LAPORAN PRAKTIKUM

**POSTTEST 6**

**ALGORITMA PEMROGRAMAN LANJUT**



**Disusun oleh:**

**Ghesya Rhegyta Al Rachman (2409106023)**

**Kelas (A2‘24)**

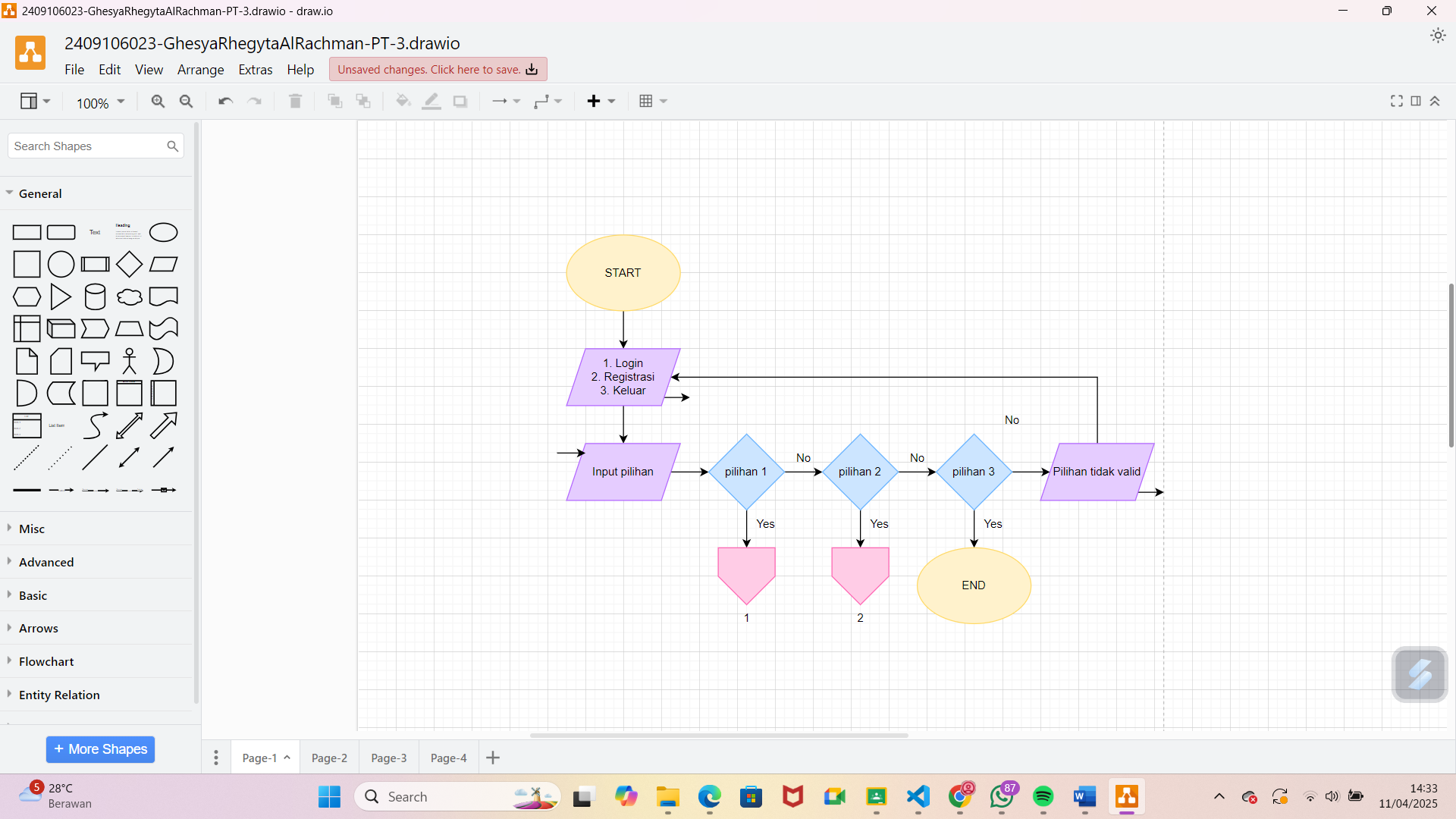
**PROGRAM STUDI INFORMATIKA**

**UNIVERSITAS MULAWARMAN**

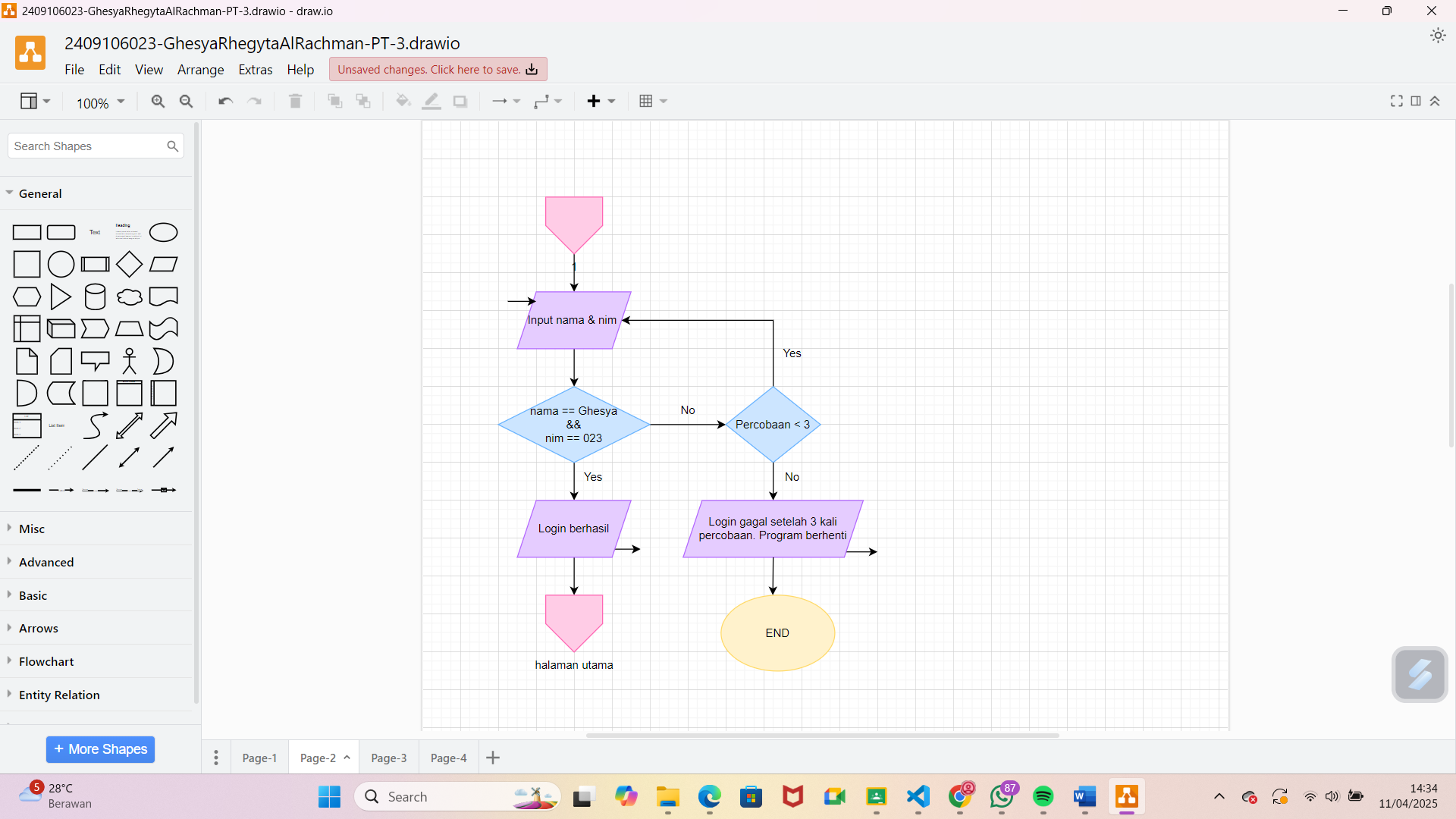
**SAMARINDA**

**2025**

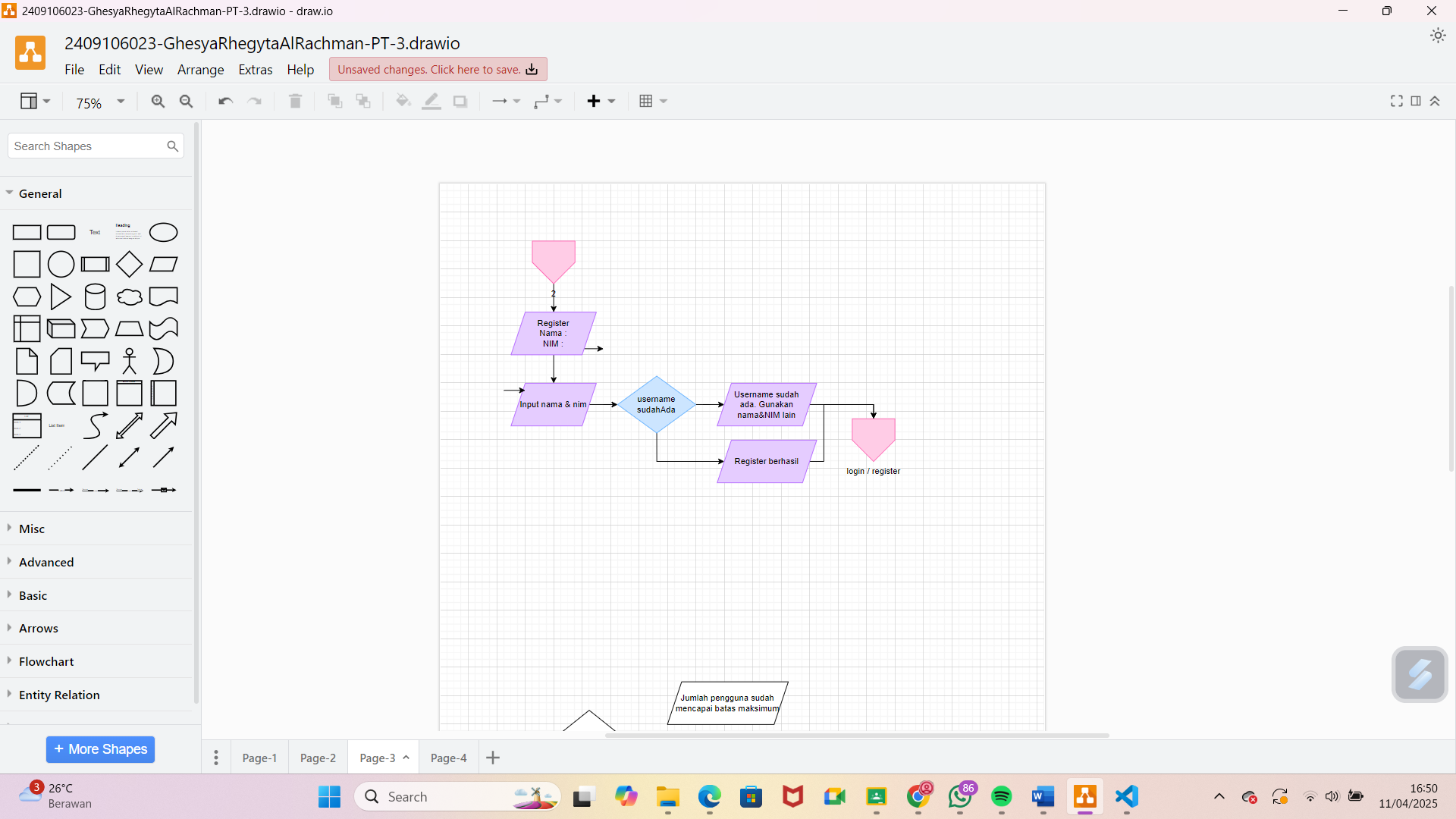
# Flowchart



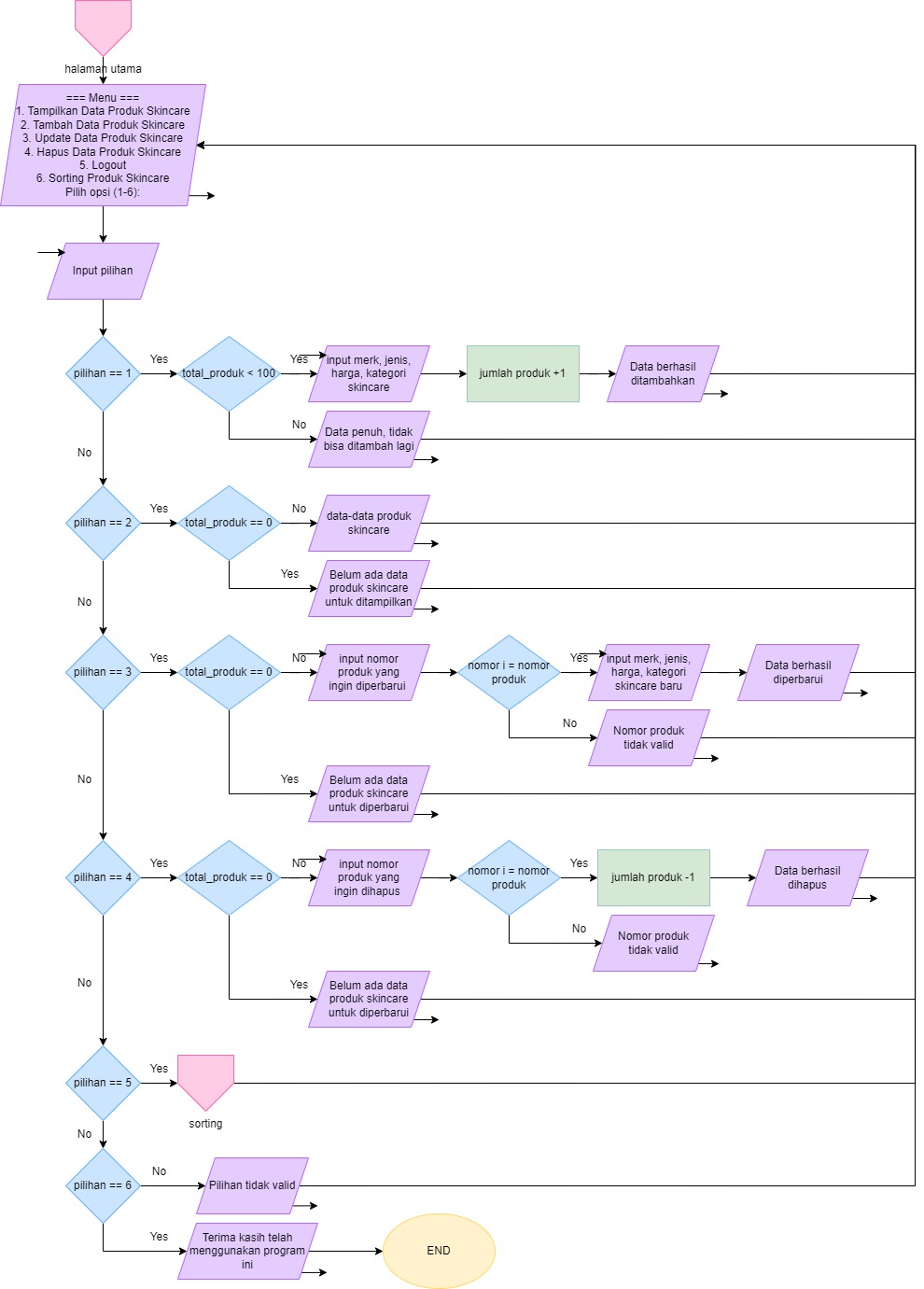
Gambar 1.1 Flowchart Pilihan Menu



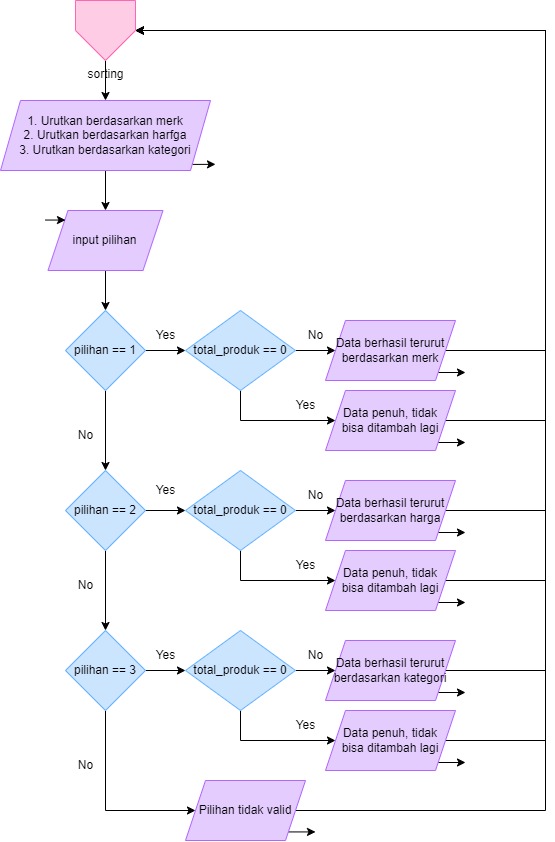
Gambar 1.2 Flowchart Login



Gambar 1.3 Flowchart Register



Gambar 1.4 Flowchart Halaman Utama



Gambar 1.5 Flowchart Sorting

1. **Analisis Program**
   1. **Deskripsi Singkat Program**

**Tujuan :**

Program ini bertujuan untuk membantu pengelolaan data produk skincare. Dengan adanya fitur multiuser, pengguna harus login atau registrasi sebelum mengakses program ini. Pengguna yang telah berhasil login dapat menambah, melihat, memperbarui, serta menghapus data produk skincare. Program ini dapat membantu bisnis kecil untuk mencatat produk skincare.

**Manfaat :**

Perulangan – Program terus berjalan hingga pengguna memilih untuk berhenti

Keamanan pengguna – Hanya pengguna yang terdaftar dan berhasil login yang dapat mengakses program tersebut

Sistem validasi – Mencegah duplikasi akun dan membatasi percobaan login

Mudah digunakan – Dengan adanya menu yang jelas, pengguna dapat mengelola data skincare tanpa kesulitan

Pencatatan – Membantu pengguna menyimpan dan mengatur data produk

Sorting data – Mengurutkan data skincare

1. **Source Code**
   1. **Pilihan Menu**

Pengguna memilih opsi 1 atau 2 untuk mengakses program. Pengguna dapat memilih opsi 3 untuk keluar.

|  |
| --- |
| cout << "\n=== LOGIN / REGISTER ===" << endl;  cout << "1. Login" << endl;  cout << "2. Register" << endl;  cout << "3. Keluar dari program" << endl;  cout << "Pilih opsi (1-3): ";  int pilihan;  cin >> pilihan;  cin.ignore(); |

Gambar 3.1 Pilihan Menu

**3.2 Login Pengguna**

Memverifikasi nama dan 3 digit terakhir NIM dengan batas maksimal 3 kali percobaan. Jika gagal, maka program akan berhenti.

|  |
| --- |
| if (pilihan == 1) {  while (attempts < 3) {  cout << "\nMasukkan nama: ";  getline(cin, nama);  cout << "Masukkan 3 digit terakhir NIM: ";  getline(cin, nim);   int login\_index = -1;  for (int i = 0; i < total\_user; ++i) {  if (users[i].nama == nama && users[i].nim == nim) {  login\_index = i;  break;  }  }   if (login\_index != -1) {  cout << "Login berhasil! Awooo " << nama << "!" << endl |

|  |
| --- |
| } else {  attempts++;  cout << "Login gagal! Percobaan ke-" << attempts << " dari 3" << endl;  if (attempts == 3) {  cout << "Login gagal setelah 3 kali percobaan:b Program berhenti wlee" << endl;  return 0;  }  } |

Gambar 3.2 Login Pengguna

**3.3 Register Pengguna**

Mendaftar akun dengan input nama dan nim. Jika nama dan NIM yang diregistrasikan sudah ada, maka sistem akan memberi output “Username sudah ada, gunakan nama atau NIM lain”. Program ini juga membatasi jumlah pengguna agar tidak melebihi batas yang ditentukan. Jika jumlah pengguna sudah maksimal, maka sistem akan memberi output “Jumlah pengguna sudah mencapai batas maksimum.”

|  |
| --- |
| else if (pilihan == 2) {  if (total\_user < MAX\_USER) {  cout << "\nMasukkan nama: ";  getline(cin, nama);  cout << "Masukkan 3 digit terakhir NIM: ";  getline(cin, nim);   bool sudahAda = false;  for (int i = 0; i < total\_user; ++i) {  if (users[i].nama == nama && users[i].nim == nim) {  sudahAda = true;  break;  }  }   if (sudahAda) {  cout << "Username sudah ada:b gunakan nama atau NIM lain dek" << endl;  } else {  users[total\_user++] = {nama, nim};  cout << "Registrasi berhasil!\n" << endl;  }    } else {  cout << "Yahhh jumlah pengguna sudah maksimal" << endl;  }  } |

Gambar 3.3 Register Pengguna

**3.4 Menu Utama**

Menampilkan pilihan menu utama produk skincare

|  |
| --- |
| while (true) {  cout << "\n++-------------------------------------++" << endl;  cout << "|| === Menu === ||" << endl;  cout << "++-------------------------------------++" << endl;  cout << "|| 1. Tambah Data Produk Skincare ||" << endl;  cout << "|| 2. Tampilkan Data Produk Skincare ||" << endl;  cout << "|| 3. Perbarui Data Produk Skincare ||" << endl;  cout << "|| 4. Hapus Data Produk Skincare ||" << endl;  cout << "|| 5. Urutkan Data Produk Skincare ||" << endl;  cout << "|| 6. Logout ||" << endl;  cout << "++-------------------------------------++" << endl;  cout << "Pilih opsi (1-6): ";  int pilihan\_menu;  cin >> pilihan\_menu;  cin.ignore(); |

Gambar 3.4 Menu Utama Produk Skincare

**3.5 CRUD Manajemen Produk Skincare**

Fitur CRUD dalam program ini yaitu Create (membuat atau menambahkan data), Read (menampilkan data yang tersimpan), Update (memperbarui data yang sudah ada), dan Delete (menghapus data yang sudah tidak diperlukan)

|  |
| --- |
| void tambahProduk(Skincare \*produk, int \*total\_produk) {  if (\*total\_produk < 100) {  cout << "Masukkan merk skincare: ";  getline(cin, produk[\*total\_produk].merk);  cout << "Masukkan jenis skincare: ";  getline(cin, produk[\*total\_produk].jenis);  cout << "Masukkan harga skincare: ";  cin >> produk[\*total\_produk].harga;  cin.ignore();  cout << "Masukkan kategori skincare (Face/Body/Hair): ";  getline(cin, produk[\*total\_produk].kategori.nama);  (\*total\_produk)++;  cout << "Data berhasil ditambahkan!" << endl;  } else {  cout << "Jumlah data sudah penuh!" << endl;  } }  void tampilkanProduk(Skincare \*produk, int total\_produk) {  if (total\_produk == 0) {  cout << "Belum ada data produk skincare." << endl;  } else {  cout << "\n=== Data Produk Skincare ===" << endl;  cout << "---+--------------+---------------+-----------+-------------" << endl;  cout << " | Merk | Jenis | Harga | Kategori " << endl;  cout << "---+--------------+---------------+-----------+-------------" << endl;  for (int i = 0; i < total\_produk; ++i) {  cout << setw(2) << i + 1 << " | " << setw(12) << produk[i].merk  << " | " << setw(13) << produk[i].jenis  << " | " << setw(9) << produk[i].harga  << " | " << setw(10) << produk[i].kategori.nama << endl;  }  cout << "---+--------------+---------------+-----------+-------------" << endl;  } }  void perbaruiProduk(Skincare \*produk, int total\_produk) {  tampilkanProduk(produk, total\_produk);  int no;  cout << "Masukkan nomor produk yang ingin diperbarui: ";  cin >> no;  cin.ignore();  if (no < 1 || no > total\_produk) {  cout << "Nomor produk tidak valid" << endl;  } else {  cout << "Masukkan merk baru: ";  getline(cin, produk[no - 1].merk);  cout << "Masukkan jenis baru: ";  getline(cin, produk[no - 1].jenis);  cout << "Masukkan harga baru: ";  cin >> produk[no - 1].harga;  cin.ignore();  cout << "Masukkan kategori baru: ";  getline(cin, produk[no - 1].kategori.nama);  cout << "Data berhasil diperbarui!" << endl;  } }  void hapusProduk(Skincare \*produk, int \*total\_produk) {  tampilkanProduk(produk, \*total\_produk);  int no;  cout << "Masukkan nomor produk yang ingin dihapus: ";  cin >> no;  cin.ignore();  if (no < 1 || no > \*total\_produk) {  cout << "Nomor produk tidak valid" << endl;  } else {  for (int i = no - 1; i < \*total\_produk - 1; ++i) {  produk[i] = produk[i + 1];  }  (\*total\_produk)--;  cout << "Data berhasil dihapus!" << endl;  } } |

Gambar 3.5 CRUD Manajemen Produk Skincare

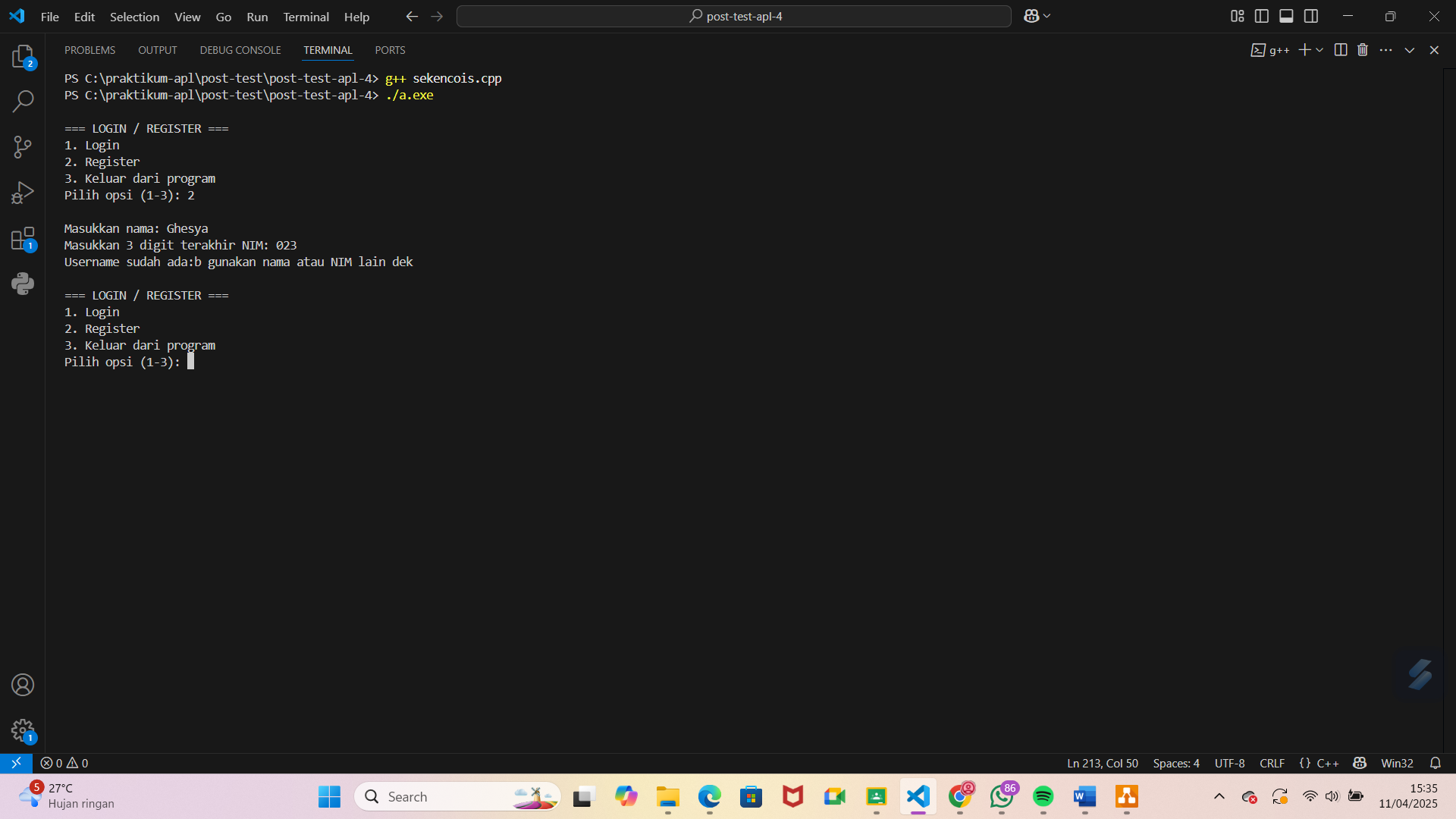
**3.6 Sorting Produk Skincare**

Metode sorting yang digunakan dalam program ini adalah sorting huruf (merk) secara ascending dengan bubble sort, sorting angka (harga) secara descending dengan selection sort, dan sorting huruf (kategori) secara ascending dengan merge sort

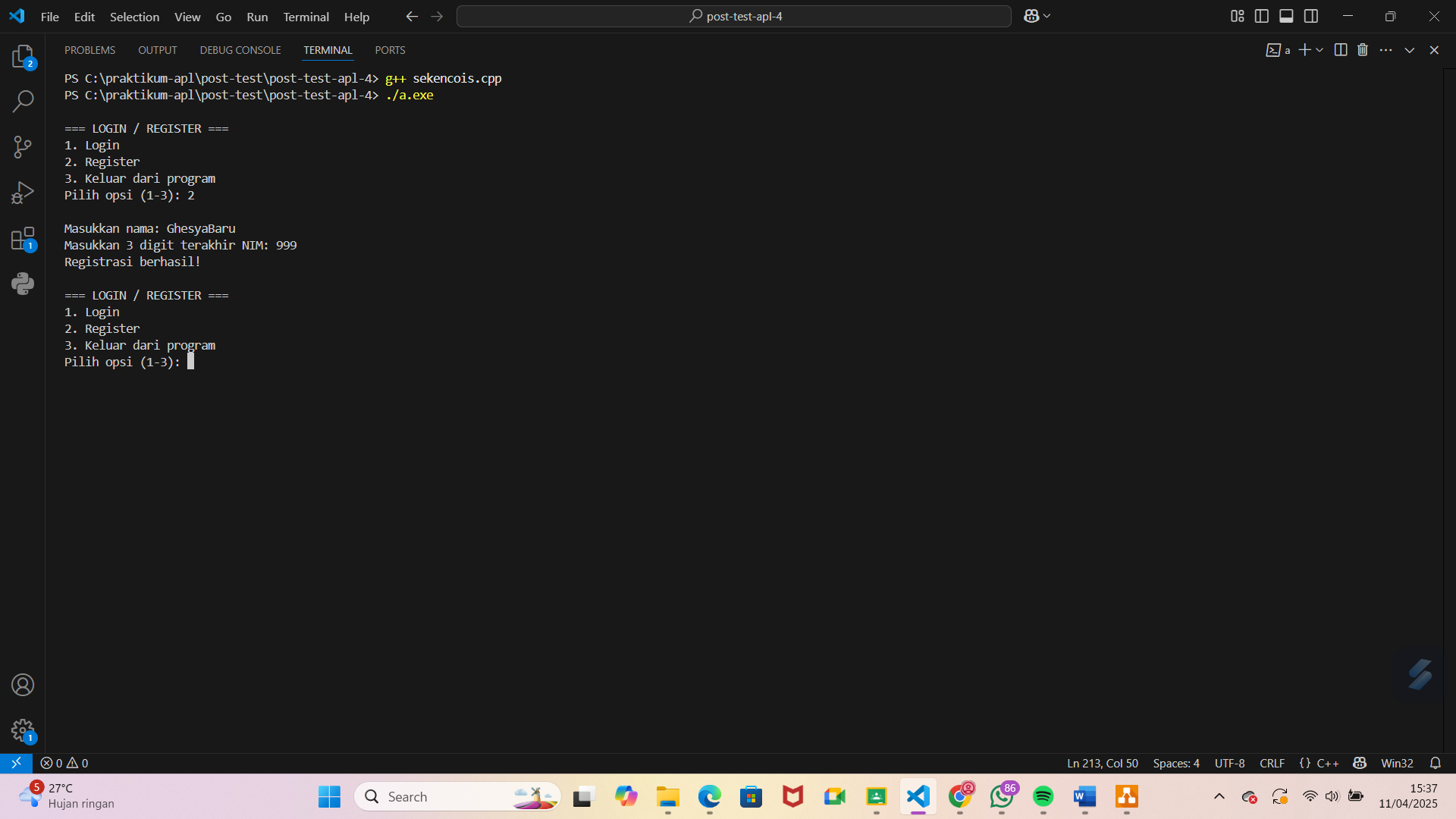
|  |
| --- |
| void bubbleSortMerk(Skincare \*produk, int total\_produk) {  for (int i = 0; i < total\_produk - 1; ++i) {  for (int j = 0; j < total\_produk - i - 1; ++j) {  if (produk[j].merk > produk[j + 1].merk) {  swap(produk[j], produk[j + 1]);  }  }  } }  void selectionSortHarga(Skincare \*produk, int total\_produk) {  for (int i = 0; i < total\_produk - 1; ++i) {  int max\_idx = i;  for (int j = i + 1; j < total\_produk; ++j) {  if (produk[j].harga > produk[max\_idx].harga) {  max\_idx = j;  }  }  swap(produk[i], produk[max\_idx]);  } }  void merge(Skincare \*produk, int left, int mid, int right) {  int n1 = mid - left + 1;  int n2 = right - mid;  Skincare L[n1], R[n2];   for (int i = 0; i < n1; ++i) L[i] = produk[left + i];  for (int j = 0; j < n2; ++j) R[j] = produk[mid + 1 + j];   int i = 0, j = 0, k = left;   while (i < n1 && j < n2) {  if (L[i].kategori.nama <= R[j].kategori.nama) {  produk[k] = L[i];  i++;  } else {  produk[k] = R[j];  j++;  }  k++;  }   while (i < n1) {  produk[k++] = L[i++];  }  while (j < n2) {  produk[k++] = R[j++];  } }  void mergeSortKategori(Skincare \*produk, int left, int right) {  if (left < right) {  int mid = left + (right - left) / 2;  mergeSortKategori(produk, left, mid);  mergeSortKategori(produk, mid + 1, right);  merge(produk, left, mid, right);  } }  void menuSorting(Skincare \*produk, int total\_produk) {  if (total\_produk == 0) {  cout << "Belum ada produk yang dapat diurutkan" << endl;  return;  } |

Gambar 3.6 Sorting Produk Skincare

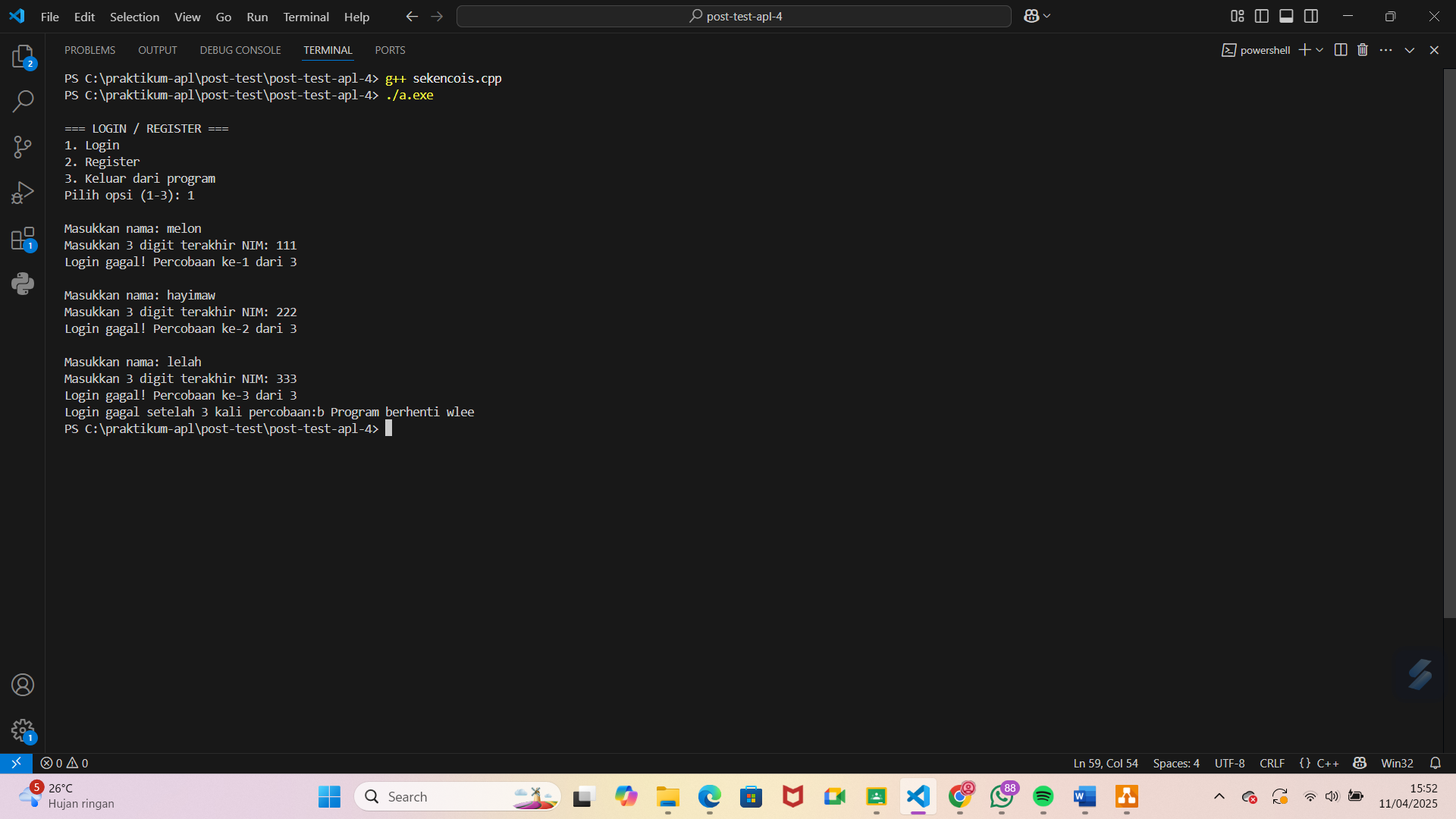
1. **Uji Coba dan Hasil Output**



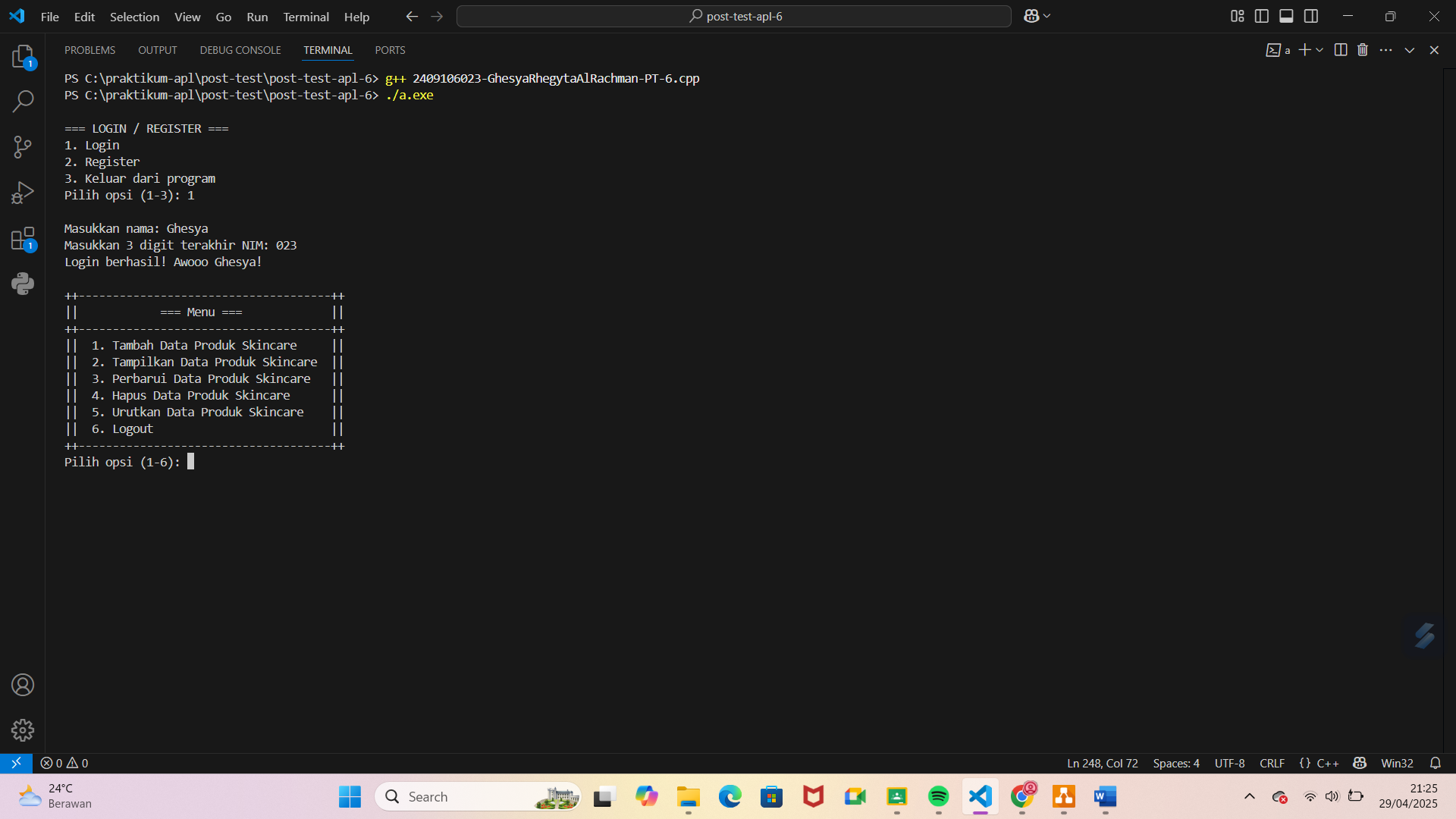
Gambar 4.1 Register gagal



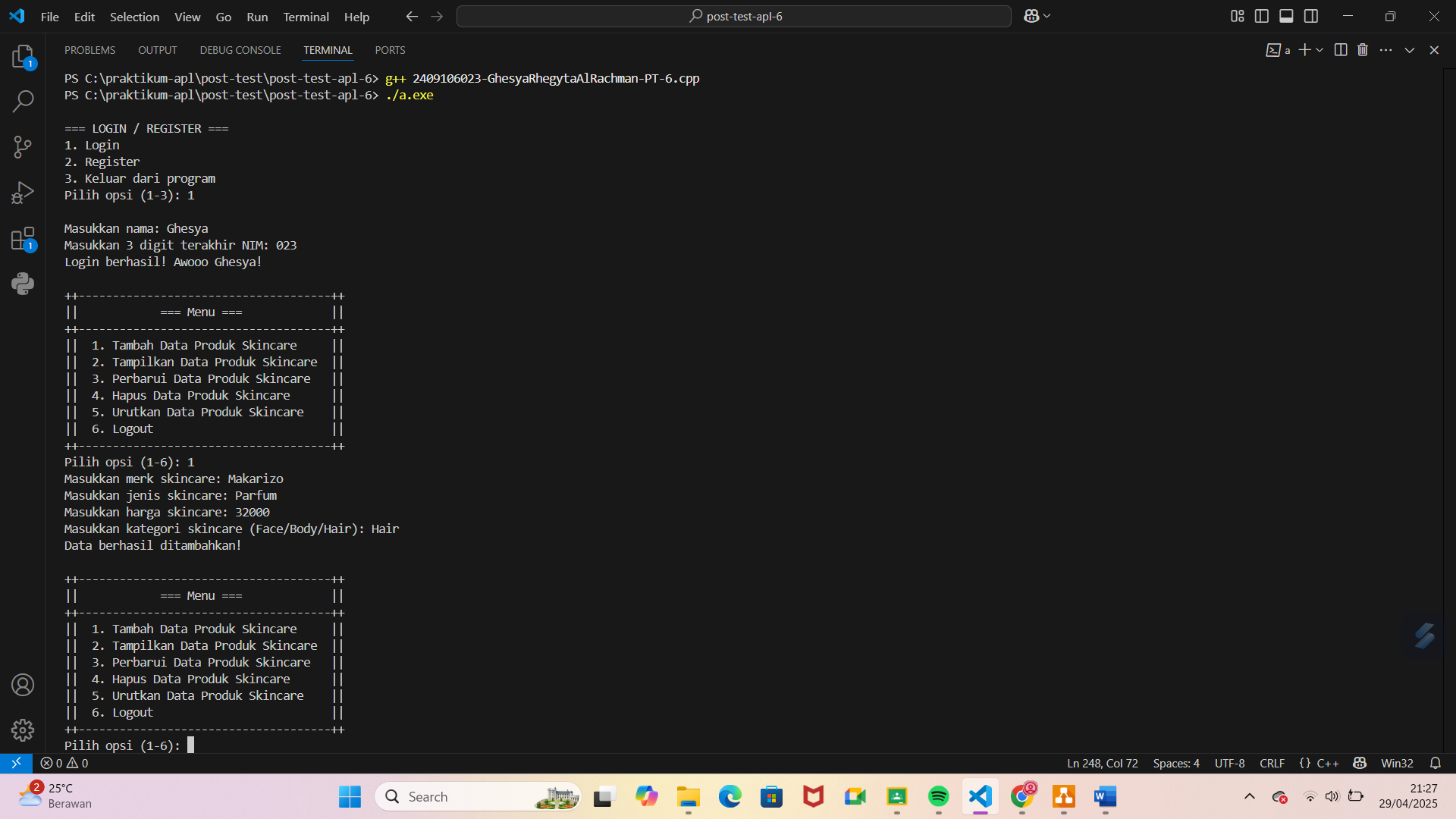
Gambar 4.2 Register berhasil



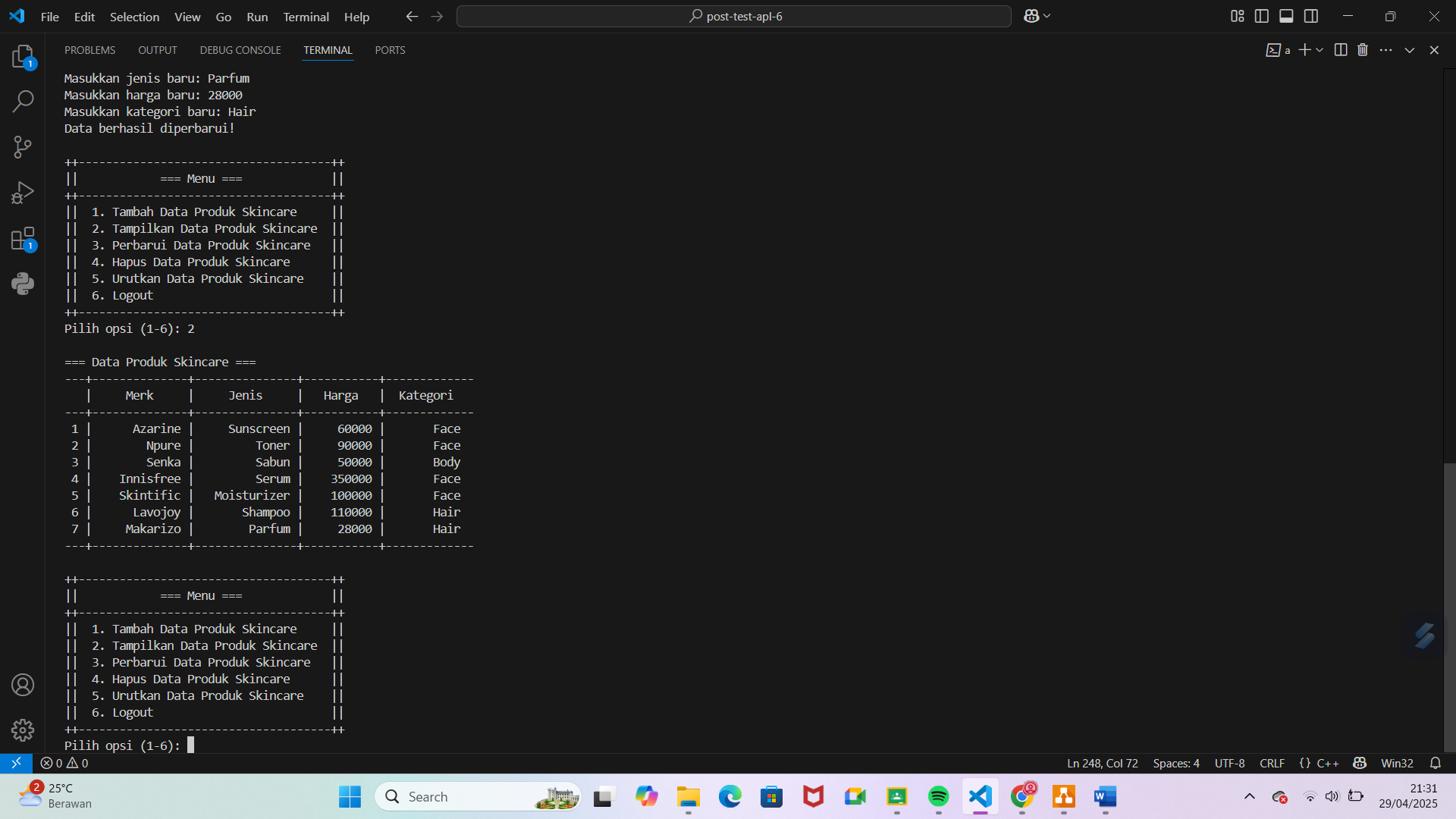
Gambar 4.3 Login gagal dan program berhenti



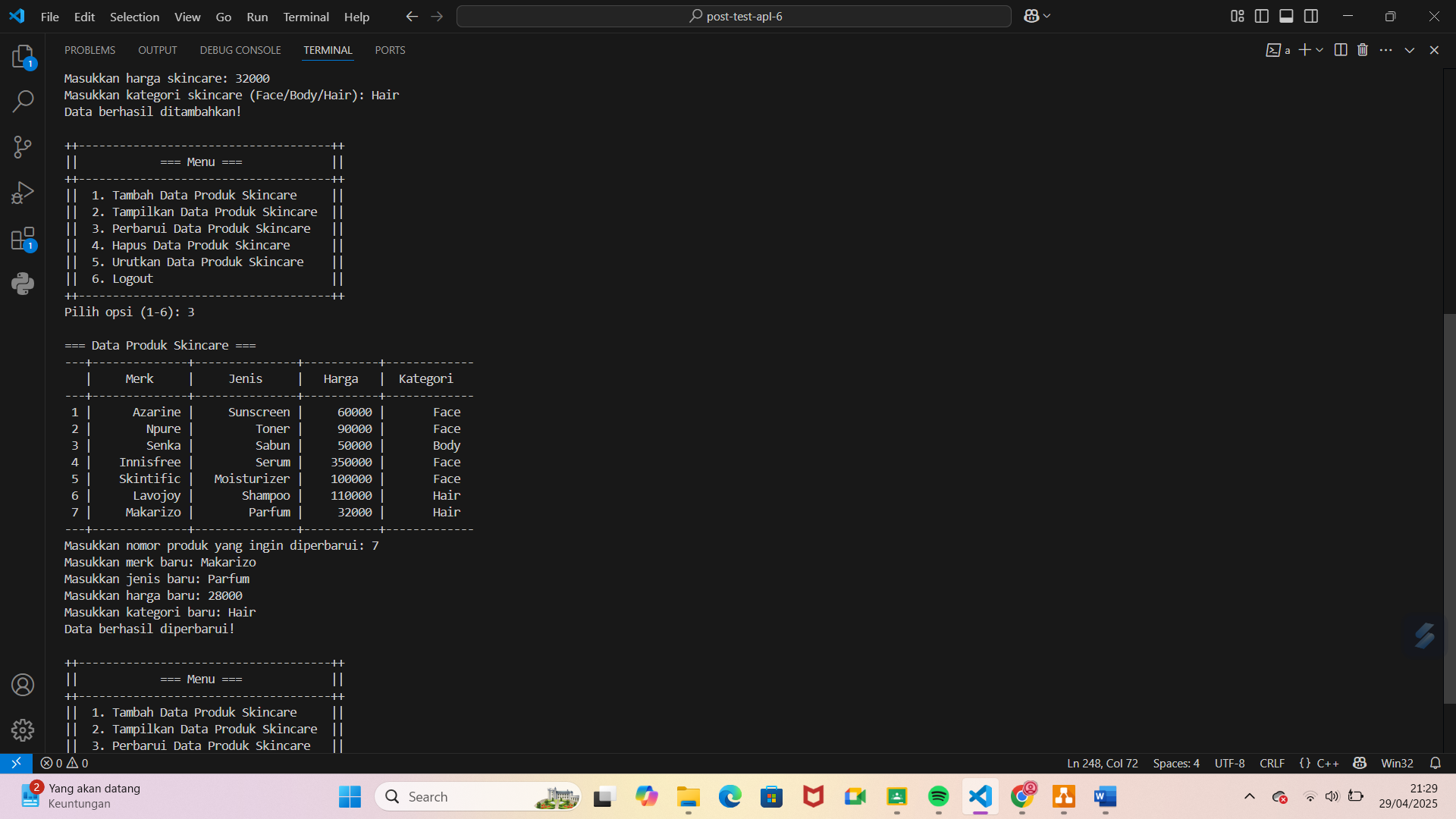
Gambar 4.4 Login berhasil dan Tampilan halaman utama



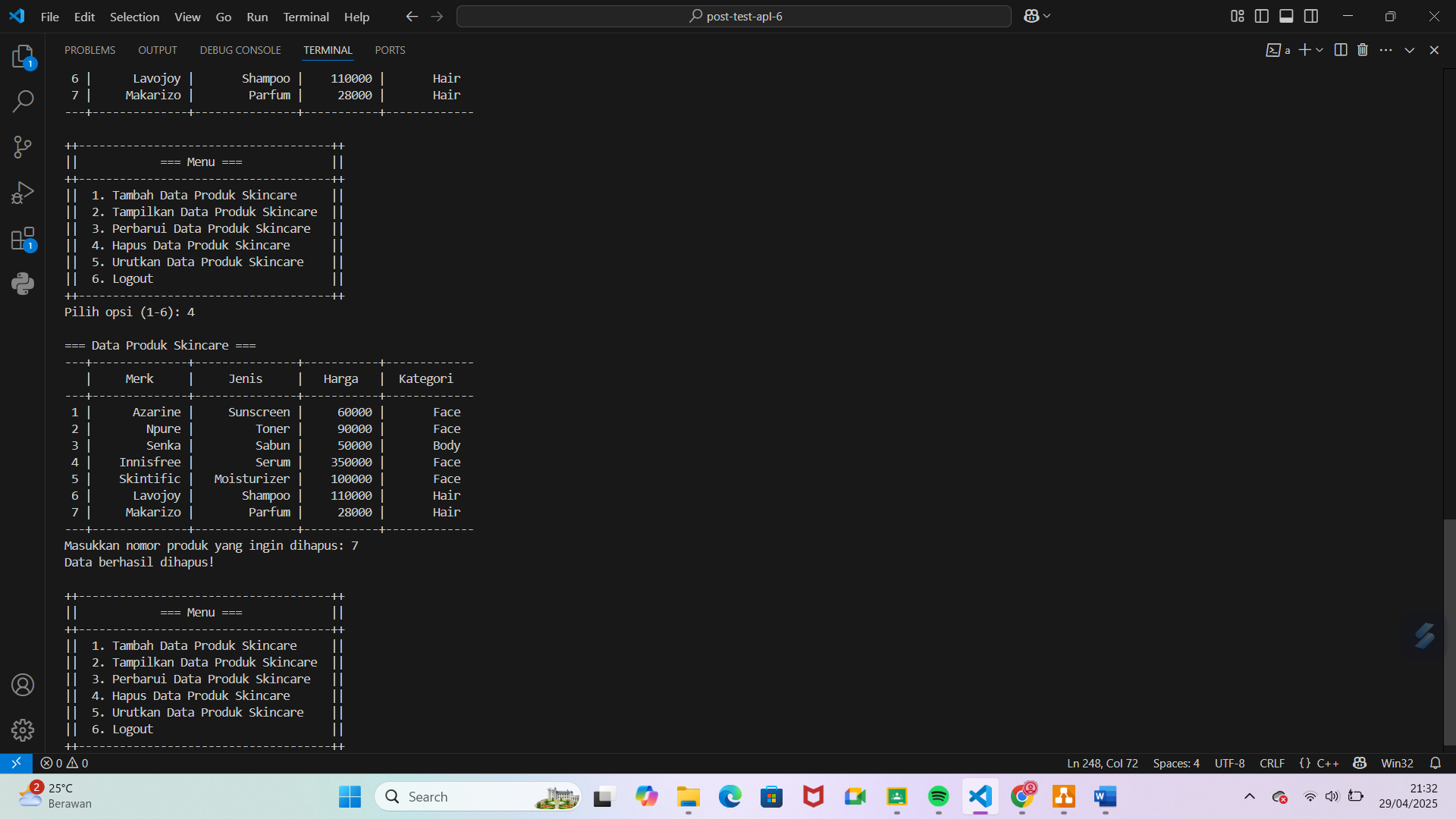
Gambar 4.5 Tambah produk skincare



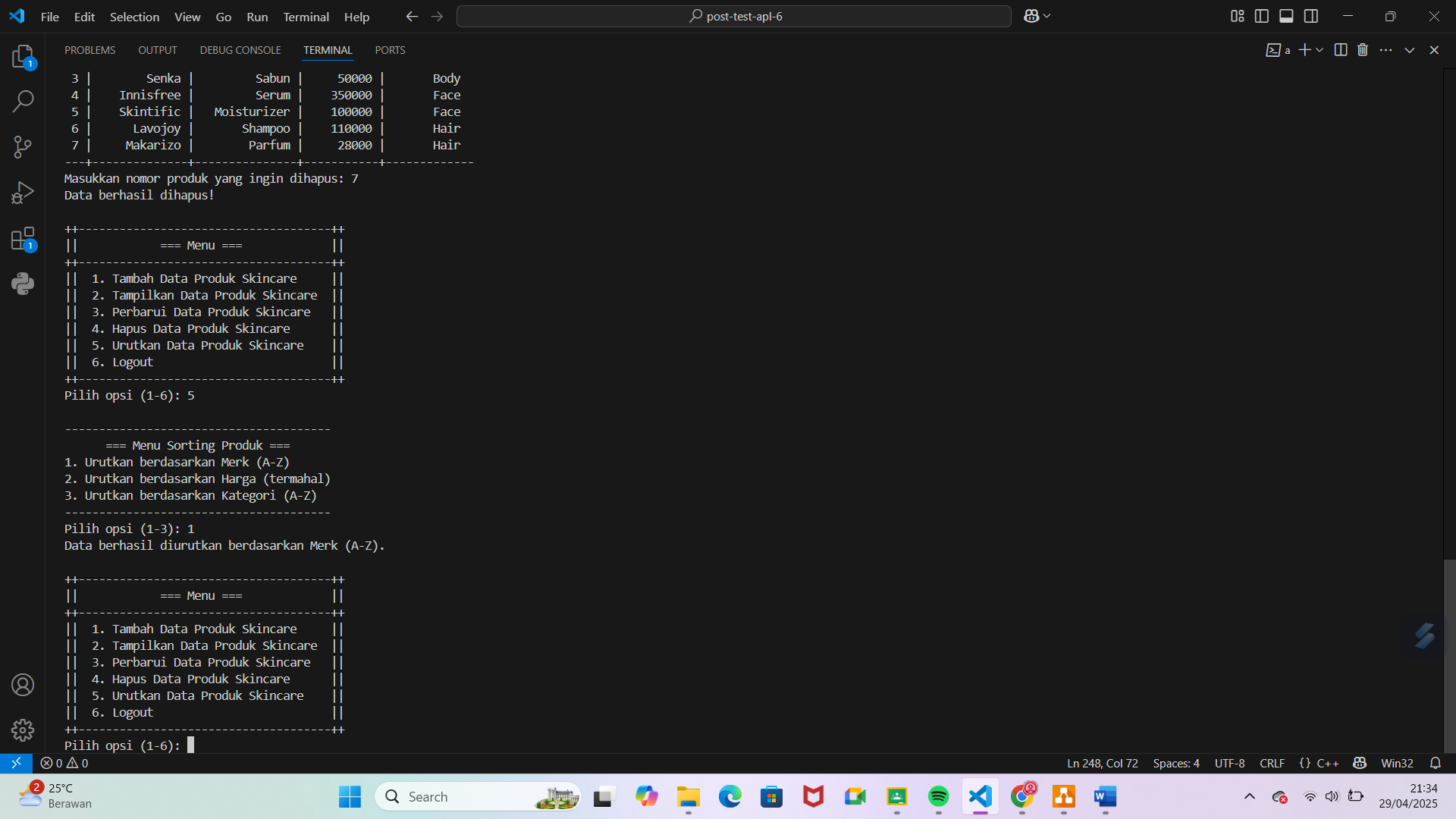
Gambar 4.6 Tampilkan produk skincare



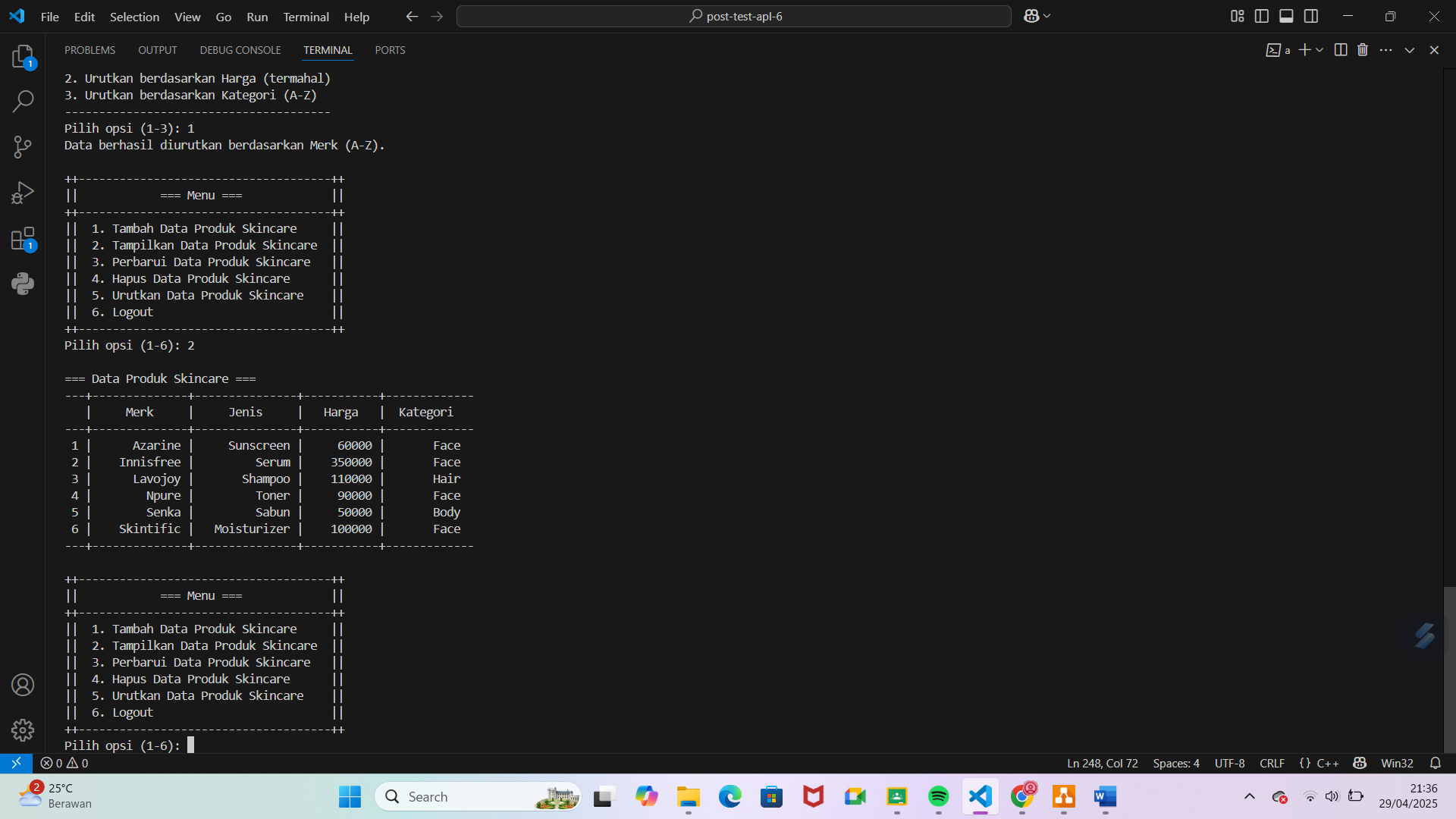
Gambar 4.7 Perbarui produk skincare



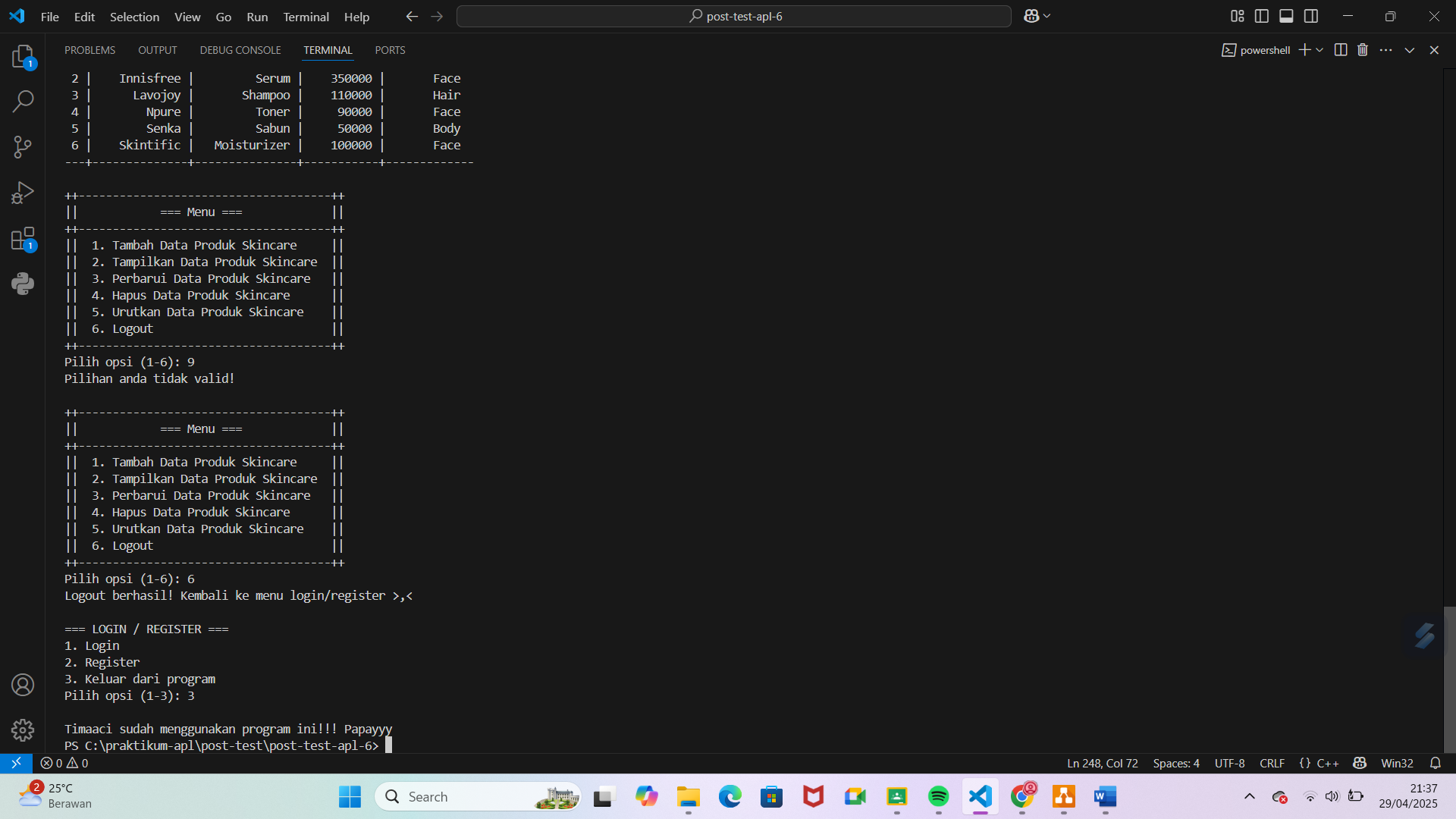
Gambar 4.8 Hapus produk skincare



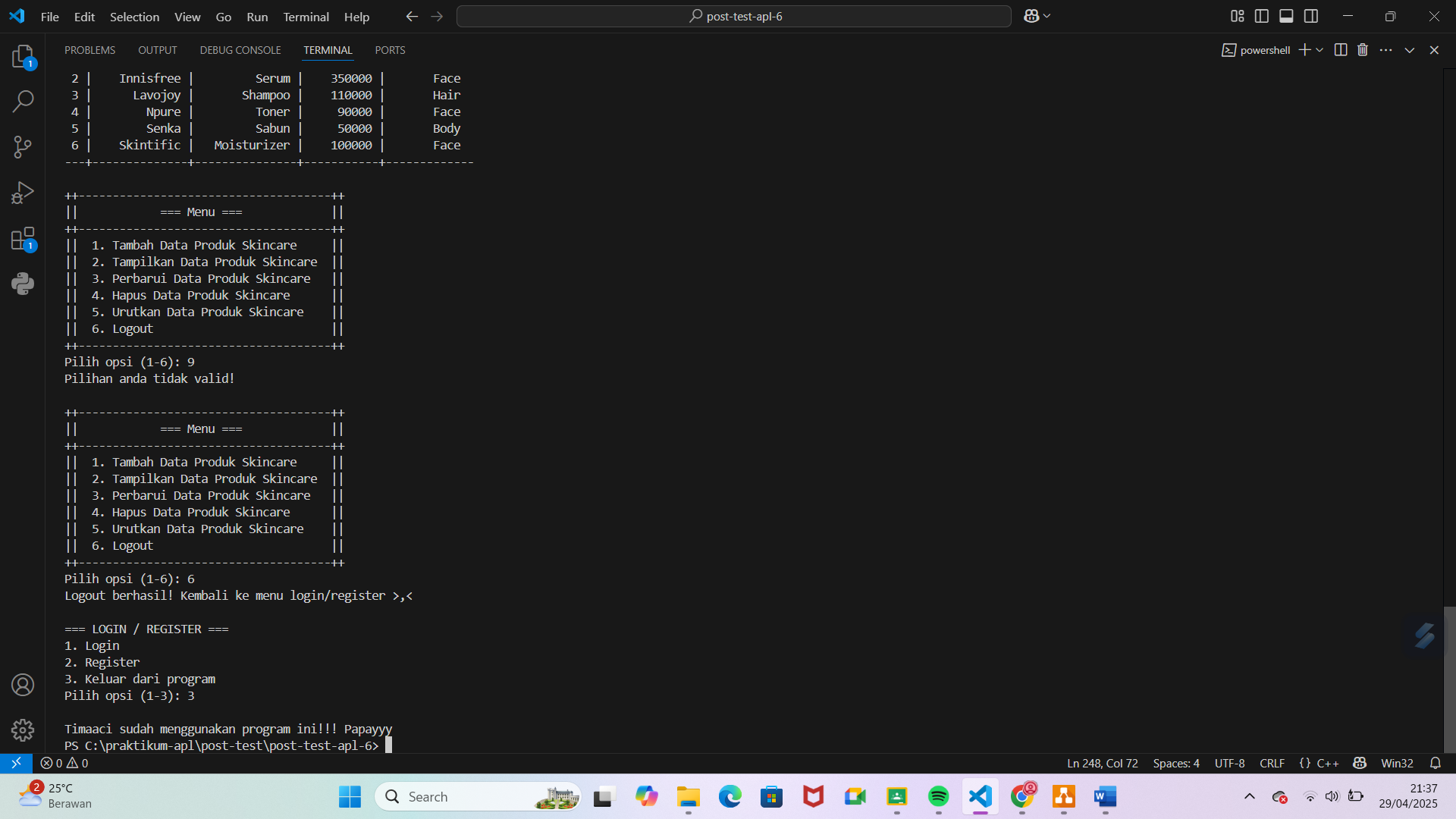
Gambar 4.9 Urutkan produk skincare



Gambar 4.10 Hasil mengurutkan berdasarkan merk



Gambar 4.11 Input nomor diluar opsi

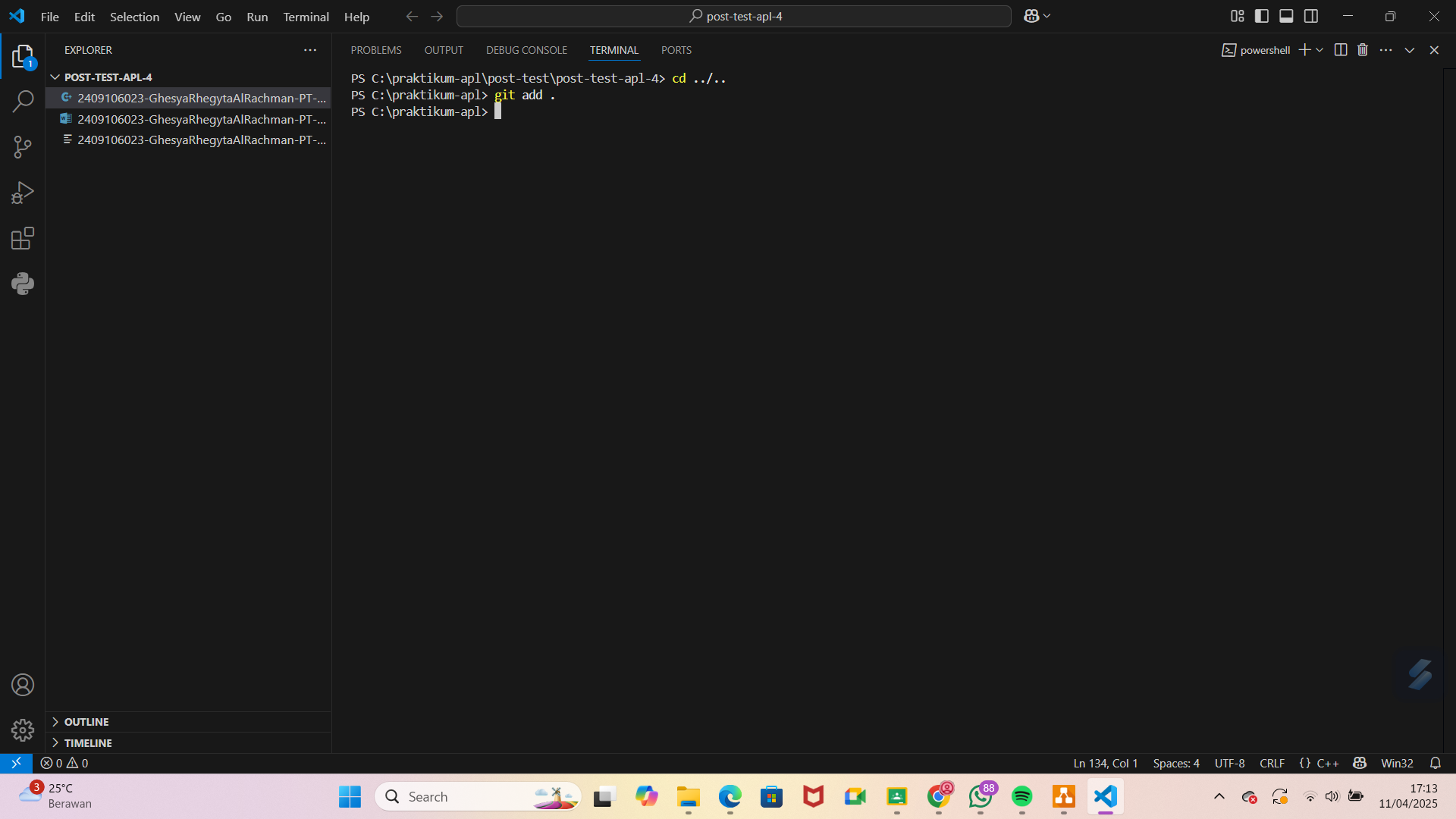


Gambar 4.12 Logout dan Keluar dari program

1. **Langkah-Langkah GIT**

1. Git Add .

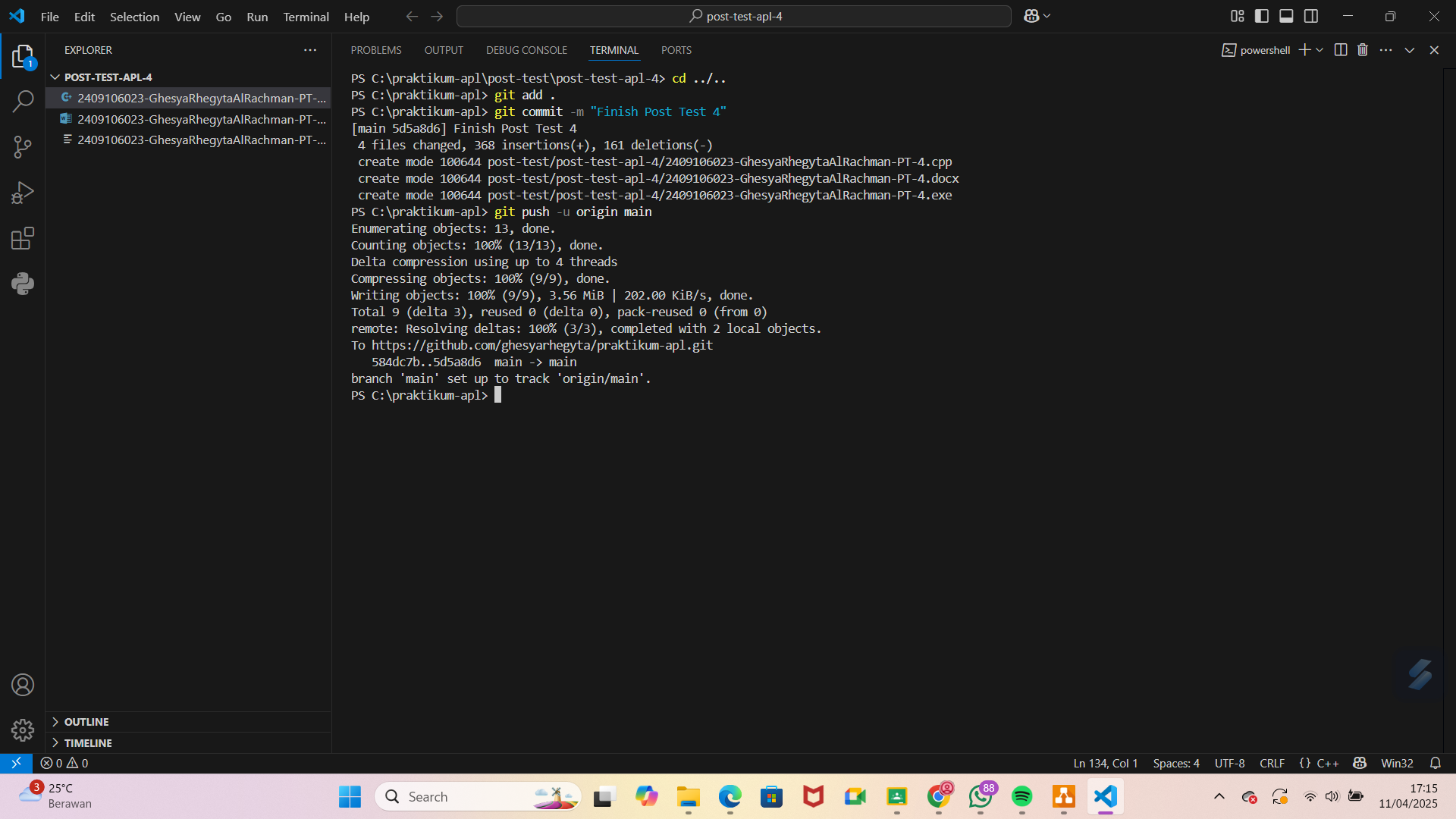
Menambahkan semua perubahan seperti file baru, file yang dimodifikasi, maupun file yang dihapus ke dalam staging area Git sebelum dikomit. Tanda titik (.) menunjukkan semua file dalam folder proyek akan ditambahkan ke staging area



Gambar 5.1 Git add .

2. Git Commit

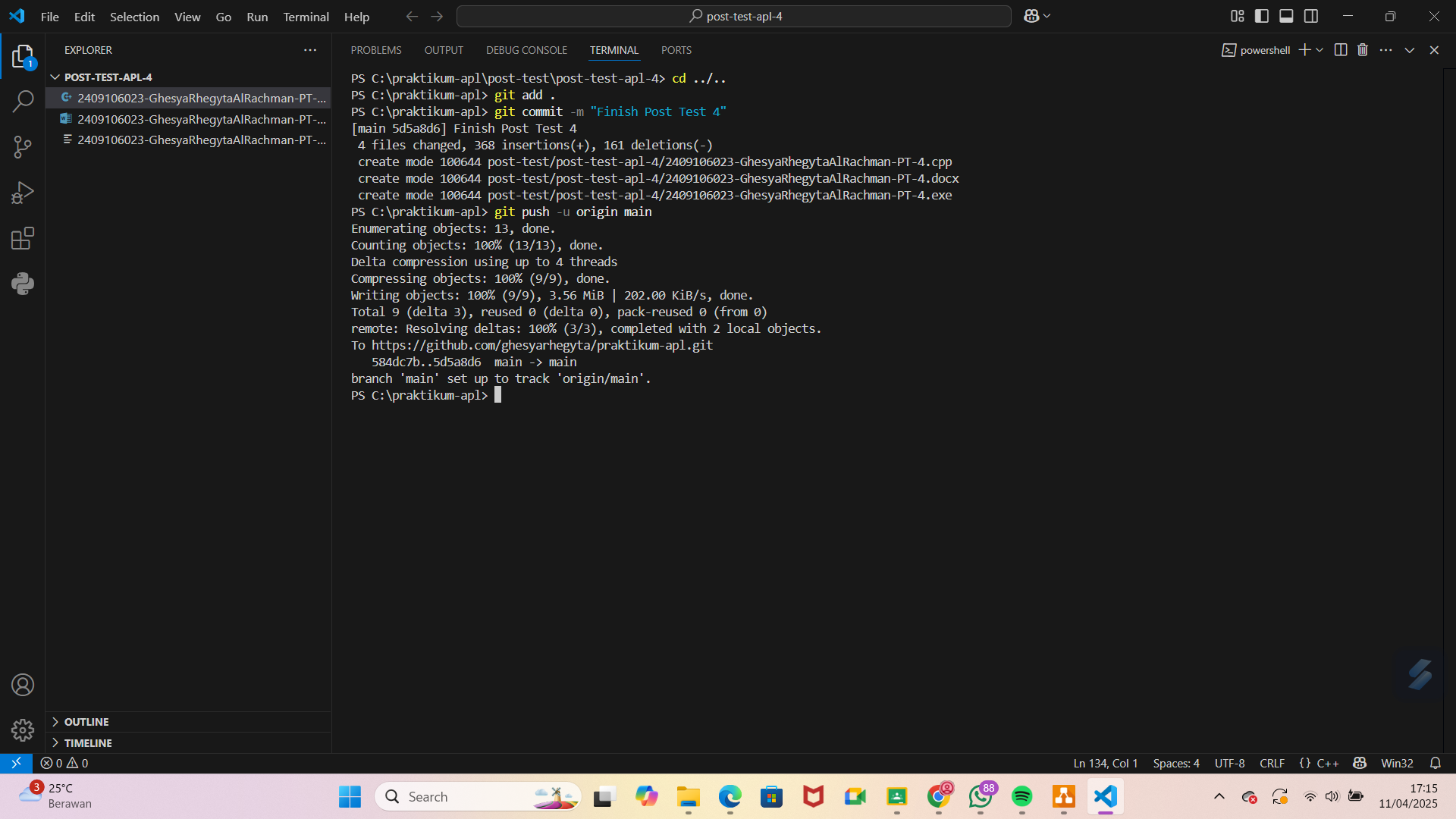
Menyimpan perubahan yang ada di staging area ke dalam commit history. Setiap commit mencatat perubahan apa yang dilakukan pada kode. (-m) digunakan untuk menyertakan pesan commit secara langsung



Gambar 5.2 Git Commit

3. Git Push

Mengirim commit yang ada di lokal ke server seperti GitHub. Digunakan agar dapat diakses oleh orang lain atau untuk kolaborasi



Gambar 5.3 Git Push