

LAPORAN PRAKTIKUM
POSTTEST 3
ALGORITMA PEMROGRAMAN LANJUT



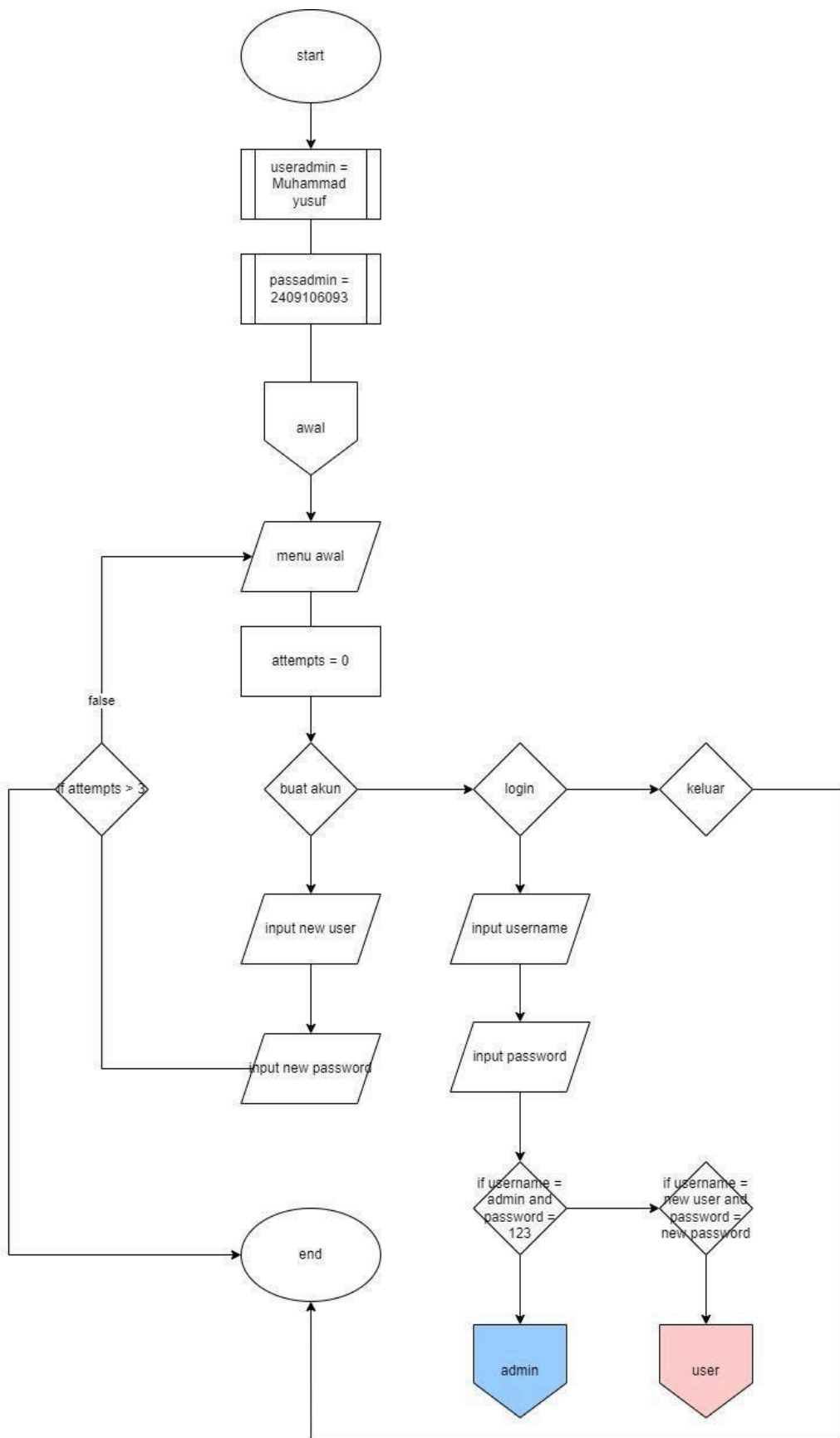
Disusun oleh:

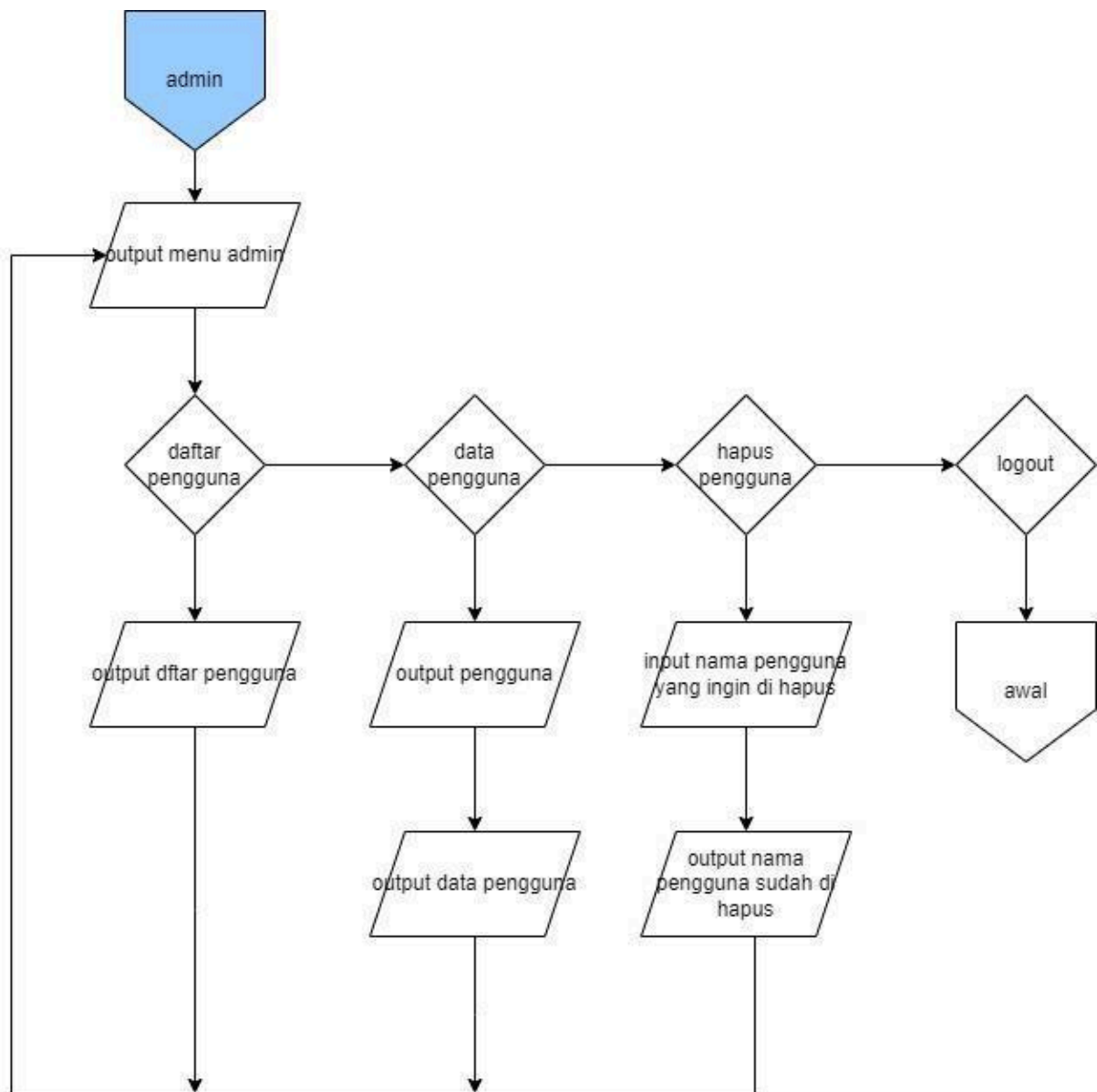
Nama (2409106093)

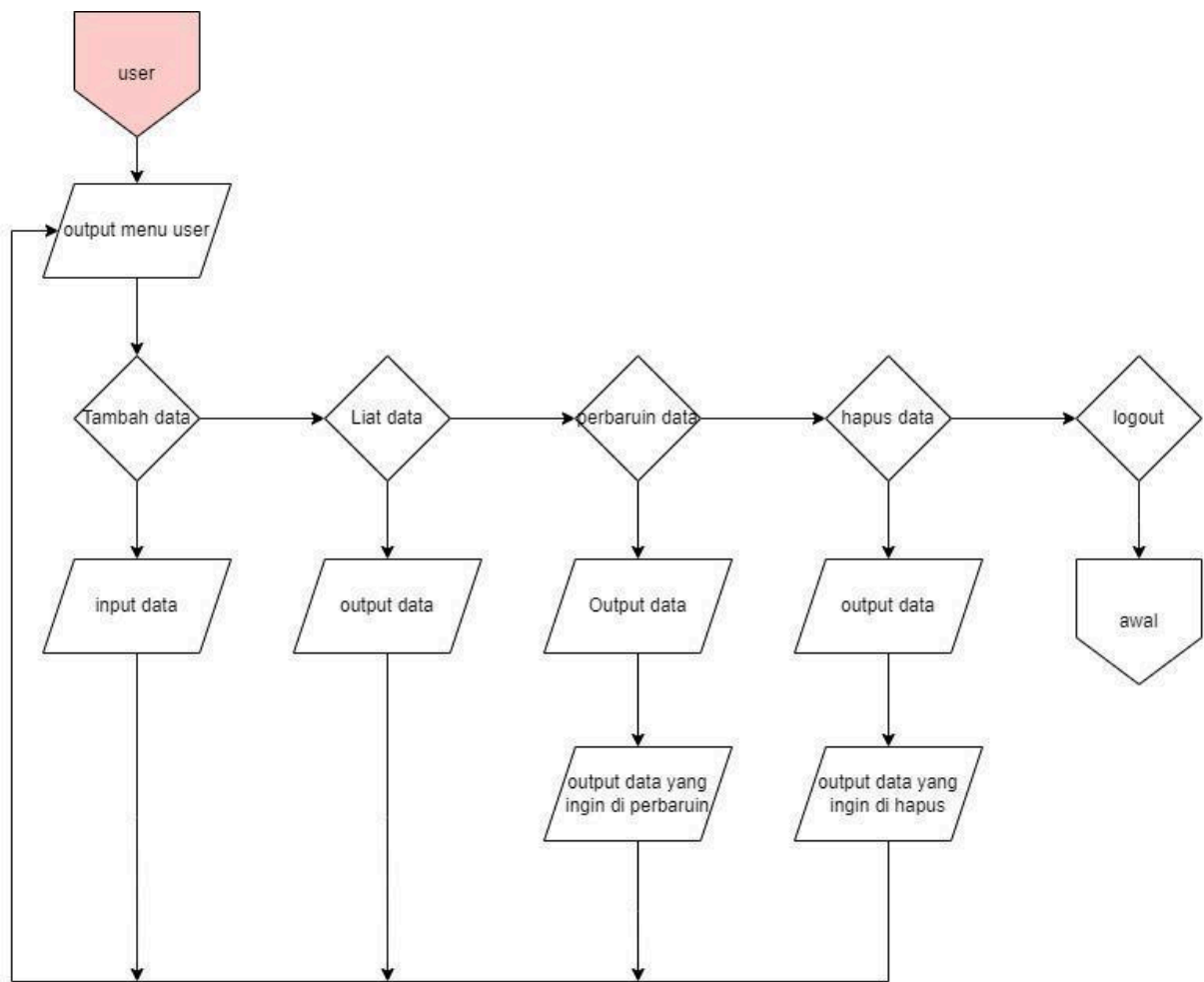
Kelas (C1'24)

PROGRAM STUDI INFORMATIKA
UNIVERSITAS MULAWARMAN
SAMARINDA
2025

1. Flowchart







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>
#include <iomanip> // Untuk setw dan setfill

using namespace std;
```

2. Pencarian user

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

- **Input:** username
- **Proses:** Mencari apakah username sudah terdaftar.
- **Output:** Mengembalikan indeks user jika ditemukan, atau -1 jika tidak ada.

3. Menambahkan user baru

```
if (findUser(username) != -1) {
    cout << "Username sudah terdaftar!\n";
} else {
    cout << "Masukkan password: ";
    getline(cin, password);
    users[total_users].username = username;
    users[total_users].password = password;
    total_users++;

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

- **Cek username** apakah sudah ada.
- **Jika** belum ada, akun ditambahkan ke array users[].
- **total_users** ditambah 1.

4. Menu utama

```
int main() {
    while (true) {
        // Tampilan menu utama dalam bentuk tabel
        cout << "\n===== AI Archive =====\n";
        cout << "+-----+-----+-----+\n";
        cout << "| No | Opsi | \n";
        cout << "+-----+-----+-----+\n";
        cout << "| 1 | Buat Akun | \n";
        cout << "| 2 | Login | \n";
        cout << "| 3 | Keluar | \n";
        cout << "+-----+-----+-----+\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 password: ";
        getline(cin, password);

        users[total_users].username = username;
        users[total_users].password = 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 password: ";
        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.

Opsi admin

```
while (admin_logged_in) {
    // Tampilan menu admin dalam bentuk tabel
    cout << "\n=== Admin Mode ===\n";
    cout << "+-----+-----+\n";
    cout << "| No | Opsi | \n";
    cout << "+-----+-----+\n";
    cout << "| 1 | Lihat Semua Pengguna | \n";
    cout << "| 2 | Lihat Data Pengguna | \n";
    cout << "| 3 | Hapus Pengguna | \n";
    cout << "| 4 | Logout | \n";
    cout << "+-----+-----+\n";
    cout << "Pilih menu: ";
}
```

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

- **Opsi 4:** Logout admin.

7. Opsi user

```
while (logged_in) {
// Tampilan menu pengguna dalam bentuk tabel
    cout << "\n=== AI Archive ===\n";
    cout << "+-----+-----+-----+\n";
    cout << "| No | Opsi | \n";
    cout << "+-----+-----+-----+\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 << "+-----+-----+-----+\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

```
// Definisi struct untuk menyimpan informasi pengguna
struct User {
    string username;
    string password;
    string data[MAX_DATA];
    int data_count = 0;
};

User users[MAX_USERS];
int total_users = 0;

// Data Admin
string useradmin = "Muhammad Yusuf";
string passadmin = "2409106093";

// Fungsi mencari indeks pengguna berdasarkan username
int findUser(string username) {
    for (int i = 0; i < total_users; i++) {
        if (users[i].username == username) return i;
    }
    return -1;
}

int main() {
    while (true) {
        // Tampilan menu utama dalam bentuk tabel
        cout << "\n===== AI Archive =====\n";
        cout << "+-----+-----+-----+\n";
        cout << "| No | Opsi | \n";
        cout << "+-----+-----+-----+\n";
        cout << "| 1 | Buat Akun | \n";
        cout << "| 2 | Login | \n";
        cout << "| 3 | Keluar | \n";
        cout << "+-----+-----+-----+\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);

    users[total_users].username = username;
    users[total_users].password = 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) {
                // Tampilan menu admin dalam bentuk tabel
                cout << "\n=== Admin Mode ===\n";
                cout << "+-----+-----+\n";
                cout << "| No | Opsi | \n";
                cout << "+-----+-----+\n";
                cout << "| 1 | Lihat Semua Pengguna | \n";
                cout << "| 2 | Lihat Data Pengguna | \n";
                cout << "| 3 | Hapus Pengguna | \n";
                cout << "| 4 | Logout | \n";
                cout << "+-----+-----+\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 << "- " << users[i].username << 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: " <<
users[i].username << "\n";

                    if (users[i].data_count == 0) {
                        cout << "      → Tidak ada data
tersimpan.\n";
                    } else {
                        for (int j = 0; j <
users[i].data_count; j++) {
                            cout << "      " << j + 1 << ". "
<< users[i].data[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++) {
                    users[i] = users[i + 1];
                }
                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 || users[userIndex].password !=
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) {
    // Tampilan menu pengguna dalam bentuk tabel
    cout << "\n=== AI Archive ===\n";
    cout << "+-----+-----+-----+\n";
    cout << "| No | Opsi | \n";
    cout << "+-----+-----+-----+\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 << "+-----+-----+-----+\n";
    cout << "Pilih menu: ";

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

    if (opsiuser == 1) {
        if (users[userIndex].data_count >= MAX_DATA) {
            cout << "Maksimal data tercapai!\n";
            continue;
        }
    }
}

```

```

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

        cout << "Masukkan nomor data yang ingin diperbarui:
";

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

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

```

```

        cout << "Tidak ada data tersimpan.\n";
    } else {
        for (int i = 0; i < users[userIndex].data_count;
i++) {
            cout << i + 1 << ". " <<
users[userIndex].data[i] << endl;
        }
    }

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

    if (index > 0 && index <=
users[userIndex].data_count) {
        for (int i = index - 1; i <
users[userIndex].data_count - 1; i++) {
            users[userIndex].data[i] =
users[userIndex].data[i + 1];
        }
        users[userIndex].data_count--;
        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 memasukan 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

```
PS D:\GITHUB> cd "d:\GITHUB\Praktikum-Apl\post-test\post-test-apl-3\" ; if ($?)
```

```
===== AI Archive =====
```

No	Opsi
1	Buat Akun
2	Login
3	Keluar

Pilih opsi: 1

Masukkan Nama: ali

Masukkan password: 123

Akun berhasil dibuat!

Pilih opsi: 2

Input Nama: ali

Input password: 123

Login berhasil!

```
=== AI Archive ===
```

No	Opsi
1	Tambah Data
2	Baca Data
3	Perbarui Data
4	Hapus Data
5	Logout

Pilih menu:

Pilih menu: 1

Masukkan data baru: catatan apl

Data berhasil ditambahkan.

```
=== AI Archive ===
```

No	Opsi
1	Tambah Data
2	Baca Data
3	Perbarui Data
4	Hapus Data
5	Logout

Pilih menu:

Pilih menu: 2

Data Anda:

1. catatan apl

=== AI Archive ===

```

+-----+
| No | Opsi |
+-----+
| 1 | Tambah Data |
| 2 | Baca Data |
| 3 | Perbarui Data |
| 4 | Hapus Data |
| 5 | Logout |
+-----+

```

Pilih menu:

Pilih menu: 3

Data Anda:

1. catatan apl

2. rangkuman praktikum

Masukkan nomor data yang ingin diperbarui: 2

Masukkan data baru: modul bahasa indonesia

Data berhasil diperbarui.

=== AI Archive ===

```
+-----+
| No | Opsi |
+-----+
| 1 | Tambah Data |
| 2 | Baca Data |
| 3 | Perbarui Data |
| 4 | Hapus Data |
| 5 | Logout |
+-----+
```

Pilih menu:

Pilih menu: 4

Data Anda:

1. catatan apl
2. modul bahasa indonesia

Masukkan nomor data yang ingin dihapus: 2

Data berhasil dihapus.

=== AI Archive ===

No	Opsi
1	Tambah Data
2	Baca Data
3	Perbarui Data
4	Hapus Data
5	Logout

Pilih menu: 2

Data Anda:

1. catatan apl

===== AI Archive =====

No	Opsi
1	Buat Akun
2	Login
3	Keluar

Pilih opsi: 2

Input Nama: Muhammad Yusuf

Input password: 2409106093

Login Admin Berhasil!

Pilih menu: 1

Daftar Pengguna:

- ali

=== Admin Mode ===

No	Opsi
1	Lihat Semua Pengguna
2	Lihat Data Pengguna
3	Hapus Pengguna
4	Logout

Pilih menu:

Pilih menu: 2

=== Data Seluruh Pengguna ===

fö|| Pengguna: ali
1. rangkuman praktikum

=== Admin Mode ===

No	Opsi
1	Lihat Semua Pengguna
2	Lihat Data Pengguna
3	Hapus Pengguna
4	Logout

Pilih menu: 1

Pilih menu: 3

Masukkan nama pengguna yang ingin dihapus: ali
Pengguna ali berhasil dihapus.

=== Admin Mode ===

No	Opsi
1	Lihat Semua Pengguna
2	Lihat Data Pengguna
3	Hapus Pengguna
4	Logout

Pilih menu: 1

Daftar Pengguna:

=== Admin Mode ===

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-3"
[main dc1cb62] post-test-3
2 files changed, 260 deletions(-)
delete mode 100644 post-test/post-test-apl-3/test.cpp
delete mode 100644 post-test/post-test-apl-3/test.exe
ASUS@LAPTOP-7HLBA3LM MINGW64 /d/GITHUB/Praktikum-Apl (main)
$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 834.54 KiB | 3.45 MiB/s, done.
Total 12 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To https://github.com/qwepr/praktikum-apl.git
   d19a125..dc1cb62  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/qwepr/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.