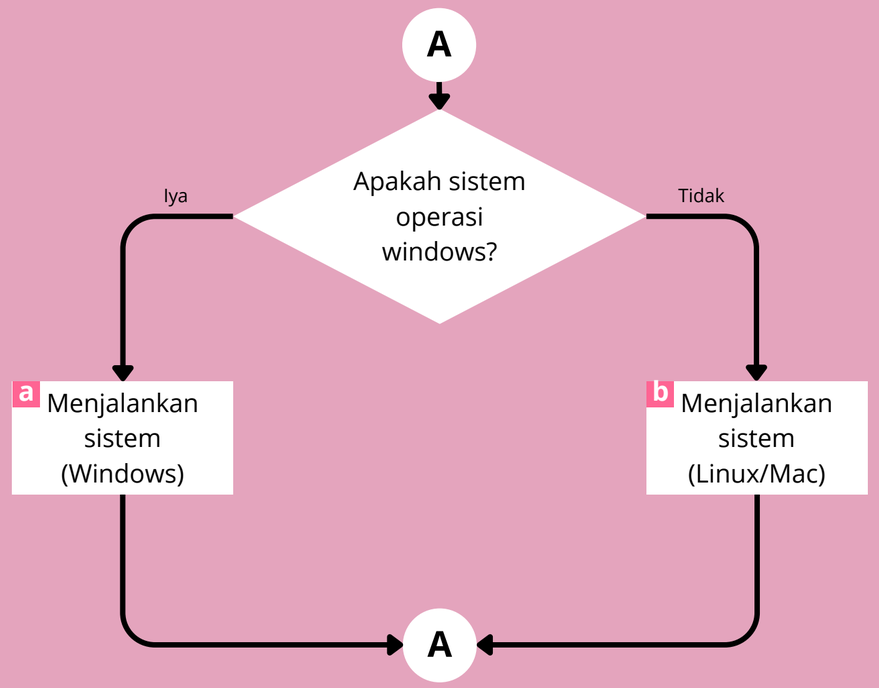



```c
void login() {
    char inputEmail[50], inputPassword[11];
    char savedEmail[50], savedPassword[11];
    int savedKalori;
    int found = 0;

    printf("\n===== Login =====\n");
    printf("Enter your email: ");
    scanf("%s", inputEmail);
    printf("Enter your password: ");
    scanf("%s", inputPassword);

    FILE *file = fopen("akun.txt", "r");
    if (file == NULL) {
        printf("No account found. Please create an account first.\n");
        return;
    }

    // Mengecek apakah email dan password cocok dengan yang tersimpan
    while (fscanf(file, "%s %s %d", savedEmail, savedPassword, &savedKalori) != EOF) {
        if (strcmp(inputEmail, savedEmail) == 0 && strcmp(inputPassword,
            savedPassword) == 0) {
            found = 1;
            break;
        }
    }
    fclose(file);

    if (found) {
        printf("Login successful! Welcome back, %s\n", inputEmail);
        printf("Your daily calorie target is: %d kcal\n", savedKalori);
        dashboard(inputEmail, savedKalori); // Mengarahkan user ke function 4.1 setelah login
    } else {
        printf("Login gagal. Email atau password salah.\n");
        int opsi;
        printf("1. Lupa Password\n2. Kembali\nPilihan: ");
        scanf("%d", &opsi);

        if (opsi == 1) {
            forgotPassword(); // Memanggil function 2.2 jika dipilih
        }
    }
}
```

```c
void forgotPassword() {
    char email[50], newPassword[11];
    char savedEmail[50], savedPassword[11];
    int found = 0;
    int count = 0;

    // Menyimpan data user sementara ke array struct
    struct {
        char email[50];
        char password[11];
        int kalori;
    } users[100];

    printf("\n===== Forgot Password =====\n");
    printf("Enter your registered email: ");
    scanf("%s", email);

    FILE *file = fopen("akun.txt", "r");
    if (file == NULL) {
        printf("No account data found.\n");
        return;
    }

    // Membaca semua akun dan cek apakah email ditemukan
    while (fscanf(file, "%s %s %d", users[count].email, users[count].password,
        &users[count].kalori) != EOF) {
        if (strcmp(email, users[count].email) == 0) {
            found = 1;
        }
        count++;
    }
    fclose(file);

    if (found) {
        // Minta password baru dan validasi panjangnya
        do {
            printf("Enter a new password (max 10 characters): ");
            scanf("%s", newPassword);
            if (strlen(newPassword) > 10) {
                printf("Password too long! Please try again.\n");
            }
        } while (strlen(newPassword) > 10);

        // Update password di array
        for (int i = 0; i < count; i++) {
            if (strcmp(email, users[i].email) == 0) {
                strcpy(users[i].password, newPassword);
            }
        }

        // Tulis ulang file dengan data yang telah diperbarui
        file = fopen("akun.txt", "w");
        for (int i = 0; i < count; i++) {
            fprintf(file, "%s %s %d\n", users[i].email, users[i].password, users[i].kalori);
        }
        fclose(file);

        printf("Password successfully updated!\n");
    } else {
        printf("Email not found. Please create an account first.\n");
    }
}
```

FLOWCHART (utils.c)

Tujuan: Menyiapkan folder yang dibutuhkan program sesuai sistem operasi (Windows atau Linux/Mac)

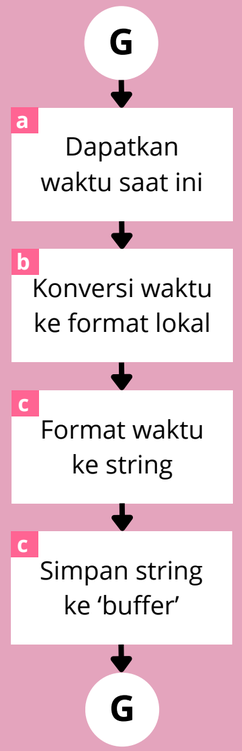



```
void createDirectories() {
#ifdef _WIN32
    system("mkdir logs 2>nul");
    system("mkdir dataset 2>nul");
#else
    system("mkdir -p logs");
    system("mkdir -p dataset");
#endif
}
```

FLOWCHART (utils.c)

Tujuan: Mendapatkan tanggal hari ini dalam format "YYYY-MM-DD" dan mendapatkan jam saat ini dalam format "HH:MM:SS"

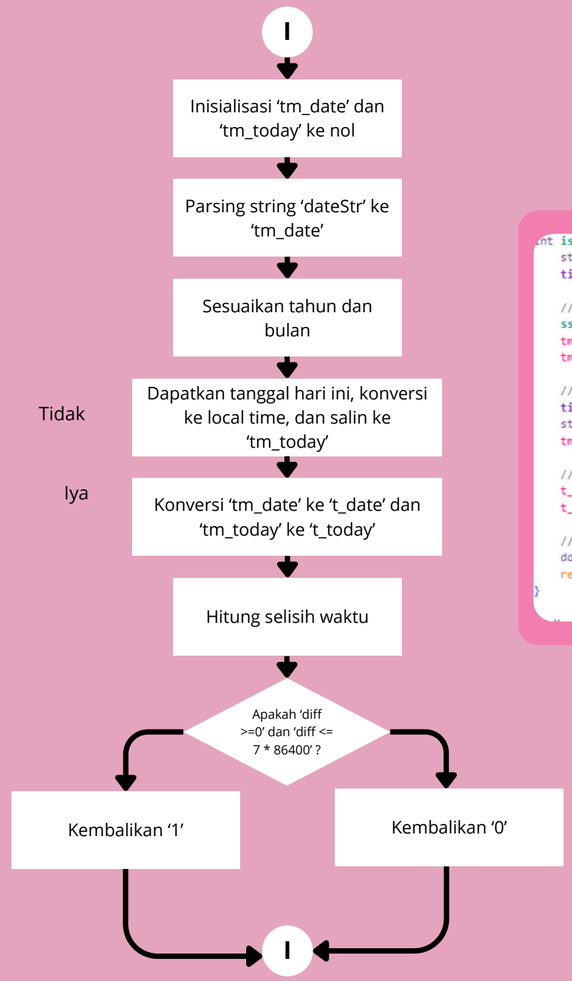
```c
void getToday(char *buffer, int size) {
  a time_t t = time(NULL);
  b struct tm *tm = localtime(&t);
  c strftime(buffer, size, "%Y-%m-%d", tm);
}
```




```c
void getCurrentTime(char *buffer, int size) {
  d time_t t = time(NULL);
  e struct tm *tm = localtime(&t);
  f strftime(buffer, size, "%H:%M:%S", tm);
}
```

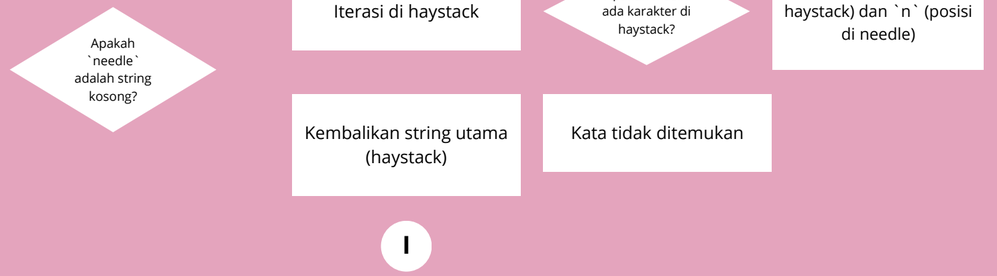
FLOWCHART (utils.c)

Tujuan: Mengecek apakah sebuah tanggal termasuk dalam 7 hari terakhir




```c
int isWithin7Days(const char *dateStr) {
    struct tm tm_date = {0}, tm_today = {0};
    time_t t_date, t_today;

    // Mengubah string tanggal input ke struct tm
    sscanf(dateStr, "%4d-%2d-%2d", &tm_date.tm_year, &tm_date.tm_mon, &tm_date.tm_mday);
    tm_date.tm_year -= 1900; // Format tahun disesuaikan (sejak 1900)
    tm_date.tm_mon -= 1;     // Format bulan disesuaikan (0-11)

    // Menyimpan tanggal hari ini
    time_t t = time(NULL);
    struct tm *lt = localtime(&t);
    tm_today = *lt;

    // Menghitung selisih waktu
    t_date = mktime(&tm_date);
    t_today = mktime(&tm_today);

    // Mengecek apakah selisih kurang dari atau sama dengan 7 hari
    double diff = difftime(t_today, t_date);
    return (diff >= 0 && diff <= 7 * 86400);
}
```

```c
char *strcasestr_custom(const char *haystack, const char *needle) {
    if (!*needle) return (char *) haystack;
    for (; *haystack; haystack++) {
        const char *h = haystack;
        const char *n = needle;

        // Selama karakter masih sama dan belum habis, membandingkan tetap dila
        while (*h && *n && tolower((unsigned char)*h) == tolower((unsigned char
            h++;
            n++;
        }

        // Jika seluruh substring sudah selesai dicocokkan, return posisi
        if (!*n) return (char *) haystack;
    }
    return NULL;
}
```



FLOWCHART (dashboard.c)

Tujuan: Menampilkan ringkasan kalori harian pengguna serta opsi untuk log aktivitas atau melihat riwayat berdasarkan email

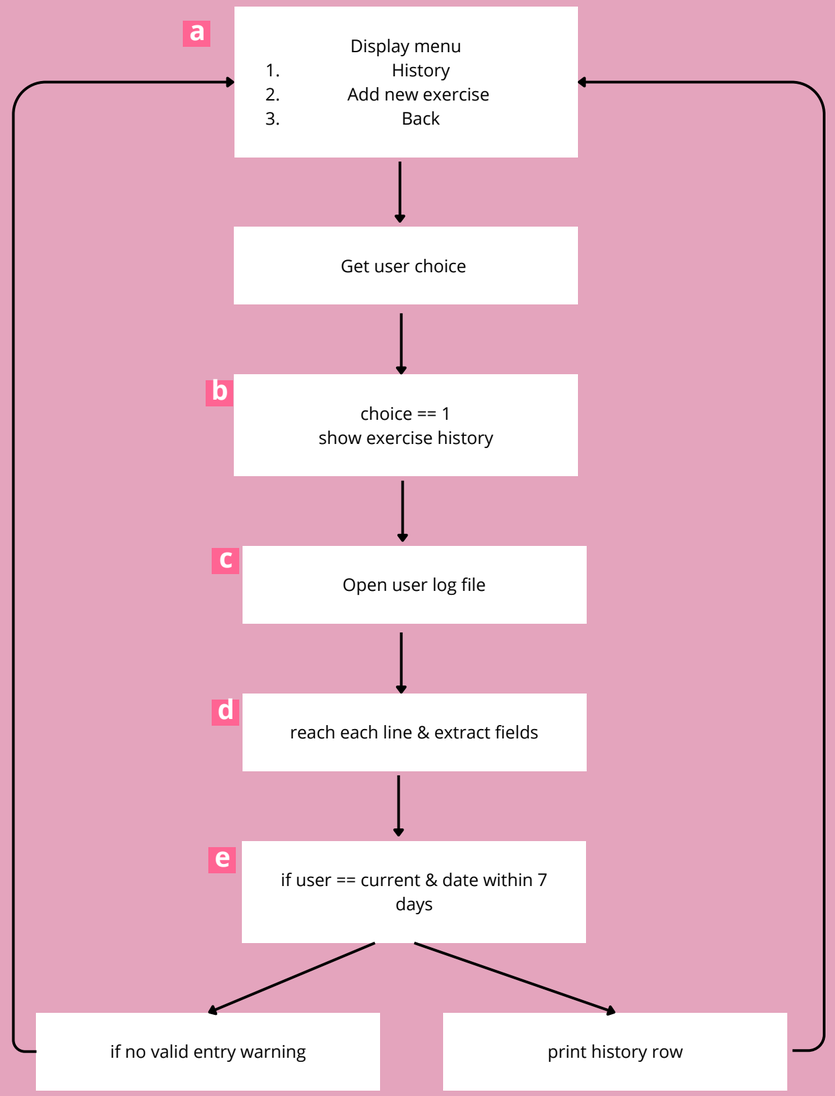
```
#define MAX_LINE 256
```

Membaca maksimum panjang baris di file .txt

```c
void dashboard(const char *email, int targetKalori) {
    int pilihan;
    char today[20];
    getToday(today, sizeof(today));

    // Reset nilai untuk dihitung ulang setiap loop
    int kaloriMakanHariIni = 0;
    int jumlahMakananHariIni = 0;
    int kaloriExerciseHariIni = 0;
    int jumlahExerciseHariIni = 0;
    char user[50], tanggal[20], waktu[20], mealType[20], food[50];
    int kalori;

    // ===== Hitung Total Kalori Makan Hari Ini Dari File Log Makanan =====
    char line[MAX_LINE];
    char type[20], desc[100];
    int duration, caloriesBurned, sets, reps;

    while (1) {
        FILE *fmeal = fopen("logs/calories_logs.txt", "r");
        if (fmeal) {
            while (fgets(line, sizeof(line), fmeal)) {

                // Format baris: user tanggal waktu mealType food kalori
                if (sscanf(line, "%s %s %s %s %s %d", user, tanggal, waktu,
                    mealType, food, &kalori) == 6) {
                    // Jika log milik user yang sedang login dan untuk hari ini
                    if (strcmp(user, email) == 0 && strcmp(tanggal, today) == 0) {
                        kaloriMakanHariIni += kalori;
                        jumlahMakananHariIni++;
                    }
                }
            }
            fclose(fmeal);
        }

        // ===== Hitung Total Kalori Latihan Hari Ini Dari File Log Exercise =====
        FILE *fex = fopen("logs/exercise_logs.txt", "r");
        if (fex) {
            while (fgets(line, sizeof(line), fex)) {
                // Format baris: user tanggal waktu type desc duration sets reps caloriesBurne
                if (sscanf(line, "%s %s %s %s %s %d %d %d %d",
                           user, tanggal, waktu, type, desc, &duration, &sets,
                           &reps, &caloriesBurned) >= 6) {
                    // Jika log milik user dan untuk hari ini
                    if (strcmp(user, email) == 0 && strcmp(tanggal, today) == 0) {
                        kaloriExerciseHariIni += caloriesBurned;
                        jumlahExerciseHariIni++;
                    }
                }
            }
            fclose(fex);
        }
```

```c
        // ===== Hitung Total Kalori Latihan Hari Ini Dari File Log Exercise =====
        FILE *fex = fopen("logs/exercise_logs.txt", "r");
        if (fex) {
            while (fgets(line, sizeof(line), fex)) {
                // Format baris: user tanggal waktu type desc duration sets reps caloriesBurne
                if (sscanf(line, "%s %s %s %s %s %d %d %d %d",
                           user, tanggal, waktu, type, desc, &duration, &sets,
                           &reps, &caloriesBurned) >= 6) {
                    // Jika log milik user dan untuk hari ini
                    if (strcmp(user, email) == 0 && strcmp(tanggal, today) == 0) {
                        kaloriExerciseHariIni += caloriesBurned;
                        jumlahExerciseHariIni++;
                    }
                }
            }
            fclose(fex);
        }

        // Eksekusi Menu
        if (pilihan == 1) {
            exerciseTrackMenu(email);
        } else if (pilihan == 2) {
            dailyMealMenu(email);
        } else if (pilihan == 3) {
            showWeeklySummary(email, targetKalori);
        } else if (pilihan == 4) {
            printf("Successfully logged out.\n");
            break;
        } else {
            printf("Invalid choice.\n");
        }
    }
}
```

FLOWCHART (exercise.c)



a
```
    // Menampilkan menu fitur yang ada di daily exercise
    while (1) {
        printf("\n===== DAILY EXERCISE =====\n");
        printf("1. History\n");
        printf("2. Add New Exercise\n");
        printf("3. Back\n");
        printf("Your choice: ");
        scanf("%d", &pilih);
        getchar();
```

b
```
        // Mengksekusi fitur daily exercise
        if (pilih == 1) {
            showExerciseHistory(email);
        } else if (pilih == 2) {
            addNewExercise(email);
        } else if (pilih == 3) {
            break;
        } else {
            printf("Invalid choice.\n");
        }
    }
}
```

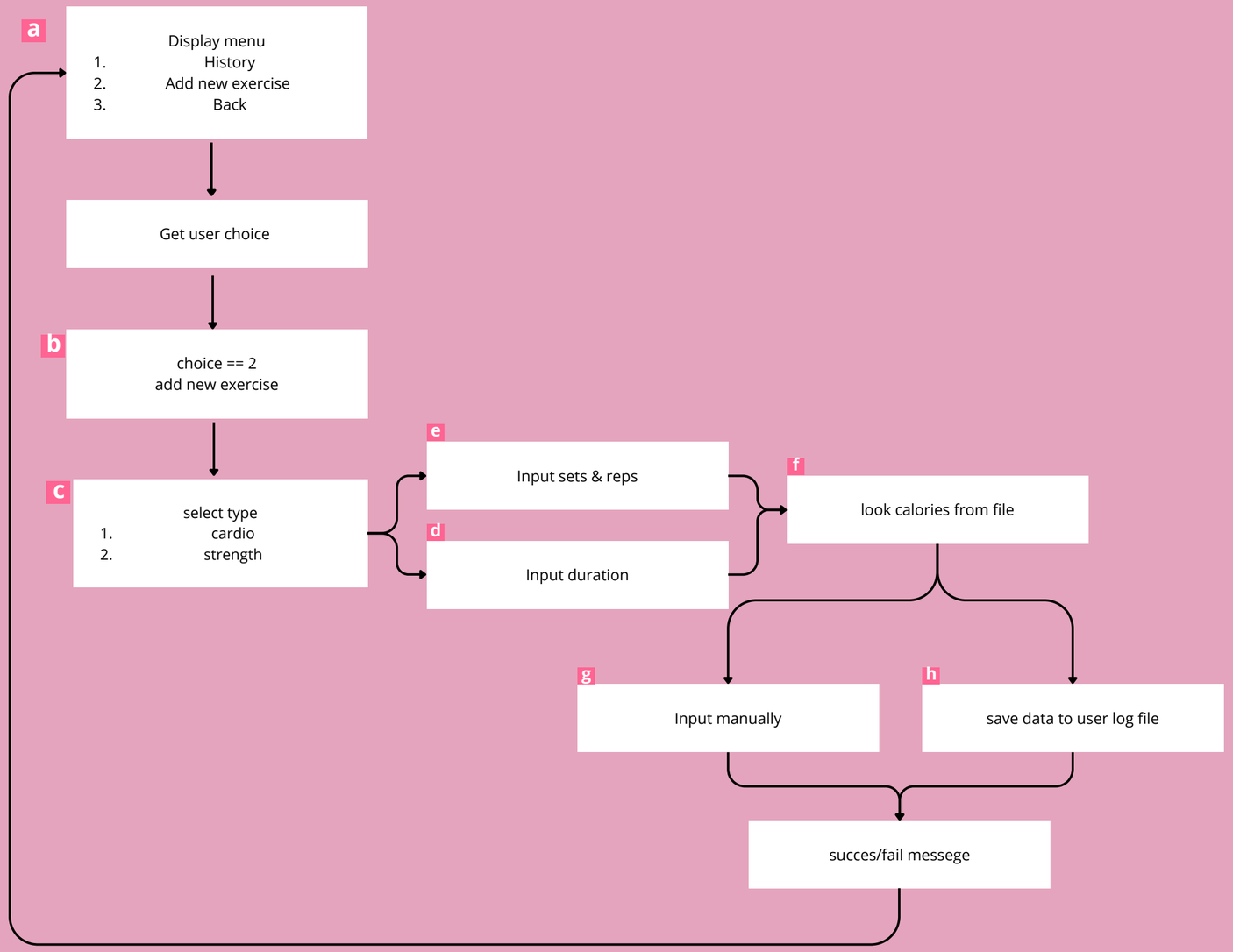
c
```
// ----------------------------------------------------------------
//                      5.2 Riwayat Exercise
// ----------------------------------------------------------------
// Tujuan: Menampilkan riwayat exercise yang dilakukan user selama 7 hari terakhir
void showExerciseHistory(const char *email) {
    // Membuat nama file log berdasarkan email user
    char filename[100];
    sprintf(filename, "logs/%s_exercise_logs.txt", email);
    FILE *fp = fopen(filename, "r");
    if (fp == NULL) {
        printf("You haven't logged any exercises yet.\n");
        return;
    }

    char line[MAX_LINE];
    int found = 0; // Variabel untuk mengecek apakah data valid

    // Header tampilan history
    printf("\n--- 7-Day Exercise History ---\n");
    printf("Date        Time      Category    Description
        Duration/Set-Reps    Calories Burned\n");
```

d
```
    // Membaca file baris demi baris lalu menyimpan datanya sementara
    while (fgets(line, sizeof(line), fp)) {
        char user[50], tanggal[20], waktu[20], type[20];
        char desc[100] = "";
        int duration = 0, sets = 0, reps = 0, calories = 0;

        // Mengambil 4 elemen awal dari baris
        if (sscanf(line, "%s %s %s %s", user, tanggal, waktu,
            type) != 4)
            continue;

        // Mengarahkan pointer ke bagian deskripsi dan angka
            setelah 4 elemen awal
        char *ptr = line;
        for (int i = 0; i < 4; i++) {
            ptr = strchr(ptr, ' ');
            if (ptr) ptr++;
            else break;
        }

        if (!ptr) continue;

        // Parsing deskripsi + angka: durasi, sets, reps,
            kalori
        char tempDesc[100];
        int matched = sscanf(ptr, "%[^0-9] %d %d %d %d",
            tempDesc, &duration, &sets, &reps, &calories);
        if (matched >= 2) {
            strcpy(desc, tempDesc);
            desc[strcspn(desc, "\n")] = '\0';
            while (isspace(desc[strlen(desc) - 1])) desc[strlen
                (desc) - 1] = '\0';
```

e
```
            // Filtering data user saat ini dan menampilkan
                data selama 7 hari terakhir
            if (strcmp(user, email) == 0 && isWithin7Days
                (tanggal)) {
                found = 1;
                if (strcmp(type, "cardio") == 0) {
                    printf("%-11s %-9s %-11s %-15s %2d menit
                        %15d kkal\n",
                            tanggal, waktu, "Cardio", desc,
                                duration, calories);
                } else {
                    printf("%-11s %-9s %-11s %-15s %d sets x %d
                        reps %5d kkal\n",
                            tanggal, waktu, "Strength", desc,
                                sets, reps, calories);
                }
            }
        }
    }

    fclose(fp);
    if (!found) {
        printf("You haven't logged any exercise in the last 7
            days.\n");
    }
}
```

FLOWCHART (exercise.c)

Tujuan: ....



a
```
// Menampilkan menu fitur yang ada di daily exercise
while (1) {
    printf("\n===== DAILY EXERCISE =====\n");
    printf("1. History\n");
    printf("2. Add New Exercise\n");
    printf("3. Back\n");
    printf("Your choice: ");
    scanf("%d", &pilih);
    getchar();
```

b
```
    // Mengksekusi fitur daily exercise
    if (pilih == 1) {
        showExerciseHistory(email);
    } else if (pilih == 2) {
        addNewExercise(email);
    } else if (pilih == 3) {
        break;
    } else {
        printf("Invalid choice.\n");
    }
}
```

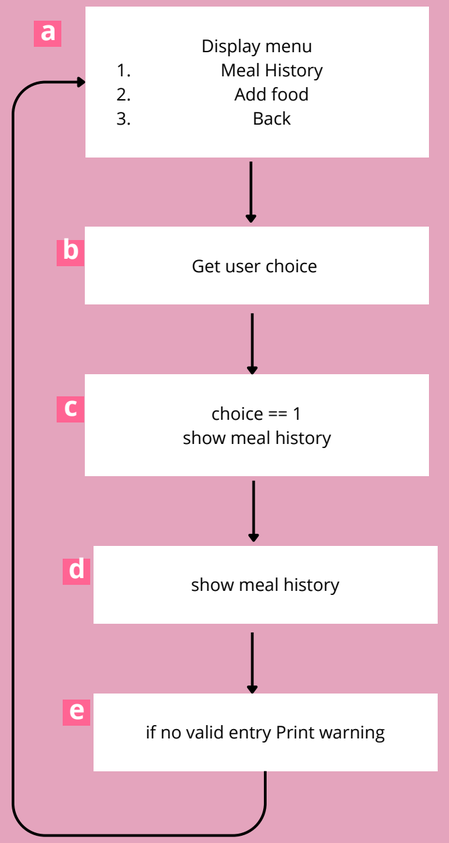
```
//                    5.3 Tambah Exercise
// ------------------------------------------------------------
    ----
// Tujuan: Menambahkan data exercise baru dari user lalu
    menyimpannya ke file log pribadi user
void addNewExercise(const char *email) {
```

c
```
    int choice;
    printf("\n=== Add New Exercise ===\n");
    printf("1. Cardiovascular\n");
    printf("2. Strength\n");
    printf("Your choice: ");
    scanf("%d", &choice);

    char description[100];
    int kalori = 0, durasi = 0, beratkg = 0;
    int sets = 0, reps = 0;
    getchar(); // flush newline

    // Input yang perlu diberikan oleh user
    printf("Activity description: ");
    fgets(description, sizeof(description), stdin);
    description[strcspn(description, "\n")] = '\0';

    printf("Enter your current weight (kg): ");
    scanf("%d", &beratkg);
    int beratlb = beratkg * 2.20462;
    printf("Your body weight (lbs): %d lb\n", beratlb);

    // Input yang perlu diberikan oleh user berdasarkan
        kategori exercise
    if (choice == 1) {
        // Cardiovascular
        printf("Duration (minute): ");
        scanf("%d", &durasi);
    } else if (choice == 2) {
        // Strength
        printf("Number of sets: ");
        scanf("%d", &sets);
        printf("Reps/set: ");
        scanf("%d", &reps);
        durasi = sets; // Asumsi 1 set = ±1 menit
    } else {
        printf("Invalid choice.\n");
        return;
    }
```

d
```
    // Mencari data kalori exercise dari dataset
    kalori = lookupExerciseCalories(description, durasi,
        beratlb);
    if (kalori == -1) {
        printf("We can't found this activity in our dataset.
            Enter calories manually: ");
        scanf("%d", &kalori);
    } else {
        printf("Automatically calculated calories: %d kkal\n",
            kalori);
    }
```

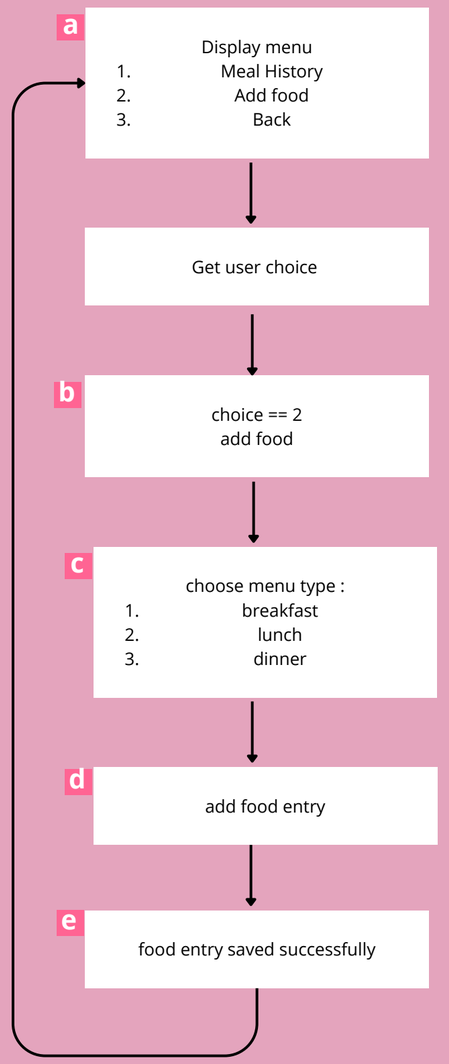
e
```
    // Mengatur tanggal dan waktu sekarang
    char tanggal[20], waktu[20];
    getToday(tanggal, sizeof(tanggal));
    getCurrentTime(waktu, sizeof(waktu));

    // Menyimpan data ke file log user
    char filename[100];
    sprintf(filename, "logs/%s_exercise_logs.txt", email);
    FILE *fp = fopen(filename, "a");
    if (fp != NULL) {
        if (choice == 1) {
            // Cardio
            fprintf(fp, "%s %s %s cardio %s %d 0 0 %d\n", email
                , tanggal, waktu, description, durasi, kalori);
        } else {
            // Strength
            fprintf(fp, "%s %s %s strength %s %d %d %d %d\n",
                email, tanggal, waktu, description, durasi,
                sets, reps, kalori);
        }
        fclose(fp);
        printf("Your exercise has been saved.\n");
    } else {
        printf("Error. Could not save log.\n");
    }
}
    ----
//                    5.4 HITUNG KALORI OLAHRAGA
// ------------------------------------------------------------
    ----
// Tujuan: Mencari jumlah kalori aktivitas olahraga berdasarkan
    berat & durasi dari dataset "exercise.txt"
int lookupExerciseCalories(const char *description, int
    durasiMenit, int beratBadan) {
    FILE *fp = fopen("exercise.txt", "r");
    if (fp == NULL) {
        perror("Oops! We couldn't open the exercise data.");
        return -1;
    }

    char line[512];
    char aktivitas[150];
    int kal130, kal155, kal180, kal205;
    int perJamKalori;

    // Melewati baris header
    fgets(line, sizeof(line), fp);

    // Membaca file baris per baris
    while (fgets(line, sizeof(line), fp)) {
        line[strcspn(line, "\n")] = '\0';

    // Membaca file baris per baris
    while (fgets(line, sizeof(line), fp)) {
        line[strcspn(line, "\n")] = '\0';

        // Mengambil data dari tiap kolom
        if (sscanf(line, "%[^\t]\t%d\t%d\t%d\t%d", aktivitas,
            &kal130, &kal155, &kal180, &kal205) == 5) {
            // Jika deskripsi ditemukan dalam aktivitas (tanpa
                case sensitive)
            if (strcasestr_custom(aktivitas, description) !=
                NULL) {
                fclose(fp);

                // Menentukan kalori per jam sesuai berat badan

                if (beratBadan <= 140) perJamKalori = kal130;
                else if (beratBadan <= 170) perJamKalori =
                    kal155;
                else if (beratBadan <= 190) perJamKalori =
                    kal180;
                else perJamKalori = kal205;

                // Menghitung total kalori
                return (perJamKalori * durasiMenit) / 60;
            }
        }
    }
    fclose(fp);
    return -1;
```

FLOWCHART (exercise.c)




```
//                          6.1 Menu Daily Meal
// ------------------------------------------------------------
    ----
// Tujuan: Menampilkan menu kepada user sehigga fitur daily
    meal dapat diakses
void dailyMealMenu(const char *email) {
    int pilih;

    // Menampilkan menu fitur yang ada di daily meal
    while (1) {
        printf("\n===== DAILY MEAL =====\n");
        printf("1. Meal History\n");
        printf("2. Add Food\n");
        printf("3. Back\n");
        printf("Your choice: ");
        scanf("%d", &pilih);
        getchar();

        // Mengeksekusi fitur daily meal
        if (pilih == 1) {
            showMealHistory(email);
        } else if (pilih == 2) {
            printf("\nPick your meal type:\n");
            printf("1. Breakfast\n2. Lunch\n3. Dinner\nYour
                choice: ");
            int mealTypeChoice;
            scanf("%d", &mealTypeChoice);
            getchar();
    ----
//                  6.2 Riwayat Semua MealType
// ------------------------------------------------------------
    ----
// Menampilkan semua meal history untuk semua mealType
void showMealHistory(const char *email) {
    FILE *f = fopen("logs/calories_logs.txt", "r");
    if (!f) {
        printf("You haven't log any meals yet.\n");
        return;
    }

    char line[MAX_LINE];
    int found = 0; // Variabel untuk validasi

    printf("\n--- 7-Day Meal History ---\n");
    printf("Date        Time      Type       Meal
                Calories\n");

    // loop membaca tiap baris dalam file log
    while (fgets(line, sizeof(line), f)) {
        char user[50], tanggal[20], waktu[20], mealType[20];
        char makanan[150];
        int kalori;

        // Mengambil bagian awal (4 elemen)
        int matched = sscanf(line, "%s %s %s %s", user, tanggal
            , waktu, mealType);
        if (matched != 4) continue;

        // Mengambil string setelah mealType
        char *afterMealType = strstr(line, mealType);
        if (!afterMealType) continue;
        afterMealType += strlen(mealType); // skip mealType

        // Membuang newline
        afterMealType[strcspn(afterMealType, "\n")] = '\0';

        // Mencari spasi terakhir dalam bagian makanan dan
            kalori yang diasumsikan diikuti kalori
        char *lastSpace = strrchr(afterMealType, ' ');
        if (!lastSpace) continue;
        kalori = atoi(lastSpace + 1);
        *lastSpace = '\0';
        strcpy(makanan, afterMealType);

        // Filtering data user saat ini dan menampilkannya
        if (strcmp(user, email) == 0 && isWithin7Days(tanggal))
            {
            found = 1;
            printf("%-11s %-9s %-9s %-27s %d kkal\n",
                tanggal, waktu, mealType, makanan, kalori);
        }
    }

    fclose(f);

    if (!found) {
        printf("You haven't logged any meals in the last 7 days
            .\n");
    }
}
```

FLOWCHART (meal.c)



a
b
c
d

```c
//                          6.1 Menu Daily Meal
// ------------------------------------------------------------
    ----
// Tujuan: Menampilkan menu kepada user sehigga fitur daily
    meal dapat diakses
void dailyMealMenu(const char *email) {
    int pilih;

    // Menampilkan menu fitur yang ada di daily meal
    while (1) {
        printf("\n===== DAILY MEAL =====\n");
        printf("1. Meal History\n");
        printf("2. Add Food\n");
        printf("3. Back\n");
        printf("Your choice: ");
        scanf("%d", &pilih);
        getchar();

        // Mengeksekusi fitur daily meal
        if (pilih == 1) {
            showMealHistory(email);
        } else if (pilih == 2) {
            printf("\nPick your meal type:\n");
            printf("1. Breakfast\n2. Lunch\n3. Dinner\nYour
                choice: ");
            int mealTypeChoice;
            scanf("%d", &mealTypeChoice);
            getchar();
```

```c
    // Meminta input nama makanan dari user
    printf("Please enter the name of the food: ");
    fgets(namaMakanan, sizeof(namaMakanan), stdin);
    namaMakanan[strcspn(namaMakanan, "\n")] = '\0'; //
        Menghapus newline

    // Jika makanan tidak ditemukan di dataset
    if (!foodExistsInDataset(namaMakanan)) {
        printf("This food is not found in our dataset.\n");
        printf("Please enter the estimated calories per 100
            grams for %s: ", namaMakanan);
        scanf("%d", &kaloriPer100g);
        getchar();

        // Meminta jumlah gram yang dikonsumsi
        printf("How many grams of %s did you eat? ",
            namaMakanan);
        scanf("%d", &gram);
        getchar();

        kaloriTotal = (kaloriPer100g * gram) / 100;
        printf("Estimated calories: %d kcal (from %d grams).\n"
            , kaloriTotal, gram);
```

e

```c
        // Menyimpan makanan baru ke dalam dataset (sebagai
            UserAdded)
        FILE *fappend = fopen("dataset/calories.txt", "a");
        if (fappend != NULL) {
            float kj = kaloriPer100g * 4.184;
            fprintf(fappend, "UserAdded,%s,100g,%d cal,%.0f
                kJ\n", namaMakanan, kaloriPer100g, kj);
            fclose(fappend);
            printf("Thanks! %s has been added to the dataset
                .\n", namaMakanan);
        } else {
            printf("Oops! Failed to save the food to the
                dataset.\n");
        }
```

UNIVERSITAS INDONESIA
Veritas Probitas Iustitia
FAKULTAS TEKNIK

LIBRARY

DEMONSTRATION

THANK YOU

KELOMPOK 10