**LAPORAN PRAKTIKUM BAHASA PEMROGRAMAN 1**

**MODUL 10 : Project**

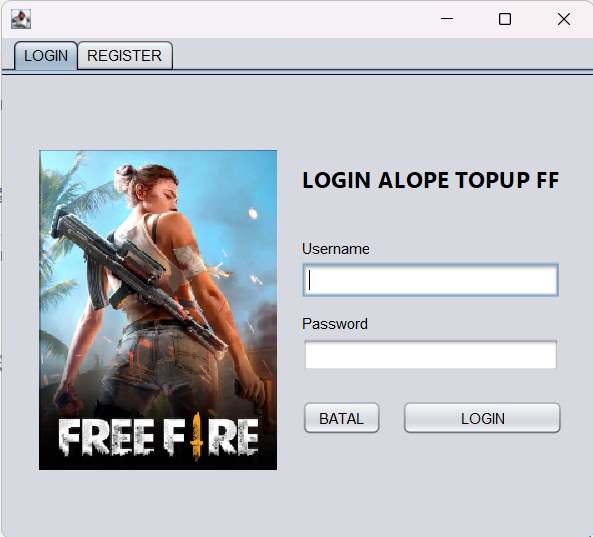
Nama : Ilham Hafidz

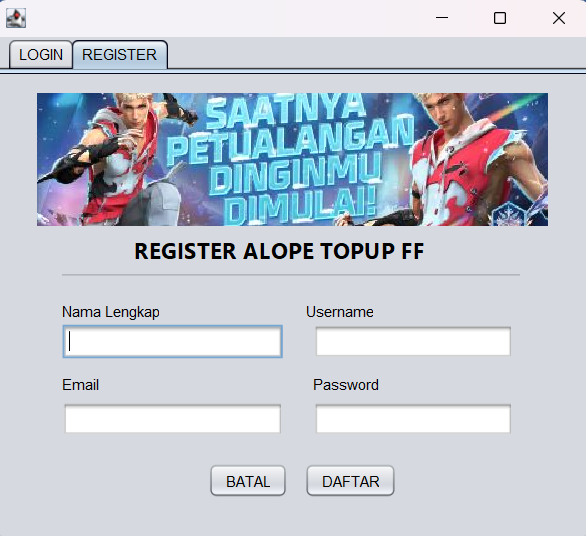
NIM : 20220810052

Kelas : TINFC-2022-01

**Result**

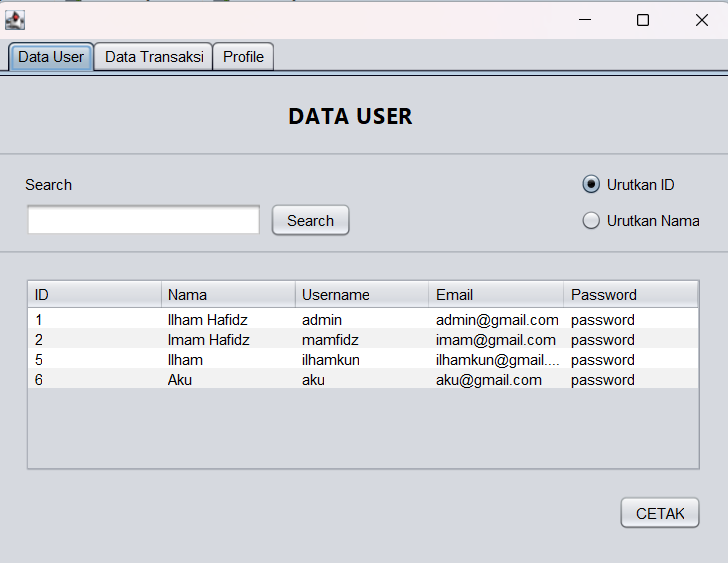
Ini adalah tampilan Authentication (Login & Register). Untuk level nya sendiri ada 2 level yaitu admin dan user, khusus untuk register, ketika register dia akan otomatis menjadi role user. Admin hanya akan ada 1.



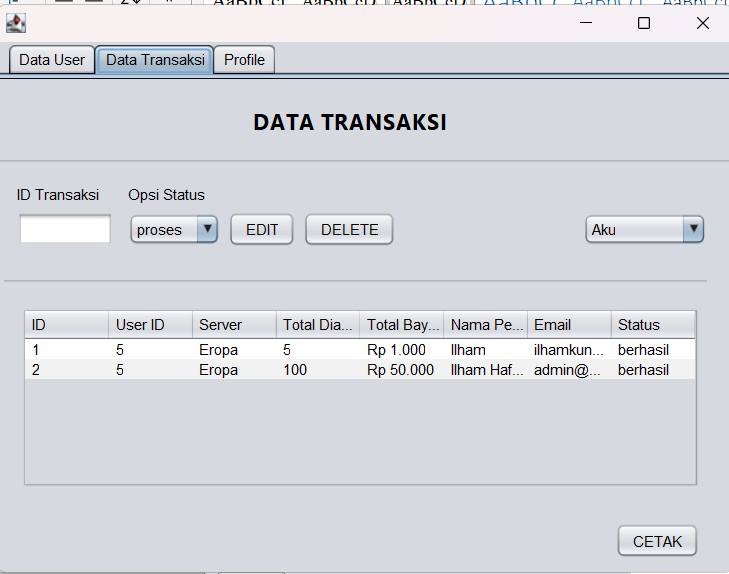


Berikut adalah tampilan windows jika berhasil login sebagai admin. Terdapat tab Data User, Data Transaksi dan Profile.

Tab data user akan menampilkan user yang terdaftar.

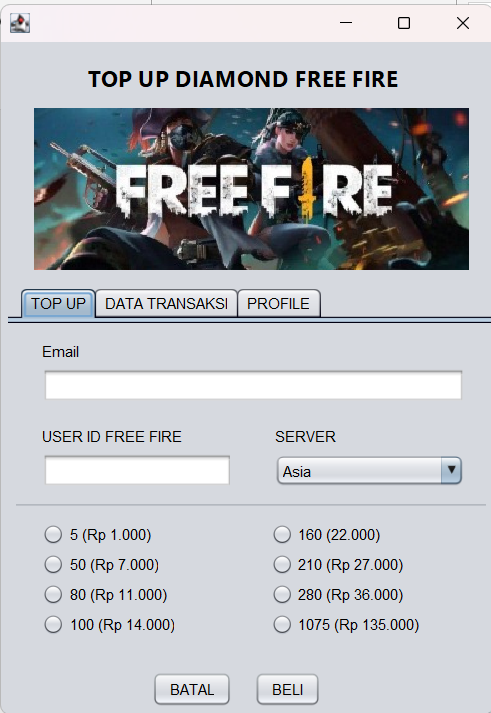


Tab data transaksi merupakan tab untuk manajemen transaksi dari user.

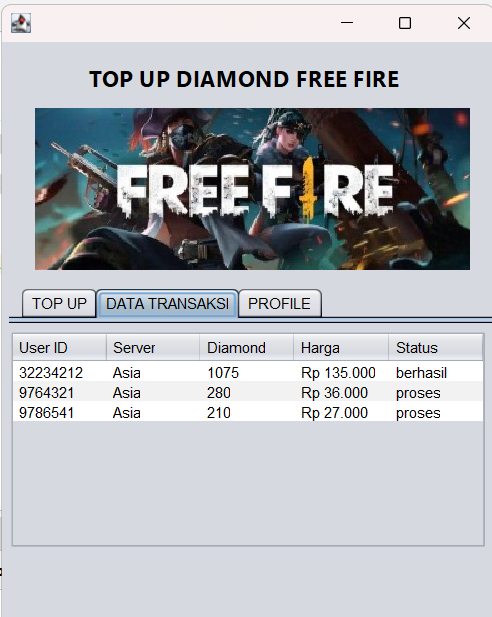


Berikut adalah windows jika berhasil login sebagai pengguna. Terdapat tab top up, data transaksi dan profile.

Tab top up merupakan tab untuk user melakukan request top up diamond-nya.



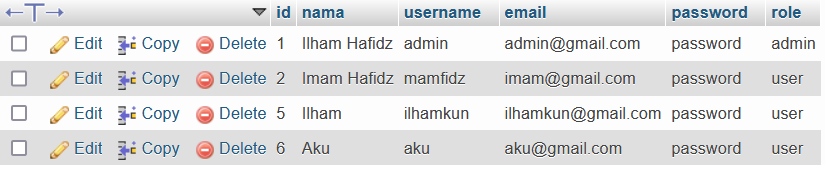
Tab Data Transaksi merupakan tab untuk melihat history dan status dari transaksi.



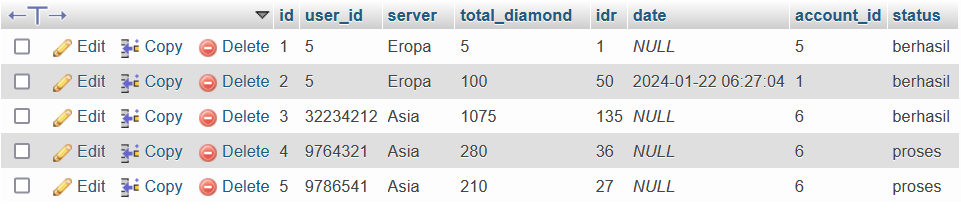
Pada tab Profile di admin maupun user yaitu hanya untuk melakukan logout.

**Berikut Adalah Skema Database nya:**

* Tabel accounts



* Tabel transactions



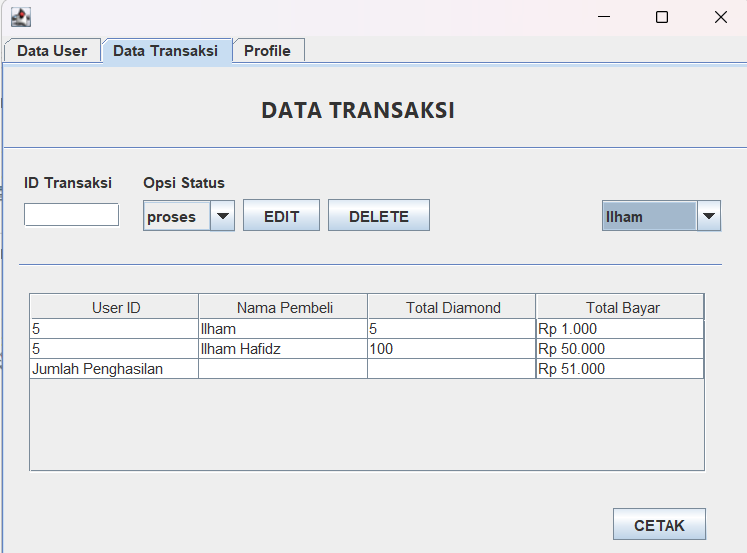
**Penerapan Algoritma**

|  |
| --- |
| Menghitung Jumlah Penghasilan dan Filter Data |
| private void comboBoxFilterTransaksiActionPerformed(java.awt.event.ActionEvent evt) {  int userId= 0;    String sqlTransaction= "SELECT \* FROM transactions INNER JOIN accounts ON transactions.account\_id = accounts.id WHERE user\_id=" + userId;    int total= 0;    try{  st= connect.con.createStatement();  rs= st.executeQuery(sqlTransaction);    while(rs.next()){  String col1= rs.getString(2);  String col2= rs.getString("nama");  String col3= rs.getString(4);  String col4= "Rp " + rs.getString(5) + ".000";    total += rs.getInt(5);    String k[]= {col1, col2, col3, col4};  data.addRow(k);  }  String k[]= {"Jumlah Penghasilan", "", "", "Rp " + Integer.toString(total) + ".000"};  data.addRow(k);  } catch(SQLException e) {  JOptionPane.showMessageDialog(null, e);  }  } |

Analisis :

* Untuk filterisasi data transaksinya, saya menggunakan SQL syntax yaitu “WHERE”, dimana ini akan berfungsi agar hanya yang memenuhi kondisi WHERE-nya saja yang akan ditampilkan pada tabel data transaksi.
* Untuk menghitung total penghasilan, cukup menggunakan operasi penjumlahan setiap perulangan while nya. Disini saya menggunakan variabel total yang di inisialisasi dengan nilai 0 dan di setiap perulangannya akan dilakukan “total = total + hargaTopup”.

Output



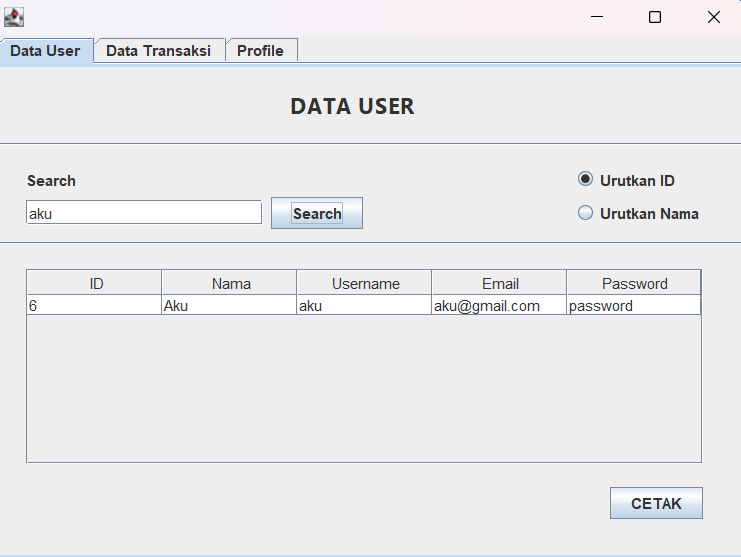
Disini misalkan saya memfilter data transaksi yang nama pembelinya Ilham. Terdapat 2 transaksi yang sudah dilakukan user Ilham ini. Dan jumlah penghasilan ALOPE TOP UP FF dari user Ilham adalah 51 ribu.

|  |
| --- |
| Search Data User |
| private void buttonSearchActionPerformed(java.awt.event.ActionEvent evt) {  String sql= "SELECT \* FROM accounts WHERE nama LIKE '%" + textFieldSearch.getText() + "%'";  ………..  **}** |

Analisis :

Untuk proses search, saya menggunakan SQL Syntax dengan WHERE dan LIKE. LIKE syntax merupakan syntax SQL untuk mengambil data “yang mengandung kata” dari value LIKE nya. Misalkan “WHERE nama LIKE ilham”, berarti saya ingin memfilter search data yang namnya mengandung nama ilham.

Output:



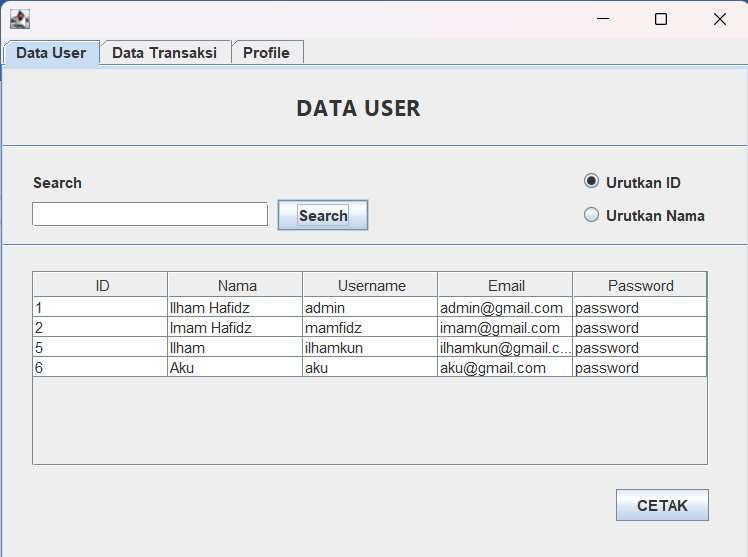
|  |
| --- |
| Pengurutan Data User |
| private void radioButtonOrderNamaActionPerformed(java.awt.event.ActionEvent evt) {  String sql= "SELECT \* FROM accounts ORDER BY nama ASC";………..  **}** |

Analisis

Dengan syntax ORDER BY pada SQL, disini saya bisa mengatur pengurutan dari get data nya. ASC merupakan syntax untuk melakukan data secara ascending atau berurut dari atas ke bawah, kebalikan dari DESC (descending) yaitu pengurutan dari bawah ke atas.

Output :

Pengurutan ID



Pengurutan Nama

