# LAPORAN PRAKTIKUM

# POSTTEST 2

# ALGORITMA PEMROGRAMAN LANJUT


**Disusun oleh:**

**Nama (2409106093)**

**Kelas (C1 ‘24)**

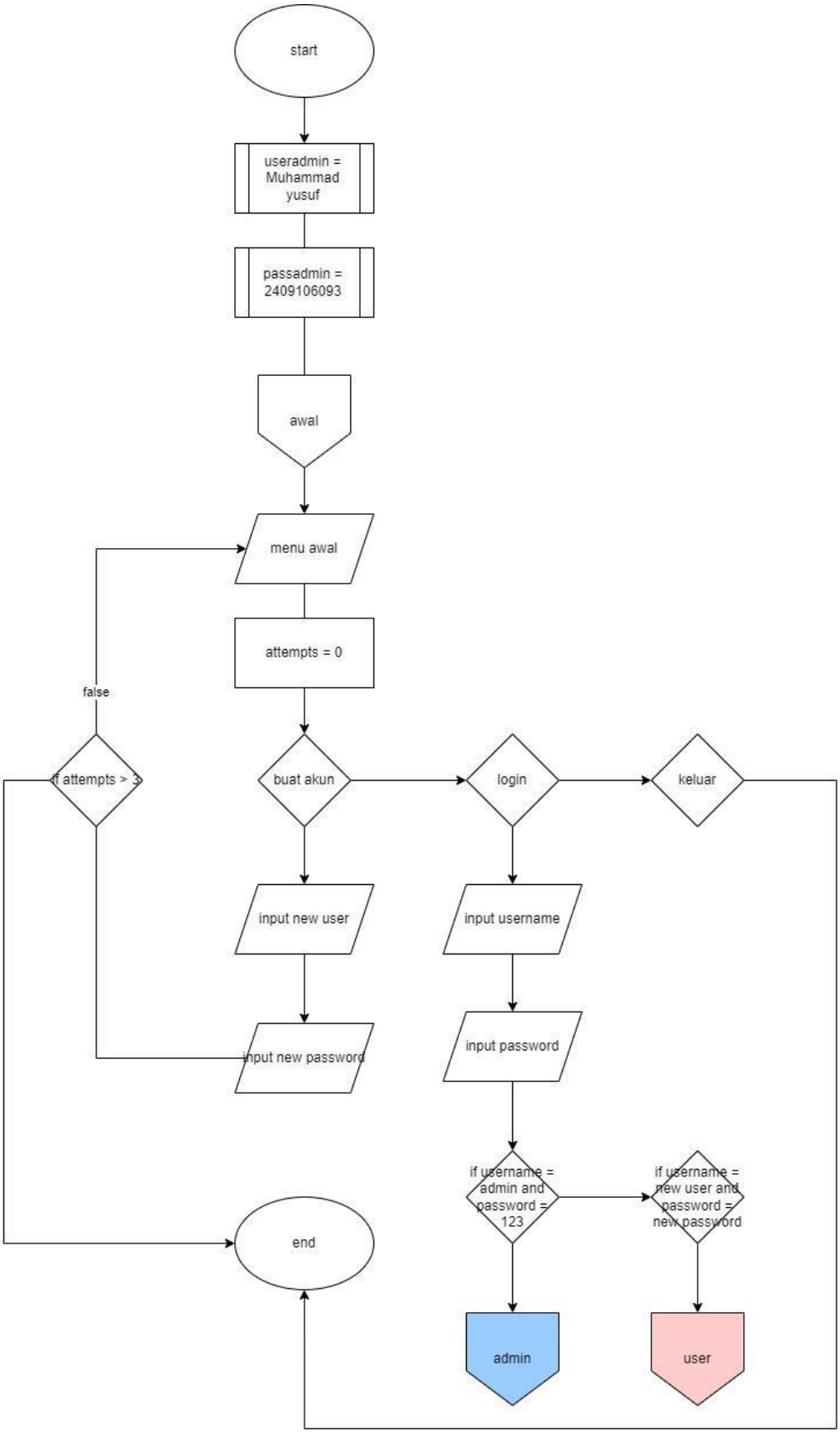
**PROGRAM STUDI INFORMATIKA**

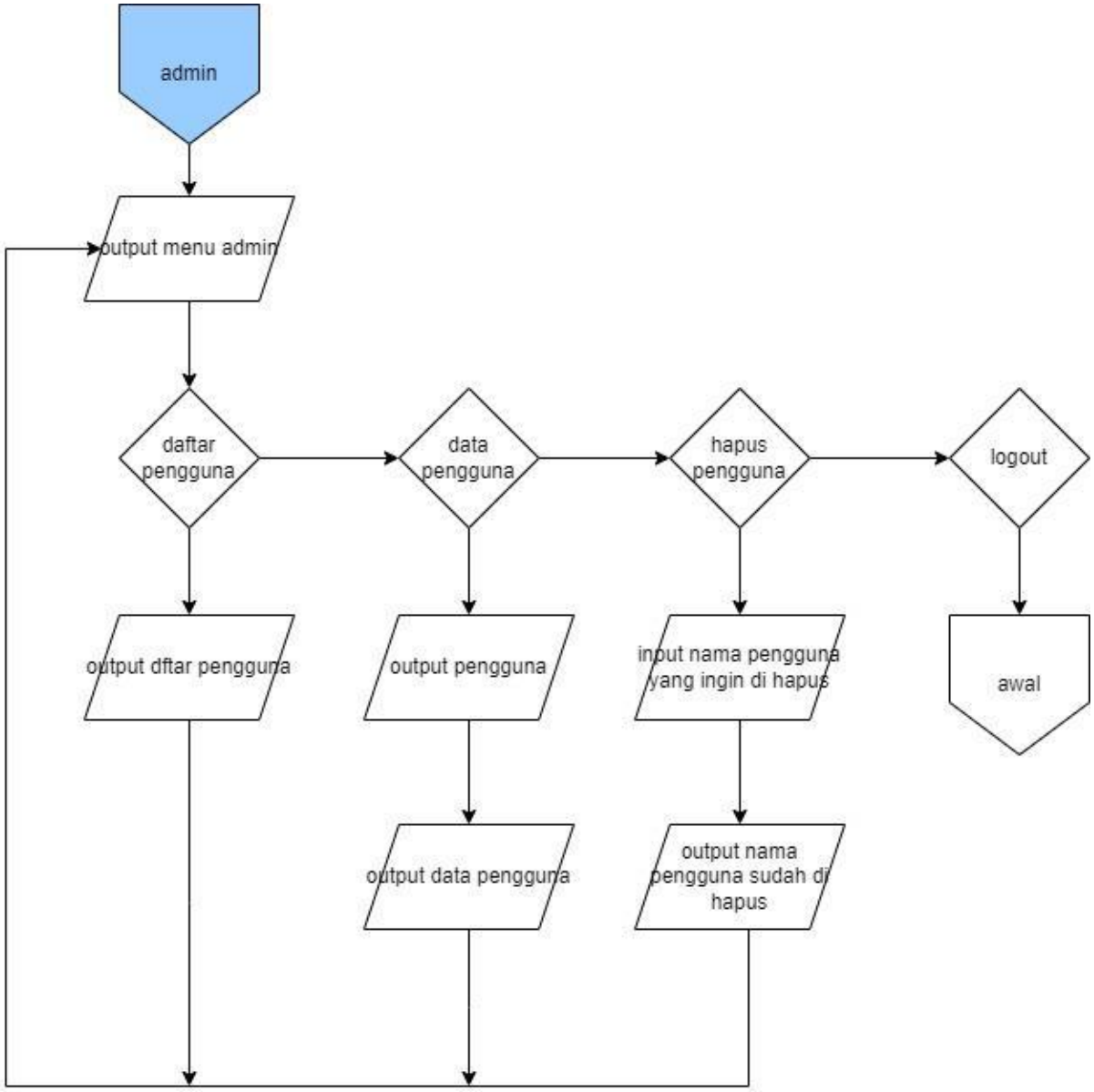
**UNIVERSITAS MULAWARMAN**

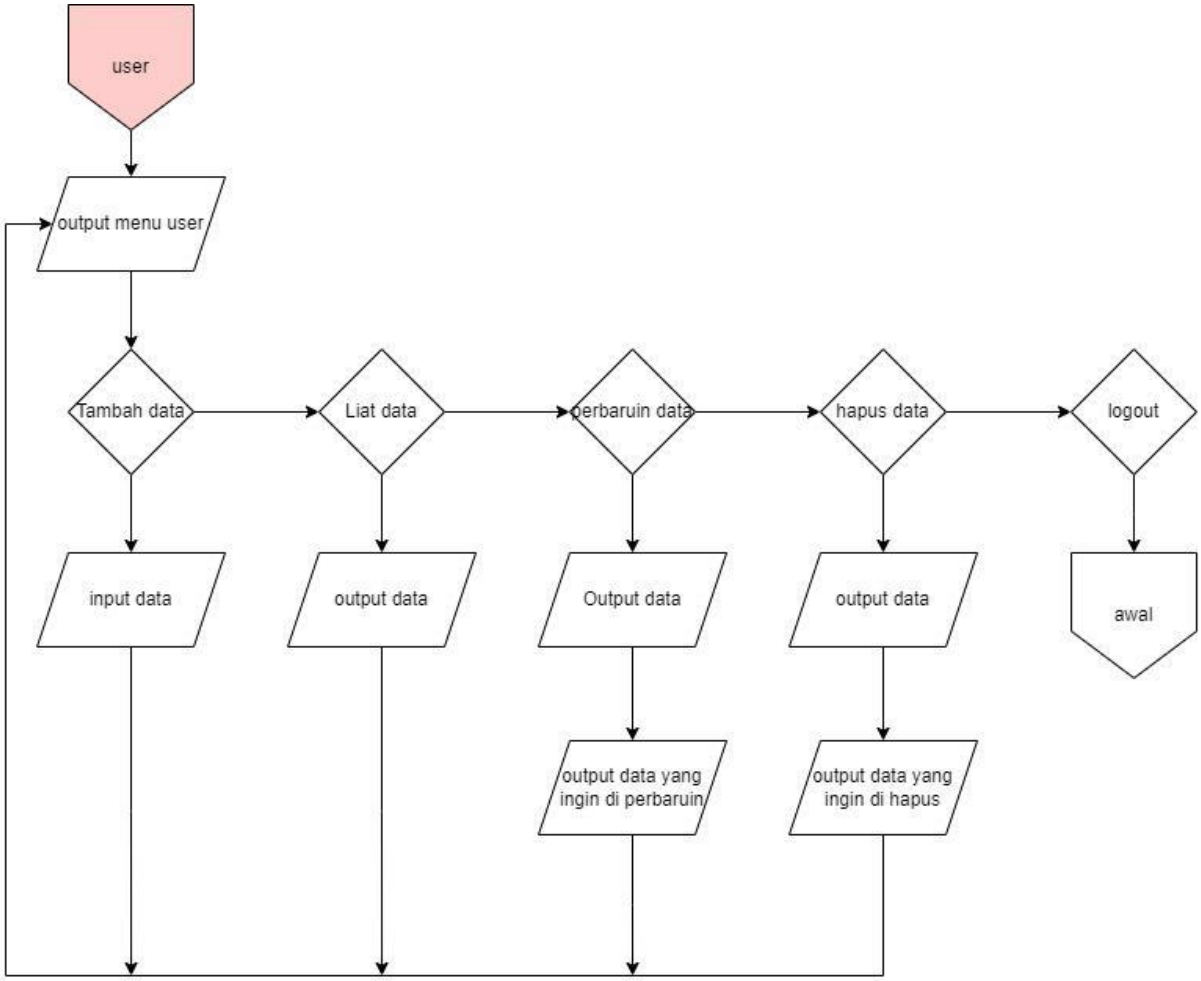
**SAMARINDA**

**2025**

## 1. Flowchart


start
useradmin = Muhammad yusuf
passadmin = 2409106093
awal
menu awal
attempts = 0
false
if attempts > 3
buat akun
login
keluar
input new user
input new password
input username
input password
if username = admin and password = 123
if username = new user and password = new password
end
admin
user


admin
output menu admin
daftar pengguna
data pengguna
hapus pengguna
logout
output dftar pengguna
output pengguna
input nama pengguna yang ingin di hapus
awal
output data pengguna
output nama pengguna sudah di hapus


user
output menu user
Tambah data
Liat data
perbaruin data
hapus data
logout
input data
output data
Output data
output data
awal
output data yang ingin di perbaruin
output data yang ingin di hapus

## 2. Analisis Program

### 2.1 Deskripsi Singkat Program

Program **AI Archive** ini bertujuan untuk menyediakan sistem penyimpanan dan pengelolaan data berbasis akun pengguna. Manfaat utama program ini adalah:

1. **Manajemen Pengguna**
    - Memungkinkan pengguna untuk membuat akun dengan nama (username) dan NIM (password).
    - Admin memiliki kontrol penuh untuk melihat, mengelola, dan menghapus akun pengguna.
2. **Penyimpanan Data pengguna**
    - Setiap pengguna dapat menambahkan, membaca, memperbarui, dan menghapus data mereka sendiri.
    - Data yang dimasukkan tersimpan secara terstruktur sesuai dengan akun pengguna masing-masing.
3. **Keamanan dan Validasi**
    - Sistem login dengan batasan tiga kali percobaan untuk mencegah akses tidak sah.
    - Admin memiliki akses khusus untuk melihat seluruh data pengguna dan melakukan pengelolaan akun.
4. **Interaksi yang Mudah**
    - Menggunakan menu berbasis teks yang intuitif untuk navigasi dan pengelolaan data.
    - Menyediakan fitur logout agar pengguna bisa keluar dari sesi mereka dengan aman.

Dengan fitur-fitur ini, program dapat digunakan sebagai sistem pencatatan sederhana yang memungkinkan pengguna menyimpan dan mengelola informasi pribadi dengan aman.

## 2.2 Penjelasan Alur & Algoritma

1. Deklarasi dan validasi

```
#include <iostream>
#include <string>

using namespace std;

const int MAX_USERS = 100;
const int MAX_DATA = 50;

string usernames[MAX_USERS];
string passwords[MAX_USERS];
string user_data[MAX_USERS][MAX_DATA];
int data_count[MAX_USERS] = {0};
int total_users = 0;
```

2. Data admin

```
string useradmin = "Muhammad Yusuf";
string passadmin = "2409106093";
```

Admin memiliki username "Muhammad Yusuf" dan password "2409106093".

3. Fungsi mencari pengguna

```
int findUser(string username) {
    for (int i = 0; i < total_users; i++) {
        if (usernames[i] == username) return i;
    }
    return -1;
}
```

- **Mencari user dalam array usernames**.
- Jika ditemukan, mengembalikan indeks pengguna.
- Jika tidak ditemukan, mengembalikan -1.

4. Menu utama

```
int main() {
    while (true) {
        cout << "\n====== AI Archive ======\n";
        cout << "1. Buat Akun\n";
        cout << "2. Login\n";
        cout << "3. Keluar\n";
        cout << "Pilih opsi: ";
```

5. Pembuatan akun

```
if (opsi == 1) {
            if (total_users >= MAX_USERS) {
                cout << "Maksimal pengguna telah tercapai!\n";
                continue;
            }

            string username, password;
            cout << "Masukkan Nama: ";
            getline(cin, username);

            if (findUser(username) != -1) {
                cout << "Username sudah terdaftar!\n";
            } else {
                cout << "Masukkan NIM: ";
                getline(cin, password);

                usernames[total_users] = username;
                passwords[total_users] = password;
                total_users++;

                cout << "Akun berhasil dibuat!\n";
            }
        }
```

- Jika jumlah user sudah mencapai batas, pendaftaran tidak diperbolehkan.
- Meminta user memasukkan nama dan NIM (password).
- Jika username sudah ada, pendaftaran gagal.

- Jika username belum ada, akun dibuat dan ditambahkan ke database.

6. Login user atau admin

```
else if (opsi == 2) {
            string username, password;
            int attempts = 3;

            while (attempts > 0) {
                cout << "Input Nama: ";
                getline(cin, username);
                cout << "Input NIM: ";
                getline(cin, password);

                if (username == useradmin && password == passadmin) {
                    cout << "Login Admin Berhasil!\n";
                    bool admin_logged_in = true;
```

- Maksimal 3 percobaan login.
- Jika username dan password cocok dengan admin → Masuk sebagai admin.
- Jika cocok dengan user biasa → Masuk sebagai user.

7. Opsi admin

```
while (admin_logged_in) {
     cout << "\n=== Admin Mode ===\n";
     cout << "1. Lihat Semua Pengguna\n";
     cout << "2. Lihat Data Pengguna\n";
     cout << "3. Hapus Pengguna\n";
     cout << "4. Logout\n";
     cout << "Pilih menu: ";
```

- **Opsi 1:** Menampilkan daftar semua pengguna.
- **Opsi 2:** Menampilkan data semua pengguna.
- **Opsi 3:** Menghapus pengguna.
- **Opsi 4:** Logout admin.

8. Opsi user

```
while (logged_in) {
     cout << "\n=== AI Archive ===\n";
     cout << "1. Tambah Data\n";
     cout << "2. Baca Data\n";
     cout << "3. Perbarui Data\n";
     cout << "4. Hapus Data\n";
     cout << "5. Logout\n";
     cout << "Pilih menu: ";
```

- **Opsi 1:** Menambahkan data ke akun.
- **Opsi 2:** Menampilkan data yang telah disimpan.
- **Opsi 3:** Memperbarui data yang sudah ada.
- **Opsi 4:** Menghapus data.
- **Opsi 5:** Logout.

### 3. Source Code

```
int main() {
    while (true) {
        cout << "\n====== AI Archive ======\n";
        cout << "1. Buat Akun\n";
        cout << "2. Login\n";
        cout << "3. Keluar\n";
        cout << "Pilih opsi: ";

        int opsi;
        cin >> opsi;
        cin.ignore();

        if (opsi == 1) {
            if (total_users >= MAX_USERS) {
                cout << "Maksimal pengguna telah tercapai!\n";
                continue;
            }

            string username, password;
            cout << "Masukkan Nama: ";
            getline(cin, username);

            if (findUser(username) != -1) {
                cout << "Username sudah terdaftar!\n";
            } else {
                cout << "Masukkan password: ";
                getline(cin, password);

                usernames[total_users] = username;
                passwords[total_users] = password;
                total_users++;

                cout << "Akun berhasil dibuat!\n";
            }
        }
```

```
        else if (opsi == 2) {
            string username, password;
            int attempts = 3;

            while (attempts > 0) {
                cout << "Input Nama: ";
                getline(cin, username);
                cout << "Input password: ";
                getline(cin, password);

                if (username == useradmin && password == passadmin) {
                    cout << "Login Admin Berhasil!\n";
                    bool admin_logged_in = true;

                    while (admin_logged_in) {
                        cout << "\n=== Admin Mode ===\n";
                        cout << "1. Lihat Semua Pengguna\n";
                        cout << "2. Lihat Data Pengguna\n";
                        cout << "3. Hapus Pengguna\n";
                        cout << "4. Logout\n";
                        cout << "Pilih menu: ";

                        int opsiadmin;
                        cin >> opsiadmin;
                        cin.ignore();

                        if (opsiadmin == 1) {
                            cout << "\nDaftar Pengguna:\n";
                            for (int i = 0; i < total_users; i++) {
                                cout << "- " << usernames[i] << endl;
                            }
                        }
                        else if (opsiadmin == 2) {
                            cout << "\n=== Data Seluruh Pengguna ===\n";
                            if (total_users == 0) {
                                cout << "Tidak ada pengguna terdaftar.\n";
                            } else {
                                for (int i = 0; i < total_users; i++) {
                                    cout << "\n Pengguna: " << usernames[i]
<< "\n";

                                    if (data_count[i] == 0) {
                                        cout << "   →Tidak ada data
tersimpan.\n";

                                    } else {
                                        for (int j = 0; j < data_count[i];
j++) {

                                            cout << "   " << j + 1 << ". "
<< user_data[i][j] << endl;
```

```
                                    }
                                }
                            }
                        }
                    }
                    else if (opsiadmin == 3) {
                        cout << "Masukkan nama pengguna yang ingin
dihapus: ";
                        string user;
                        getline(cin, user);

                        int index = findUser(user);
                        if (index != -1) {
                            for (int i = index; i < total_users - 1;
i++) {
                                usernames[i] = usernames[i + 1];
                                passwords[i] = passwords[i + 1];
                                data_count[i] = data_count[i + 1];
                                for (int j = 0; j < MAX_DATA; j++) {
                                    user_data[i][j] = user_data[i +
1][j];
                                }
                            }
                            total_users--;
                            cout << "Pengguna " << user << " berhasil
dihapus.\n";
                        } else {
                            cout << "Pengguna tidak ditemukan.\n";
                        }
                    }
                    else if (opsiadmin == 4) {
                        cout << "Logout berhasil.\n";
                        admin_logged_in = false;
                    }
                    else {
                        cout << "Pilihan tidak valid.\n";
                    }
                }
                break;
            }

            int userIndex = findUser(username);
            if (userIndex == -1 || passwords[userIndex] != password) {
                cout << "Username atau password salah!\n";
                attempts--;
                cout << "Sisa percobaan: " << attempts << endl;
                if (attempts == 0) {
                    cout << "Gagal login 3 kali. Coba lagi nanti.\n";
```

```
                        break;
                    }
                    continue;
                }

                cout << "Login berhasil!\n";
                bool logged_in = true;

                while (logged_in) {
                    cout << "\n=== AI Archive ===\n";
                    cout << "1. Tambah Data\n";
                    cout << "2. Baca Data\n";
                    cout << "3. Perbarui Data\n";
                    cout << "4. Hapus Data\n";
                    cout << "5. Logout\n";
                    cout << "Pilih menu: ";

                    int opsiuser;
                    cin >> opsiuser;
                    cin.ignore();

                    if (opsiuser == 1) {
                        if (data_count[userIndex] >= MAX_DATA) {
                            cout << "Maksimal data tercapai!\n";
                            continue;
                        }
                        cout << "Masukkan data baru: ";
                        getline(cin,
user_data[userIndex][data_count[userIndex]]);
                        data_count[userIndex]++;
                        cout << "Data berhasil ditambahkan.\n";
                    }
                    else if (opsiuser == 2) {
                        cout << "\nData Anda:\n";
                        if (data_count[userIndex] == 0) {
                            cout << "Tidak ada data tersimpan.\n";
                        } else {
                            for (int i = 0; i < data_count[userIndex]; i++)
{
                                cout << i + 1 << ". " <<
user_data[userIndex][i] << endl;
                            }
                        }
                    }
                    else if (opsiuser == 3) {
                        cout << "\nData Anda:\n";
                        if (data_count[userIndex] == 0) {
                            cout << "Tidak ada data tersimpan.\n";
```

```
                        } else {
                            for (int i = 0; i < data_count[userIndex]; i++)
{
                                cout << i + 1 << ". " <<
user_data[userIndex][i] << endl;
                            }
                        }

                        cout << "Masukkan nomor data yang ingin diperbarui:
";
                        int index;
                        cin >> index;
                        cin.ignore();

                        if (index > 0 && index <= data_count[userIndex]) {
                            cout << "Masukkan data baru: ";
                            getline(cin, user_data[userIndex][index - 1]);
                            cout << "Data berhasil diperbarui.\n";
                        } else {
                            cout << "Nomor data tidak valid!\n";
                        }
                    }
                    else if (opsiuser == 4) {
                        cout << "\nData Anda:\n";
                        if (data_count[userIndex] == 0) {
                            cout << "Tidak ada data tersimpan.\n";
                        } else {
                            for (int i = 0; i < data_count[userIndex]; i++)
{
                                cout << i + 1 << ". " <<
user_data[userIndex][i] << endl;
                            }
                        }

                        cout << "Masukkan nomor data yang ingin dihapus: ";
                        int index;
                        cin >> index;
                        cin.ignore();

                        if (index > 0 && index <= data_count[userIndex]) {
                            for (int i = index - 1; i <
data_count[userIndex] - 1; i++) {
                                user_data[userIndex][i] =
user_data[userIndex][i + 1];
                            }
                            data_count[userIndex]--;
                            cout << "Data berhasil dihapus.\n";
                        } else {
```

```
                            cout << "Nomor data tidak valid!\n";
                        }
                    }
                    else if (opsiuser == 5) {
                        cout << "Logout berhasil.\n";
                        logged_in = false;
                    }
                    else {
                        cout << "Pilihan tidak valid.\n";
                    }
                }
                break;
            }
        }
        else if (opsi == 3) {
            cout << "Keluar dari sistem.\n";
            break;
        }
        else {
            cout << "Pilihan tidak valid.\n";
        }
    }
    return 0;
}
```

## 4. Uji Coba dan Hasil Output

### 4.1 Uji Coba

1. Pengguna memilih membuat akun
   Pengguna di minta mamasukan new user dan new password

2. Pengguna memilih login
   Jika kombinasi nama dan password benar, login berhasil.
   Jika salah, program memberikan **3 kesempatan** sebelum gagal login.

3. Metode admin (jika login menggunakan admin)
   Pengguna memasukan user = “Muhammad Yusuf” dan password “2409106093”
   maka program masuk ke **Mode Admin**, dengan opsi:
   - Admin bisa melihat semua user dan data mereka.
   - Admin juga bisa **menghapus pengguna** dengan memasukkan nama.

4. Mode User (Jika Login sebagai Pengguna Biasa)
   Setelah login, pengguna bisa mengelola data dengan menu
   - Jika memilih **1 (Tambah Data)**, pengguna bisa memasukkan teks
   - Jika memilih **2 (Baca Data)**, program menampilkan semua data pengguna
   - Jika memilih **3 (Perbarui Data)**, pengguna bisa memilih nomor data untuk diperbarui
   - Jika memilih **4 (Hapus Data)**, pengguna bisa menghapus data tertentu

5. Log out
   - Jika memilih **5 (Logout)**, pengguna akan keluar ke menu utama.
   - Jika memilih **3 (Keluar)** di menu utama, program berhenti.

## 4.2 Hasil Output

```
====== AI Archive ======
1. Buat Akun
2. Login
3. Keluar
Pilih opsi: 1
Masukkan Nama: ali
Masukkan password: 123
Akun berhasil dibuat!
```

```
====== AI Archive ======
1. Buat Akun
2. Login
3. Keluar
Pilih opsi: 2
Input Nama: ali
Input password: 123
Login berhasil!

=== AI Archive ===
1. Tambah Data
2. Baca Data
3. Perbarui Data
4. Hapus Data
5. Logout
Pilih menu:
```

```
=== AI Archive ===
1. Tambah Data
2. Baca Data
3. Perbarui Data
4. Hapus Data
5. Logout
Pilih menu: 1
Masukkan data baru: catatan kuliah ai
Data berhasil ditambahkan.
```

```
Data Anda:
1. catatan kuliah ai
Masukkan nomor data yang ingin diperbarui: 1
Masukkan data baru: praktikum
Data berhasil diperbarui.

=== AI Archive ===
1. Tambah Data
2. Baca Data
3. Perbarui Data
4. Hapus Data
5. Logout
Pilih menu: 2

Data Anda:
1. praktikum
```

```
====== AI Archive ======
1. Buat Akun
2. Login
3. Keluar
Pilih opsi: 2
Input Nama: Muhammad Yusuf
Input password: 2409106093
Login Admin Berhasil!
```

```
Daftar Pengguna:
- ali

=== Admin Mode ===
1. Lihat Semua Pengguna
2. Lihat Data Pengguna
3. Hapus Pengguna
4. Logout
Pilih menu: 2

=== Data Seluruh Pengguna ===

 fö Pengguna: ali
   1. praktikum

=== Admin Mode ===
1. Lihat Semua Pengguna
2. Lihat Data Pengguna
3. Hapus Pengguna
4. Logout
Pilih menu:
```

```
=== Admin Mode ===
1. Lihat Semua Pengguna
2. Lihat Data Pengguna
3. Hapus Pengguna
4. Logout
Pilih menu: 3
Masukkan nama pengguna yang ingin dihapus: ali
Pengguna ali berhasil dihapus.

=== Admin Mode ===
1. Lihat Semua Pengguna
2. Lihat Data Pengguna
3. Hapus Pengguna
4. Logout
Pilih menu:
```

```
=== Admin Mode ===
1. Lihat Semua Pengguna
2. Lihat Data Pengguna
3. Hapus Pengguna
4. Logout
Pilih menu: 4
Logout berhasil.

====== AI Archive ======
1. Buat Akun
2. Login
3. Keluar
Pilih opsi: 3
Keluar dari sistem.
PS D:\GITHUB\Praktikum-Apl\post-test\post-test-apl-2>
```

**5. Git**

```
MINGW64:/d/GITHUB/Praktikum-Apl

ASUS@LAPTOP-7HLBA3LM MINGW64 /d/GITHUB/Praktikum-Apl (main)
$ git add .

ASUS@LAPTOP-7HLBA3LM MINGW64 /d/GITHUB/Praktikum-Apl (main)
$ git commit -m "post-test-2"
[main 0667cae] post-test-2
 2 files changed, 259 insertions(+)
 create mode 100644 post-test/post-test-apl-2/2409106093-MuhammadYusuf-PT-2.cpp
 create mode 100644 post-test/post-test-apl-2/2409106093-MuhammadYusuf-PT-2.exe

ASUS@LAPTOP-7HLBA3LM MINGW64 /d/GITHUB/Praktikum-Apl (main)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 679.27 KiB | 8.49 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/qweprr/praktikum-apl.git
   58c5d89..0667cae  main -> main

ASUS@LAPTOP-7HLBA3LM MINGW64 /d/GITHUB/Praktikum-Apl (main)
$
```

1. Menambahkan File ke Staging Area
   $ git add .
   • Perintah ini menambahkan semua file yang ada di dalam folder ke staging area.
   • Staging area adalah tempat sementara sebelum file dikomit ke dalam repository.

2. Menambahkan Remote Repository (Gagal karena Sudah Ada)
   $ git remote add origin https://github.com/qweprr/praktikum-apl
   • Perintah ini digunakan untuk menambahkan repository remote dengan nama origin.
   • Error: "remote origin already exists", ini terjadi karena sebelumnya sudah ada repository remote yang dikaitkan dengan nama origin.
   •
3. Membuat Commit dengan Pesan "Update"
   $ git commit -m "Update"
   • Perintah ini menyimpan perubahan dalam repository dengan commit dan pesan"Update".
   • File yang dikomit:
      - Post-test/Post-test-1/2409106093-Muhammadyusuf-PT-1.cpp
      - Post-test/Post-test-1/2409106093-Muhammadyusuf-PT-1.cpp
4. Mendorong (Push) Perubahan ke Repository Remote
   $ git push -u origin main
   • Perintah ini mengunggah (push) perubahan ke repository remote pada branch main.
   • Karena ini adalah push pertama, flag -u digunakan untuk mengatur branch lokal main agar terhubung dengan branch main di remote repository.
   • Proses yang terjadi:
      - Menghitung objek (Enumerating objects: 6).
      - Mengompresi objek sebelum mengunggahnya.
      - Menulis (mengunggah) objek ke GitHub.
      - Menampilkan informasi bahwa branch main sekarang dilacak oleh remote repository origin/main.