LAPORAN PRAKTIKUM

POSTTEST 3

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Renaya Putri Alika (2409106002)

Kelas A1’24

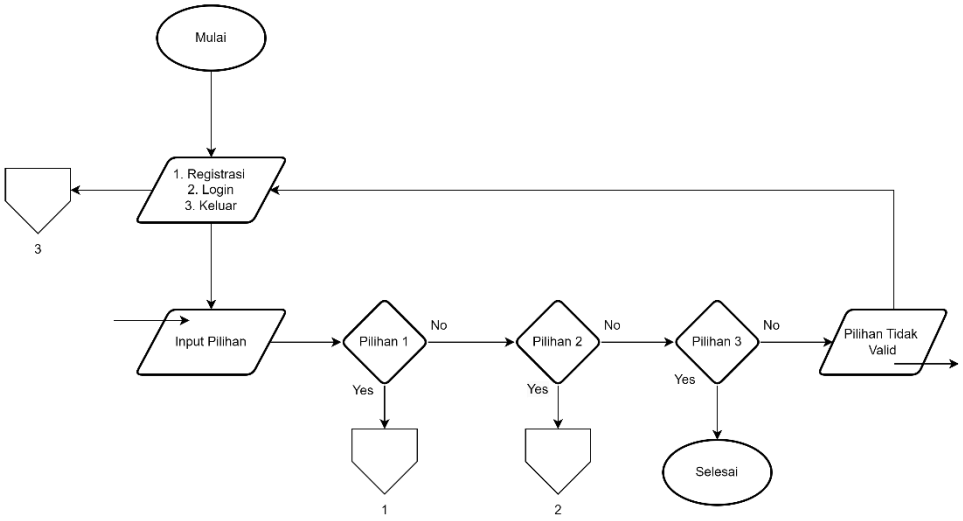
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

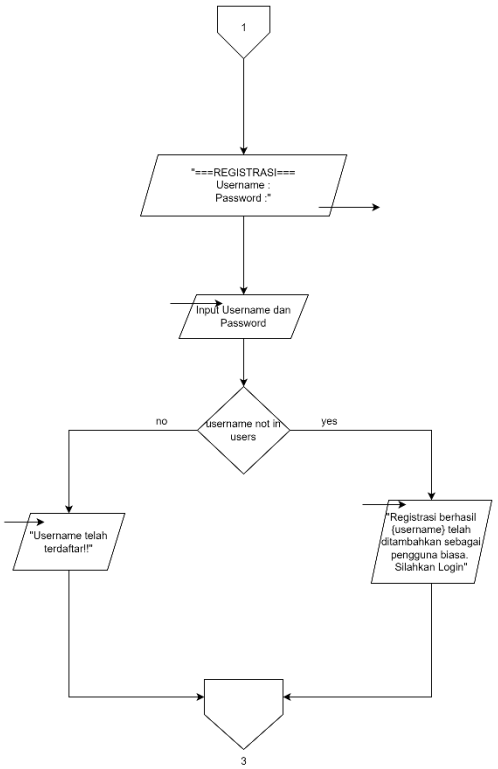
SAMARINDA

2025

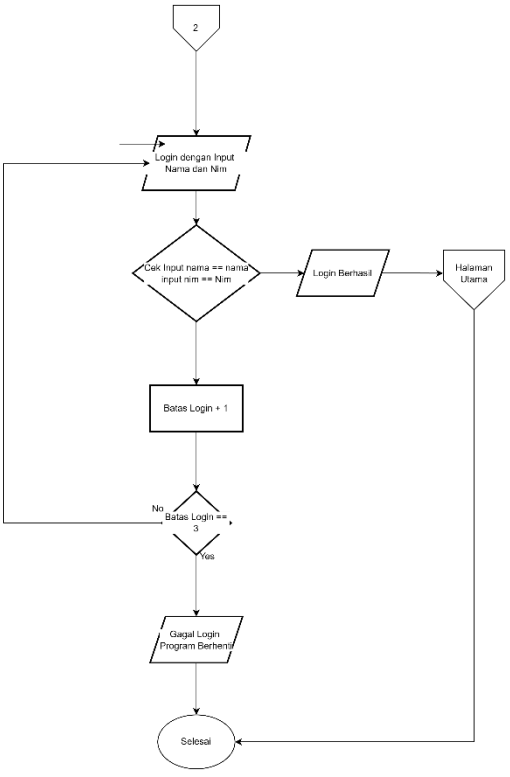
1. Flowchart



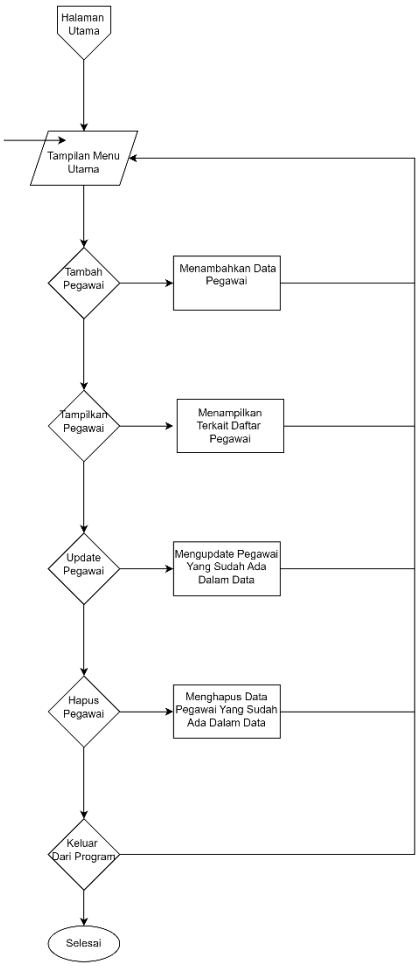
Flowchart 1.1 Pilihan Menu



Flowchart 1.2 Register



Flowchart 1.3 Login



Flowchart 1.4 Halaman Utama

2. Analisis Program

2.1 Deskripsi Singkat Program

Program ini bertujuan untuk mengelola data pegawai dalam suatu sistem. Dengan adanya fitur multiuser, setiap pengguna harus melakukan registrasi dan login sebelum dapat mengakses sistem. Program ini memungkinkan pengguna untuk menambahkan pegawai dengan mencatat informasi lengkap seperti nama, waktu masuk, waktu keluar, dan hasil kerja. Selain itu, pengguna juga dapat melihat daftar pegawai dengan tampilan yang rapi dalam format tabel, memperbarui informasi pegawai, serta menghapus data pegawai yang sudah tidak diperlukan. Sistem ini dirancang agar lebih aman dengan membatasi percobaan login hingga tiga kali untuk mencegah akses yang tidak sah. Dengan adanya fitur CRUD (Create, Read, Update, Delete), program ini mempermudah pengelolaan data pegawai secara efisien dan otomatis, sehingga tidak perlu mencatat secara manual.

3. Source Code

3.1 Registrasi Pengguna

Fitur ini memungkinkan pengguna untuk mendaftarkan akun dengan **Nama dan NIM**. Membatasi jumlah pengguna agar tidak melebihi MAX_USERS. Jika jumlah pengguna penuh, maka sistem akan memberi tahu pengguna.

```
if (pilihan == 1) {
            if (jumlah_user < MAX_USERS) {
                cout << "\n=== Register Akun ===\nMasukkan Nama: ";
                getline(cin, users[jumlah_user].username);
                cout << "Masukkan NIM: ";
                getline(cin, users[jumlah_user].nim);
                jumlah_user++;
                cout << "Registrasi berhasil! Silakan login.\n";
            } else {
                cout << "Jumlah user maksimal telah tercapai!\n";
            }
```

3.2 Login Pengguna

Pengguna memiliki 3 kali kesempatan login sebelum program berhenti. Sistem mencocokkan username dan NIM dengan data yang terdaftar. Jika sesuai, pengguna berhasil login. jika gagal 3 kali, maka program berhenti.

```
} else if (pilihan == 2) {
            string username, nim;
            int login_attempts = 0;
            while (login_attempts < 3) {
                cout << "\n=== Login ===\nMasukkan Nama: ";
                getline(cin, username);
                cout << "Masukkan NIM: ";
                getline(cin, nim);
                bool found = false;
                for (int i = 0; i < jumlah_user; i++) {
                    if (users[i].username == username && users[i].nim == nim) {
                        loggedInUser = username;
                        found = true;
                        break;
                    }
                }
                if (found) break;
                cout << "Login gagal! Coba lagi.\n";
```

```
                login_attempts++;
            }
            if (login_attempts == 3) {
                cout << "Anda telah gagal login sebanyak 3 kali. Program
berhenti.\n";
                return 0;
            }
            do {
```

3.3 Menambah Pegawai

Pengguna memasukkan semua informasi pegawai sekaligus (nama, waktu masuk, keluar, dan hasil kerja). Membatasi jumlah pegawai agar tidak melebihi MAX_PEGAWAI.

```
if (pilihan == 1) {
                    if (jumlah_pegawai < MAX_PEGAWAI) {
                        cout << "Masukkan Nama Pegawai: ";
                        getline(cin, pegawai[jumlah_pegawai].nama);
                        cout << "Masukkan Waktu Kehadiran: ";
                        getline(cin, pegawai[jumlah_pegawai].waktu_masuk);
                        cout << "Masukkan Waktu Keluar: ";
                        getline(cin, pegawai[jumlah_pegawai].waktu_keluar);
                        cout << "Masukkan Hasil Kerja: ";
                        getline(cin, pegawai[jumlah_pegawai].hasil_kerja);
                        jumlah_pegawai++;
                        cout << "Pegawai berhasil ditambahkan.\n";
                    } else {
                        cout << "Data pegawai penuh!\n";
                     }
```

3.4 Menampilkan Pegawai

Menampilkan daftar pegawai dengan format tabel rapi menggunakan setw(). Memudahkan pengguna untuk melihat data pegawai.

```
} else if (pilihan == 2) {
                    cout << "\nDaftar Pegawai:\n+----+----------------------+---
------------+----------------+----------------+\n";
                    cout << "| No | Nama                 | Waktu Masuk    |
Waktu Keluar   | Hasil Kerja    |\n";
                    cout << "+----+----------------------+----------------+-----
```

```
----------+----------------+\n";
                    for (int i = 0; i < jumlah_pegawai; i++) {
                        cout << "| " << setw(2) << i + 1 << " | " << setw(20) <<
pegawai[i].nama
                             << " | " << setw(14) << pegawai[i].waktu_masuk
                             << " | " << setw(14) << pegawai[i].waktu_keluar
                             << " | " << setw(14) << pegawai[i].hasil_kerja << "
|\n";
                    }
                    cout << "+----+----------------------+----------------+-----
----------+----------------+\n";
                }
```

3.5 Memperbarui atau Menghapus Pegawai

- Sebelum update atau hapus, daftar pegawai ditampilkan terlebih dahulu.
- Update Pegawai : Memungkinkan pengguna untuk mengubah semua data pegawai.
- Hapus Pegawai : Menggeser data ke atas setelah penghapusan.

```
else if (pilihan == 3 || pilihan == 4) {
                    if (jumlah_pegawai == 0) {
                        cout << "Tidak ada pegawai yang dapat diperbarui atau
dihapus!\n";
                        continue;
                    }
                    cout << "\nDaftar Pegawai:\n";
                    for (int i = 0; i < jumlah_pegawai; i++) {
                        cout << i + 1 << ". " << pegawai[i].nama << endl;
                    }
                    int index;
                    cout << "Masukkan nomor pegawai: ";
                    cin >> index;
                    cin.ignore();
                    if (index > 0 && index <= jumlah_pegawai) {
                        if (pilihan == 3) {
                            cout << "Masukkan Nama Pegawai: ";
                            getline(cin, pegawai[index - 1].nama);
                            cout << "Masukkan Waktu Kehadiran: ";
                            getline(cin, pegawai[index - 1].waktu_masuk);
                            cout << "Masukkan Waktu Keluar: ";
                            getline(cin, pegawai[index - 1].waktu_keluar);
                            cout << "Masukkan Hasil Kerja: ";
                            getline(cin, pegawai[index - 1].hasil_kerja);
                            cout << "Data pegawai diperbarui.\n";
                        } else {
                            for (int i = index - 1; i < jumlah_pegawai - 1; i++)
```

```
{
                                pegawai[i] = pegawai[i + 1];
                            }
                            jumlah_pegawai--;
                            cout << "Pegawai berhasil dihapus.\n";
                        }
                    } else {
                        cout << "Nomor pegawai tidak valid!\n";
                    }
                }
```

3.6 Program Berjalan Terus Menerus

Looping program terus berjalan hingga pengguna memilih logout (5) dan memastikan pengguna bisa kembali ke menu setelah melakukan operasi.

```
} while (pilihan != 5);
        } else if (pilihan == 3) {
            cout << "Program berhenti.\n";
            break;
        } else {
            cout << "Pilihan tidak valid!\n";
        }
    }
    return 0;
}
```

4. Uji Coba dan Hasil Output

```
=== Sistem Manajemen Pegawai ===
1. Register
2. Login
3. Keluar
Pilihan: 1

=== Register Akun ===
Masukkan Nama: Renaya
Masukkan NIM: 002
Registrasi berhasil! Silakan login.

=== Sistem Manajemen Pegawai ===
1. Register
2. Login
3. Keluar
Pilihan: 2

=== Login ===
Masukkan Nama: Renaya
Masukkan NIM: 002
```

Output 4.1 Register & Login

```
=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
```

Output 4.2 Tampilam Menu

```
Pilihan: 1
Masukkan Nama Pegawai: thor
Masukkan Waktu Kehadiran: 7.15
Masukkan Waktu Keluar: 17.20
Masukkan Hasil Kerja: 250.000
Pegawai berhasil ditambahkan.

=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 1
Masukkan Nama Pegawai: bruce
Masukkan Waktu Kehadiran: 7.20
Masukkan Waktu Keluar: 18.05
Masukkan Hasil Kerja: 250.000
Pegawai berhasil ditambahkan.

=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 1
Masukkan Nama Pegawai: clint
Masukkan Waktu Kehadiran: 8.00
Masukkan Waktu Keluar: 18.10
Masukkan Hasil Kerja: 280.000
Pegawai berhasil ditambahkan.
```

Output 4.3 Tambah Pegawai (Create)

```
=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 2

Daftar Pegawai:
+----+---------------------+---------------+---------------+---------------+
| No | Nama                | Waktu Masuk   | Waktu Keluar  | Hasil Kerja   |
+----+---------------------+---------------+---------------+---------------+
|  1 |                thor |          7.15 |         17.20 |       250.000 |
|  2 |               bruce |          7.20 |         18.05 |       250.000 |
|  3 |               clint |          8.00 |         18.10 |       280.000 |
+----+---------------------+---------------+---------------+---------------+
```

Output 4.4 Menampilkan Pegawai (Read)

```
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 3

Daftar Pegawai:
1. thor
2. bruce
3. clint
Masukkan nomor pegawai: 1
Masukkan Nama Pegawai: thor
Masukkan Waktu Kehadiran: 7.10
Masukkan Waktu Keluar: 17.00
Masukkan Hasil Kerja: 200.000
Data pegawai diperbarui.

=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 2

Daftar Pegawai:
+----+---------------------+---------------+---------------+---------------+
| No | Nama                | Waktu Masuk   | Waktu Keluar  | Hasil Kerja   |
+----+---------------------+---------------+---------------+---------------+
|  1 |                thor |          7.10 |         17.00 |       200.000 |
|  2 |               bruce |          7.20 |         18.05 |       250.000 |
|  3 |               clint |          8.00 |         18.10 |       280.000 |
+----+---------------------+---------------+---------------+---------------+
```

Output 4.5 Update Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 4

Daftar Pegawai:
1. thor
2. bruce
3. clint
Masukkan nomor pegawai: 2
Pegawai berhasil dihapus.

=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 2

Daftar Pegawai:
+----+---------------------+---------------+---------------+---------------+
| No | Nama                | Waktu Masuk   | Waktu Keluar  | Hasil Kerja   |
+----+---------------------+---------------+---------------+---------------+
|  1 |                thor |          7.10 |         17.00 |       200.000 |
|  2 |               clint |          8.00 |         18.10 |       280.000 |
+----+---------------------+---------------+---------------+---------------+
```

Output 4.6 Delete Pegawai

```
=== Menu Manajemen Pegawai ===
1. Tambah Pegawai
2. Tampilkan Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Logout
Pilihan: 5

=== Sistem Manajemen Pegawai ===
1. Register
2. Login
3. Keluar
Pilihan: 3
Program berhenti.
```

Output 4.7 Keluar Program & Berhenti

5. GIT

1. Git Init adalah perintah Git yang digunakan untuk menginisialisasi repository Git baru dalam sebuah folder. Cukup ketik "git init" pada terminal VSCode

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git init
Reinitialized existing Git repository in C:/Users/LENOVO/Downloads/Praktikum-apl/.git/
```

Gambar 5.1 Git Init

2. Git commit Git adalah perintah Git yang digunakan untuk menyimpan perubahan yang telah ditambahkan ke staging area ke dalam repository. Setiap commit akan memiliki hash unik, pesan commit, dan menyimpan snapshot dari perubahan yang dilakukan.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git commit -m "Finish Post Test 3"
[detached HEAD 021384a] Finish Post Test 3
 11 files changed, 410 insertions(+), 2 deletions(-)
 create mode 100644 kelas/pertemuan-3/pertemuan3.cpp
 delete mode 160000 post-test
 delete mode 160000 post-test-1
 create mode 100644 post-test/post-test1/2409106002-RenayaPutriAlika-PT-1.cpp
 create mode 100644 post-test/post-test1/2409106002-RenayaPutriAlika-PT-1.exe
 create mode 100644 post-test/post-test1/2409106002-RenayaPutriAlika-PT-1.pdf
 rename {post-test-2 => post-test/post-test2}/2409106002-RenayaPutriAlika-PT-2.cpp (100%)
 rename {post-test-2 => post-test/post-test2}/2409106002-RenayaPutriAlika-PT-2.exe (100%)
 rename {post-test-2 => post-test/post-test2}/2409106002-RenayaPutriAlika-PT-2.pdf (100%)
 create mode 100644 post-test/post-test3/2409106002-RenayaPutriAlika-PT-3.cpp
 create mode 100644 post-test/post-test3/2409106002-RenayaPutriAlika-PT-3.exe
```

Gambar 5.2 Git Commit

3. Git remote adalah perintah Git yang digunakan untuk menghubungkan repository lokal dengan repository remote (misalnya di GitHub, GitLab, atau Bitbucket). Dengan perintah ini, kita bisa mengelola koneksi ke repository jarak jauh, memungkinkan push, pull, dan fetch dari repository tersebut.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git remote set-url origin https://github.com/Renaya0207/praktikum-apl.git
```

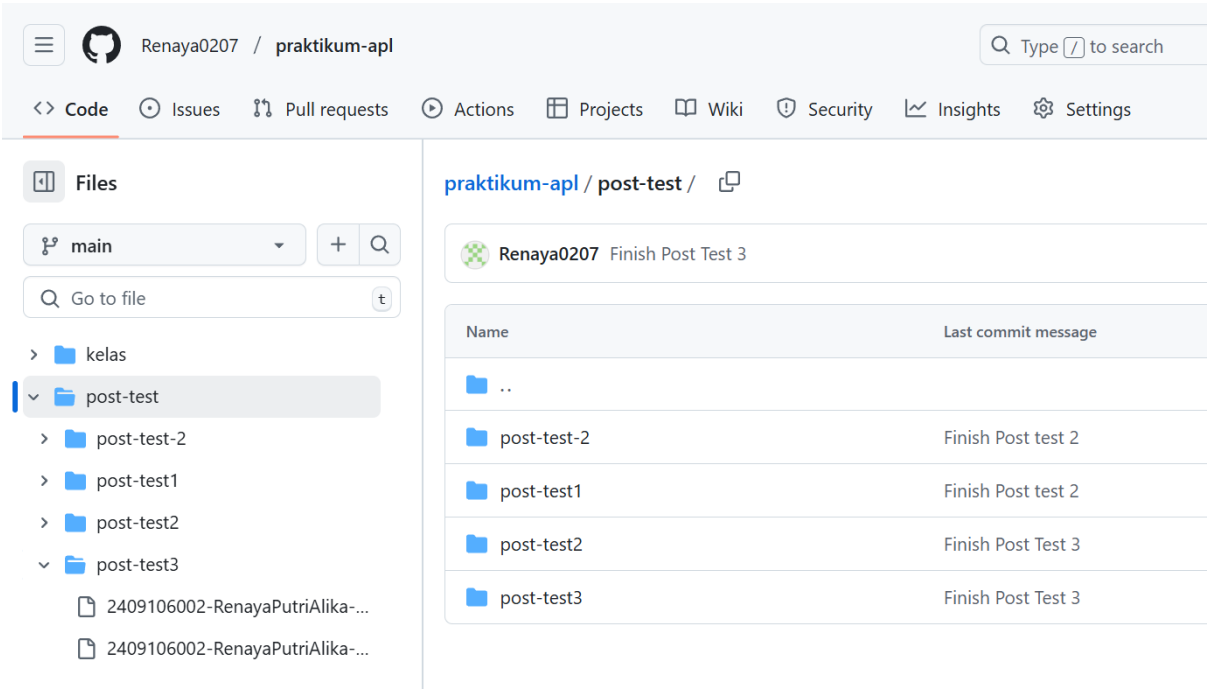
Gambar 5.3 Git Remote

4. Git Push adalah perintah Git yang digunakan untuk mengirim (mengunggah) perubahan dari repository lokal ke repository remote (seperti GitHub, GitLab, atau Bitbucket). Perintah ini memastikan bahwa perubahan yang sudah dikomit di lokal tersedia di repository jarak jauh sehingga bisa diakses oleh orang lain atau untuk cadangan.

```
PS C:\Users\LENOVO\Downloads\Praktikum-apl> git push origin main
>>
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 677.85 KiB | 3.85 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Renaya0207/praktikum-apl.git
   280ff4f..828add4  main -> main
PS C:\Users\LENOVO\Downloads\Praktikum-apl>
```

Gambar 5.4 Git Push

5. Berhasil di Up



Gambar 5.5 Berhasil