

LAPORAN TUGAS BESAR PEMROGRAMAN II

Diajukan untuk Memenuhi Kelulusan Matakuliah Pemrograman II pada
Program Studi DIV Teknik Informatika



Disusun oleh:

714230065 Dwi Puspa Firdaus

714230069 Aghni Hasna Mufida

**PROGRAM STUDI DIV TEKNIK INFORMATIKA
UNIVERSITAS LOGISTIK DAN BISNIS INTERNASIONAL
BANDUNG
2025**

PEMBAHASAN

- Source code dan penjelasan

SOURCE CODE

```
internal class Koneksi
{
    string conectionstring = "Server=localhost;Database=eventory;Uid=root;Pwd=";
    MySqlConnection kon;

    public void OpenConnection()
    {
        kon = new MySqlConnection(conectionstring);
        kon.Open();
    }
    public void CloseConnection()
    {
        kon.Close();
    }

    public void ExecuteQuery(string query)
    {
        MySqlCommand command = new MySqlCommand(query, kon);
        command.ExecuteNonQuery();
    }

    public object ShowData(string query)
    {
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conectionstring);
        DataSet data = new DataSet();

        adapter.Fill(data);
        Object datatable = data.Tables[0];
        return datatable;
    }

    public MySqlDataReader reader(string query)
    {
        MySqlCommand cmd = new MySqlCommand(query, kon);
        MySqlDataReader dr = cmd.ExecuteReader();
        return dr;
    }
}
```

Penjelasan :

Kode diatas merupakan kode untuk koneksi database, di inisialisasi sebagai class koneksi. Pertama ada method OpenConnection() itu untuk membuka koneksi ke mySql dan CloseConnection() itu untuk menutup koneksi ke database setelah operasi selesai. Lalu ada ExecuteQuery() itu untuk execute atau menjalankan perintah SQL seperti insert, delete dan update. ShowData() method yang ada untuk mengambil data dari database dalam bentuk datatable. Terakhir ada method MySqlDataReader untuk query select yang direturn dalam bentuk MySqlDataReader.

SOURCE CODE

```
Koneksi koneksi = new Koneksi();
public bool Insert(Barang barang)
{
    bool status = false;
    try
    {
        koneksi.OpenConnection();
        koneksi.ExecuteNonQuery("INSERT INTO t_barang (nama_barang, kategori, stok)
VALUES ('" + barang>Nama_barang + "', '" + barang.Kategori + "', '" +
barang.Stok + "')");
        status = true;
        MessageBox.Show("Data berhasil ditambahkan", "Informasi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Gagal", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
    return status;
}

public bool Update(Barang barang, string id)
{
    bool status = false;
    try
    {
        koneksi.OpenConnection();
        koneksi.ExecuteNonQuery("UPDATE t_barang SET nama_barang='" +
barang>Nama_barang + "', kategori='" + barang.Kategori + "', stok='" +
barang.Stok + "' WHERE id='" + barang.Id + "'");
        status = true;
        MessageBox.Show("Data berhasil diubah", "Informasi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Gagal", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
    return status;
}

public bool Delete(string id)
{
    bool status = false;
    try
    {
        koneksi.OpenConnection();
        koneksi.ExecuteNonQuery("DELETE FROM t_barang WHERE id='" + id + "'");
        status = true;
        MessageBox.Show("Data berhasil dihapus", "Informasi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {

```

<pre> MessageBox.Show(e.Message, "Gagal", MessageBoxButtons.OK, MessageBoxIcon.Error); } return status; } </pre>	
Penjelasan :	<p>Dari kode diatas merupakan kode dari class BarangController. Dalam BarangController terdapat method insert, update dan delete. Di setiap method terdapat query untuk insert, update dan delete data dari/ke table barang.</p>

<p>SOURCE CODE</p> <pre> string id, nama_barang, kategori, stok; public Barang() { } public Barang(string id, string nama_barang, string kategori, string stok) { this.Id = id; this>Nama_barang = nama_barang; this.Kategori = kategori; this.Stok = stok; } public string Id { get => id; set => id = value; } public string Nama_barang { get => nama_barang; set => nama_barang = value; } } public string Kategori { get => kategori; set => kategori = value; } public string Stok { get => stok; set => stok = value; } } </pre>	
Penjelasan :	<p>Kode diatas merupakan kode dari model class Barang. Class barang diatas merepresentasikan data barang dalam aplikasi yang dimana dengan adanya class barang ini data dapat dikelola dengan lebih terstruktur sebelum masuk database.</p>

<p>SOURCE CODE</p> <pre> Koneksi koneksi = new Koneksi(); Barang m_barang = new Barang(); BarangController barang = new BarangController(); public FormBarang() { InitializeComponent(); } private void FormBarang_Load(object sender, EventArgs e) { Tampil(); } public void Tampil() { dataGridViewDataBarang.DataSource = koneksi.ShowData("SELECT * FROM t_barang"); } </pre>	
---	--

<pre> dataGridViewDataBarang.Columns[0].HeaderText = "ID"; dataGridViewDataBarang.Columns[1].HeaderText = "Nama Barang"; dataGridViewDataBarang.Columns[2].HeaderText = "Kategori"; dataGridViewDataBarang.Columns[3].HeaderText = "Stok"; } </pre>	
Penjelasan :	Di dalam class FormBarang ini menginisialisasi objek Koneksi, Barang, dan BarangController untuk mengelola data barang dalam aplikasi. Kemudian ada method FormBarang_Load yang saat form dimuat dia memanggil method Tampil() yang dimana method ini memanggil data dari database dengan query select lalu ditampilkan di dataGridViewDataBarang.

<p>SOURCE CODE</p> <pre> private void buttonTambah_Click(object sender, EventArgs e) { if (string.IsNullOrEmpty(textBoxNamaBrg.Text) string.IsNullOrEmpty(textBoxStok.Text)) { MessageBox.Show("Data tidak boleh kosong", "Peringatan", MessageBoxButtons.OK, MessageBoxIcon.Warning); } else { m_barang.Nama_barang = textBoxNamaBrg.Text; m_barang.Kategori = comboBoxKategori.SelectedItem.ToString(); m_barang.Stok = textBoxStok.Text; if (barang.Insert(m_barang)) { Reset(); Tampil(); } } } </pre>	
Penjelasan :	Kode diatas merupakan kode buttonTambah saat di click untuk menambahkan data ke database. Kondisi pertama jika textBoxNamaBrg dan textBoxStok kosong maka akan menampilkan data tidak boleh kosong, dan jika kondidi tidak kosong, data dari textBoxNamaBrg, comboBoxKategori, dan textBoxStok akan disimpan ke dalam objek m_barang. Setelah itu, method Insert(m_barang) dari BarangController dipanggil untuk memasukkan data ke database. Jika proses penyimpanan berhasil, maka method Reset() akan dipanggil untuk mengosongkan input, dan Tampil() untuk memperbarui tampilan data di DataGridView.

<p>SOURCE CODE</p> <pre> private void buttonEdit_Click(object sender, EventArgs e) { if (dataGridViewDataBarang.SelectedRows.Count == 0) { MessageBox.Show("Pilih data yang ingin diedit!", "Peringatan", MessageBoxButtons.OK, MessageBoxIcon.Warning); } else if (string.IsNullOrEmpty(textBoxNamaBrg.Text) string.IsNullOrEmpty(textBoxStok.Text)) </pre>	
---	--

```

    {
        MessageBox.Show("Data tidak boleh kosong", "Peringatan",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }

    else
    {
        string id =
dataGridViewDataBarang.SelectedRows[0].Cells[0].Value.ToString();
        m_barang.Id = id;
        m_barang>Nama_barang = textBoxNamaBrg.Text;
        m_barang.Kategori = comboBoxKategori.SelectedItem.ToString();
        m_barang.Stok = textBoxStok.Text;

        if (barang.Update(m_barang, id))
        {
            Reset();
            Tampil();
        }
        else
        {
            MessageBox.Show("Gagal mengubah data", "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

Penjelasan :

Kode diatas merupakan kode untuk edit data barang user harus memilih data yang akan diedit dan data yang diedit tidak boleh kosong. Data yang akan diedit itu dipilih berdasarkan id nya dan jika data di update maka method update dipanggil dari BarangController untuk menyimpan perubahan data dan jika berhasil maka method reset() akan dijalankan dan method tampil() akan dipanggil untuk menampilkan data yang sudah diperbarui. Sedangkan jika gagal maka akan tampil error.

SOURCE CODE

```

private void buttonHapus_Click(object sender, EventArgs e)
{
    if (dataGridViewDataBarang.SelectedRows.Count == 0)
    {
        MessageBox.Show("Pilih data yang ingin dihapus!", "Peringatan",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    string id =
dataGridViewDataBarang.SelectedRows[0].Cells[0].Value.ToString();
    DialogResult pesan = MessageBox.Show("Apakah yakin akan menghapus data
ini?", "Perhatian", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (pesan == DialogResult.Yes)
    {
        if (barang.Delete(id))
        {
            Reset();
            Tampil();
        }
    }
}

```

Penjelasan :	Kode diatas merupakan kode button hapus, jika data yang dipilih kosong maka akan muncul peringatan untuk memilih data yang akan dihapus. Data yang dihapus ini berdasarkan id, saat klik data yang akan dihapus akan muncul messageBox untuk validasi data benar yakin dihapus atau tidak jika yes maka data dihapus dan method reset() dipanggil untuk mengosongkan tampilan dan method tampil() dipanggil kembali untuk menampilkan data terbarunya.
---------------------	--

SOURCE CODE

```
private void textBoxCari_TextChanged(object sender, EventArgs e)
{
    string query = "SELECT * FROM t_barang WHERE id LIKE '%" + textBoxCari.Text
+ "%' OR nama_barang LIKE '%" + textBoxCari.Text + "%'";
    dataGridViewDataBarang.DataSource = koneksi.ShowData(query);
}
```

Penjelasan :	Kode diatas merupakan kode untuk kolom cari, berisi query select dari table t_barang dicari berdasarkan id dan nama_barang.
---------------------	---

SOURCE CODE

```
namespace Gudangin.controller
{
    internal class BarangController
    {
        Koneksi koneksi = new Koneksi();

        // Insert Data Barang
        public bool Insert(Barang barang)
        {
            bool status = false;
            try
            {
                koneksi.OpenConnection();
                string query = "INSERT INTO t_barang (nama_barang, kategori, stok) VALUES (@nama, @kategori, @stok)";
                MySqlCommand cmd = new MySqlCommand(query, koneksi.kon);
                cmd.Parameters.AddWithValue("@nama", barang>Nama_barang);
                cmd.Parameters.AddWithValue("@kategori", barang.Kategori);
                cmd.Parameters.AddWithValue("@stok", barang.Stok);
                cmd.ExecuteNonQuery();
                status = true;

                MessageBox.Show("Data berhasil ditambahkan", "Informasi",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
                koneksi.CloseConnection();
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

```

        MessageBox.Show(e.Message, "Gagal menambahkan data",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return status;
}

```

Penjelasan :

Source code di atas adalah bagian dari controller yang menangani proses insert data barang ke dalam tabel t_barang. Objek Koneksi untuk membuka koneksi ke database, lalu menjalankan perintah sql insert into t_barang (nama_barang, kategori, stok) values (@nama, @kategori, @stok) menggunakan MySqlCommand, dengan parameter @nama, @kategori, dan @stok yang diisi dari objek Barang untuk mencegah sql injection. Jika proses berhasil, data akan ditambahkan, status dikembalikan sebagai true, dan pesan konfirmasi ditampilkan dengan MessageBox.Show(). Jika terjadi error, pesan kesalahan akan muncul. koneksi.CloseConnection() agar koneksi selalu tertutup setelah eksekusi, saat berhasil atau gagal.

SOURCE CODE

```

public bool Update(Barang barang, string id)
{
    bool status = false;

    try { koneksi.OpenConnection(); string query = "UPDATE t_barang SET nama_barang = @nama,
    kategori = @kategori, stok = @stok WHERE id = @id"; MySqlCommand cmd = new
    MySqlCommand(query, koneksi.kon); cmd.Parameters.AddWithValue("@nama",
    barang>Nama_barang); cmd.Parameters.AddWithValue("@kategori", barang.Kategori);
    cmd.Parameters.AddWithValue("@stok", barang.Stok); cmd.Parameters.AddWithValue("@id", id);
    cmd.ExecuteNonQuery(); status = true;

        MessageBox.Show("Data berhasil diubah", "Informasi",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Gagal mengubah data",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return status;
}

```


Penjelasan :	<p>Kode di atas adalah metode Update(Barang barang, string id) dalam kelas BarangController, yang digunakan buat mengubah data barang di tabel t_barang berdasarkan id. Pertama, koneksi ke database dibuka dengan koneksi.OpenConnection(), lalu perintah sql update dijalankan buat mengganti nama_barang, kategori, dan stok sesuai data dari objek Barang. Kalau berhasil, data bakal diperbarui, status berubah jadi true, dan muncul notifikasi MessageBox.Show(). kalau gagal, pesan error bakal ditampilkan.</p> <p>.</p>
--------------	---

SOURCE CODE

```

public bool Delete(string id)
{
    bool status = false;
    try
    {
        koneksi.OpenConnection();
        string query = "DELETE FROM t_barang WHERE id = @id";
        MySqlCommand cmd = new MySqlCommand(query, koneksi.kon);
        cmd.Parameters.AddWithValue("@id", id);
        cmd.ExecuteNonQuery();
        status = true;

        MessageBox.Show("Data berhasil dihapus", "Informasi", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Gagal menghapus data", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    return status;
}

```

Penjelasan :	<p>Kode di atas adalah metode Delete(string id) dalam kelas BarangController, yang digunakan buat menghapus data barang di tabel t_barang berdasarkan id. Pertama, koneksi ke database dibuka pakai koneksi.OpenConnection(), lalu perintah sql delete from t_barang where id = @id dijalankan buat menghapus data yang sesuai dengan id yang diberikan. Kalau proses berhasil, data bakal terhapus, status berubah jadi true, dan muncul notifikasi pakai MessageBox.Show(). Tapi kalau gagal, pesan error bakal muncul. Sama kayak metode sebelumnya, koneksi cuma ditutup kalau berhasil, jadi kalau ada error, koneksi bisa tetap terbuka. Solusinya, koneksi.CloseConnection() lebih baik ditaruh di blok finally biar koneksi selalu tertutup, baik saat sukses maupun gagal.</p>
--------------	---

SOURCE CODE

```
namespace Gudangin.model
{
    internal class User { string id_user, username, password, role;

        public User()
        {
        }

        public User(string id_user, string username, string password, string
role)
        {
            this.Id_user = id_user;
            this.Username = username;
            this.Password = password;
            this.Role = role;
        }

        public string Id_user { get => id_user; set => id_user = value; }
        public string Username { get => username; set => username = value; }
        public string Password { get => password; set => password = value; }
        public string Role { get => role; set => role = value; }
    }
}
```

Penjelasan :

Kode di atas adalah kelas User di model, yang digunakan buat merepresentasikan data pengguna dalam sistem. Kelas ini punya empat atribut: id_user, username, password, dan role, yang masing-masing sudah dibuat dalam bentuk property biar bisa diakses dengan get dan set. Ada dua constructor, satu kosong buat inisialisasi tanpa parameter, dan satu lagi buat langsung mengisi data saat objek User dibuat. Kelas ini dipakai buat menyimpan dan mengelola informasi pengguna dalam aplikasi.

SOURCE CODE

```
internal class UserController { Koneksi koneksi = new Koneksi();

// Login User
public bool Authenticate(User user)
{
    bool status = false;
    try
    {
```

```

        koneksi.OpenConnection();
        string query = "SELECT * FROM t_user WHERE username = @username
AND password = @password";
        MySqlCommand cmd = new MySqlCommand(query, koneksi.kon);
        cmd.Parameters.AddWithValue("@username", user.Username);
        cmd.Parameters.AddWithValue("@password", user.Password);

        MySqlDataReader reader = cmd.ExecuteReader();

        if (reader.Read())
        {
            user.Id_user = reader["id_user"].ToString();
            user.Role = reader["role"].ToString();
            status = true;
        }
        reader.Close();
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Gagal Login", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    return status;
}
}

```

Penjelasan :

Kode di atas adalah kelas UserController yang menangani login pengguna. Metode Authenticate(User user) mengecek username dan password di tabel t_user dengan query select. Kalau data cocok, id_user dan role disimpan ke objek User, lalu status login jadi true. Koneksi dibuka sebelum query dijalankan dan ditutup setelahnya. Kalau gagal, pesan error muncul.

SOURCE CODE

```

internal class UserController { Koneksi koneksi = new Koneksi();

// Login User
public bool Authenticate(User user)
{
    bool status = false;
    try
    {
        koneksi.OpenConnection();
        string query = "SELECT * FROM t_user WHERE username =
@username AND password = @password";
        MySqlCommand cmd = new MySqlCommand(query, koneksi.kon);
    }
}

```

```

        cmd.Parameters.AddWithValue("@username", user.Username);
        cmd.Parameters.AddWithValue("@password", user.Password);

        MySqlDataReader reader = cmd.ExecuteReader();

        if (reader.Read())
        {
            user.Id_user = reader["id_user"].ToString();
            user.Role = reader["role"].ToString();
            status = true;
        }
        reader.Close();
        koneksi.CloseConnection();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Gagal Login",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return status;
}
}

```

Penjelasan :

Kode di atas adalah kelas UserController yang menangani login pengguna. Metode Authenticate(User user) mengecek username dan password di tabel t_user dengan query select. Kalau data cocok, id_user dan role disimpan ke objek User, lalu status login jadi true. Koneksi dibuka sebelum query dijalankan dan ditutup setelahnya. Kalau gagal, pesan error muncul.

SOURCE CODE

```

public partial class FormLogin : Form { UserController userController = new
UserController(); User m_user = new User(); // Model User dibuat di dalam FormLogin
public FormLogin() { InitializeComponent(); btnLogin.Click += btnLogin_Click; }

private void btnLogin_Click(object sender, EventArgs e)
{
    m_user.Username = tbUsername.Text.Trim();
    m_user.Password = tbPassword.Text.Trim();

    if (string.IsNullOrEmpty(m_user.Username) ||
    string.IsNullOrEmpty(m_user.Password))
    {

```

```

        MessageBox.Show("Username dan password harus diisi!",
"Peringatan", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (userController.Authenticate(m_user))
    {
        MessageBox.Show("Login berhasil sebagai " + m_user.Role,
"Sukses", MessageBoxButtons.OK, MessageBoxIcon.Information);

        this.Hide();

        if (m_user.Role == "admin")
        {
            FormBarang formMain = new FormBarang();
            formMain.ShowDialog();
        }
        else
        {
            FormUser userDashboard = new
FormUser(Convert.ToInt32(m_user.Id_user));
            userDashboard.ShowDialog();
        }

        this.Close();
    }
    else
    {
        MessageBox.Show("Username atau password salah!", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Penjelasan :

Kode di atas adalah FormLogin yang menangani proses login pengguna. Saat tombol Login ditekan (btnLogin_Click), username dan password diambil dari input lalu diperiksa apakah kosong. Kalau kosong, muncul peringatan. Kalau terisi, data dikirim ke userController.Authenticate(m_user). Kalau login berhasil, muncul pesan sukses, lalu form login disembunyikan (this.Hide()). Jika pengguna adalah admin, form FormBarang dibuka, sedangkan selain admin masuk ke FormUser dengan parameter Id_user. Setelah itu, form login ditutup (this.Close()). Kalau login gagal, pesan error ditampilkan.

SOURCE CODE

```
public partial class FormUser : Form { private PermintaanController permintaanController =
new PermintaanController(); private Koneksi koneksi = new Koneksi(); private int id_user;

public FormUser(int id_user)
{
    InitializeComponent();
    this.id_user = id_user;
}

private void FormUser_Load(object sender, EventArgs e)
{
    LoadDataBarang(); // Tampilkan daftar barang
}

private void LoadDataBarang()
{
    DataTable dt = koneksi.ShowData("SELECT id, nama_barang, kategori,
stok FROM t_barang");

    if (dt.Rows.Count > 0)
    {
        dataGridViewDataBarang.DataSource = dt;
        dataGridViewDataBarang.Columns[0].HeaderText = "ID";
        dataGridViewDataBarang.Columns[1].HeaderText = "Nama Barang";
        dataGridViewDataBarang.Columns[2].HeaderText = "Kategori";
        dataGridViewDataBarang.Columns[3].HeaderText = "Stok";
    }
    else
    {
        MessageBox.Show("Tidak ada data barang!", "Informasi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        dataGridViewDataBarang.DataSource = null;
    }
}
```

Penjelasan :

Kode di atas adalah FormUser, yang digunakan buat menampilkan daftar barang kepada pengguna berdasarkan data di tabel t_barang. Saat form dimuat (FormUser_Load), metode LoadDataBarang() dijalankan buat mengambil data barang dari database menggunakan query select id, nama_barang, kategori, stok from t_barang. Kalau ada data, hasilnya ditampilkan di DataGridView dengan header kolom yang sesuai (ID, Nama Barang, Kategori, Stok). Kalau nggak ada data, muncul pesan "Tidak ada data barang!" dan DataGridView dikosongkan.

SOURCE CODE

```
public partial class FormPesanan : Form
{
    private PermintaanController permintaanController = new PermintaanController();
    private Koneksi koneksi = new Koneksi();
    private int id_user; // ID user yang sedang login
    private int selectedIdPermintaan = -1;
    private string selectedStatus = ""; // Menyimpan status pesanan

    public FormPesanan(int userId)
    {
        InitializeComponent();
        this.id_user = userId;
    }

    private void FormPesanan_Load(object sender, EventArgs e)
    {
        LoadPesananUser();
    }

    private void LoadPesananUser()
    {
        DataTable dt = permintaanController.GetPermintaanUser(id_user);

        if (dt.Rows.Count > 0)
        {
            dataGridViewDataPesanan.DataSource = dt;
            dataGridViewDataPesanan.Columns[0].HeaderText = "ID Permintaan";
            dataGridViewDataPesanan.Columns[1].HeaderText = "Nama Barang";
            dataGridViewDataPesanan.Columns[2].HeaderText = "Jumlah";
            dataGridViewDataPesanan.Columns[3].HeaderText = "Status";
        }
    }
}
```

```

        dataGridViewDataPesanan.Columns[4].HeaderText = "Tanggal Permintaan";
    }
    else
    {
        dataGridViewDataPesanan.DataSource = null;

        MessageBox.Show("Tidak ada pesanan.", "Informasi", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
    }
}

```

Penjelasan :

Kode di atas adalah FormPesanan, yang digunakan buat menampilkan pesanan pengguna berdasarkan id_user yang sedang login. Saat form dimuat (FormPesanan_Load), metode LoadPesananUser() dijalankan buat mengambil data pesanan dari PermintaanController dan menampilkannya di dataGridViewDataPesanan. Kalau ada data, kolomnya diberi header yang sesuai (ID Permintaan, Nama Barang, Jumlah, Status, Tanggal Permintaan). Kalau nggak ada data, tabel dikosongkan dan muncul pesan "Tidak ada pesanan".

SOURCE CODE

```

private void dataGridViewDataPesanan_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewDataPesanan.Rows[e.RowIndex];

        selectedIdPermintaan = Convert.ToInt32(row.Cells[0].Value); // Simpan ID permintaan
        textBoxNamaBrg.Text = row.Cells[1].Value.ToString();
        textBoxJumlah.Text = row.Cells[2].Value.ToString();
        selectedStatus = row.Cells[3].Value.ToString(); // Simpan status

        // Hanya bisa membatalkan pesanan jika status masih "Pending"
        buttonBatal.Enabled = selectedStatus == "pending";
    }
}

```


}	
Penjelasan :	Kode di atas menangani event klik pada DataGridView (dataGridViewDataPesanan_CellClick). Saat pengguna klik baris tertentu, data dari baris tersebut diambil dan ditampilkan di textBoxNamaBrg dan textBoxJumlah. selectedIdPermintaan menyimpan ID permintaan, sedangkan selectedStatus menyimpan status pesanan. Tombol Batal (buttonBatal) hanya bisa diklik jika status pesanan masih "pending", sehingga pengguna hanya bisa membatalkan pesanan yang belum diproses.

SOURCE CODE <pre> private void buttonBatal_Click(object sender, EventArgs e) { if (selectedIdPermintaan == -1) { MessageBox.Show("Pilih pesanan yang ingin dibatalkan!", "Peringatan", MessageBoxButtons.OK, MessageBoxIcon.Warning); return; } // Konfirmasi pembatalan DialogResult dialogResult = MessageBox.Show("Apakah Anda yakin ingin membatalkan pesanan ini?", "Konfirmasi", MessageBoxButtons.YesNo, MessageBoxIcon.Question); if (dialogResult == DialogResult.Yes) { if (permintaanController.HapusPermintaan(selectedIdPermintaan)) { MessageBox.Show("Pesanan berhasil dibatalkan.", "Sukses", MessageBoxButtons.OK, MessageBoxIcon.Information); LoadPesananUser(); // Refresh daftar pesanan ResetForm(); } else </pre>	
---	--

```

{
    MessageBox.Show("Gagal membatalkan pesanan.", "Error", MessageBoxButtons.OK,
    MessageBoxIcon.Error);

}

}

}

```

Penjelasan :

Kode di atas menangani pembatalan pesanan saat tombol Batal ditekan. Pertama, dicek apakah ada pesanan yang dipilih (`selectedIdPermintaan == -1`). Kalau belum, muncul peringatan. Kalau sudah, muncul dialog konfirmasi. Jika pengguna memilih Yes, metode `HapusPermintaan(selectedIdPermintaan)` di `permintaanController` dijalankan buat menghapus pesanan dari database. Kalau berhasil, muncul pesan sukses, daftar pesanan direfresh dengan `LoadPesananUser()`, dan form di-reset dengan `ResetForm()`. Kalau gagal, muncul pesan error.

SOURCE CODE

```

internal class Permintaan
{
    public int Id_permintaan { get; set; }
    public int Id_user { get; set; }
    public int Id_barang { get; set; }
    public int Jumlah { get; set; }
    public string Status { get; set; }
    public DateTime Tanggal_permintaan { get; set; }

    public Permintaan() { }

    public Permintaan(int idUser, int idBarang, int jumlah)
    {
        this.Id_user = idUser;
        this.Id_barang = idBarang;
        this.Jumlah = jumlah;
    }
}

```

<pre> this.Status = "Pending"; // Status default this.Tanggal_permintaan = DateTime.Now; } } </pre>	
Penjelasan :	<p>Kode di atas adalah kelas Permintaan, yang digunakan buat merepresentasikan data permintaan pesanan dalam sistem. Kelas ini punya enam property: Id_permintaan, Id_user, Id_barang, Jumlah, Status, dan Tanggal_permintaan. Ada dua constructor, satu kosong untuk inisialisasi tanpa data, dan satu lagi yang menerima idUser, idBarang, dan jumlah, dengan status default "Pending" dan Tanggal_permintaan diisi dengan waktu saat ini (DateTime.Now). Kelas ini digunakan buat menyimpan dan mengelola data pesanan pengguna.</p>

<p>SOURCE CODE</p> <pre> internal class PermintaanController { Koneksi koneksi = new Koneksi(); public bool TambahPermintaan(Permintaan permintaan) { try { string query = \$"INSERT INTO t_permintaan (id_user, id_barang, jumlah, status, tanggal_permintaan) " + \$"VALUES ({permintaan.Id_user}, {permintaan.Id_barang}, {permintaan.Jumlah}, " + \$"'Pending', CURDATE())"; koneksi.ExecuteQuery(query); // Jalankan query return true; // Jika berhasil } catch (Exception ex) { Console.WriteLine("Error: " + ex.Message); return false; // Jika gagal } } </pre>	
Penjelasan :	<p>Kode di atas adalah kelas PermintaanController, yang menangani penambahan permintaan pesanan ke tabel t_permintaan. Metode TambahPermintaan(Permintaan permintaan) membuat query insert untuk menyimpan id_user, id_barang, jumlah, status default "Pending", dan</p>

	tanggal otomatis (curdate()). Query dijalankan dengan koneksi.ExecuteQuery(query). Kalau berhasil, metode mengembalikan true, kalau gagal, error ditampilkan.
--	---

SOURCE CODE

```
public DataTable GetPermintaanUser(int idUser) { string query = $"SELECT p.id_permintaan,
b.nama_barang, p.jumlah, p.status, p.tanggal_permintaan " + $"FROM t_permintaan p JOIN
t_barang b ON p.id_barang = b.id " + $"WHERE p.id_user = {idUser} ORDER BY
p.tanggal_permintaan DESC";

return koneksi.ShowData(query);

}
```

Penjelasan :

Kode di atas adalah metode GetPermintaanUser(int idUser) yang digunakan buat mengambil daftar permintaan pesanan berdasarkan idUser. Query select mengambil id_permintaan, nama_barang, jumlah, status, dan tanggal_permintaan dari tabel t_permintaan (p) yang digabungkan (join) dengan tabel t_barang (b) berdasarkan id_barang. Hasilnya diurutkan (order by) berdasarkan tanggal_permintaan terbaru. Data dikembalikan dalam bentuk DataTable lewat koneksi.ShowData(query).

SOURCE CODE

```
public DataTable GetPermintaanPending() { string query = $"SELECT p.id_permintaan,
b.nama_barang, p.jumlah, p.status, p.tanggal_permintaan, u.username " + $"FROM
t_permintaan p " + $"JOIN t_barang b ON p.id_barang = b.id " + $"JOIN t_user u ON p.id_user =
u.id_user " + $"WHERE p.status = 'Pending' " + $"ORDER BY p.tanggal_permintaan DESC";

DataTable dt = koneksi.ShowData(query);

if (dt == null)
{
    MessageBox.Show("Query tidak mengembalikan data!", "Error",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}

return dt ?? new DataTable(); // Hindari null reference
```

```
}
```

Penjelasan :

Kode di atas berisi metode dalam kelas PermintaanController yang menangani pengelolaan status permintaan pesanan. GetPermintaanPending() untuk mengambil daftar permintaan dengan status "Pending" dari tabel t_permintaan. Data diambil dengan join ke tabel t_barang dan t_user buat mendapatkan nama_barang dan username. Hasilnya diurutkan berdasarkan tanggal_permintaan terbaru. Kalau query tidak mengembalikan data (dt == null), muncul pesan error. Metode ini mengembalikan DataTable atau objek kosong buat mencegah null reference.

SOURCE CODE

```
public bool UpdateStatusPermintaan(int idPermintaan, string newStatus) { try { string query =  
$"UPDATE t_permintaan SET status = '{newStatus}' WHERE id_permintaan = {idPermintaan}";  
koneksi.ExecuteQuery(query); return true; } catch (Exception ex) { Console.WriteLine("Error: " +  
ex.Message); return false; } }
```

Penjelasan :

UpdateStatusPermintaan(int idPermintaan, string newStatus) → Mengupdate status permintaan berdasarkan id_permintaan. Query update langsung dijalankan dengan koneksi.ExecuteQuery(query). Kalau berhasil, return true, kalau gagal, error dicetak di console dan return false.

SOURCE CODE

```
public partial class FormPermintaan : Form
```

```
{
```

```
    private PermintaanController permintaanController = new PermintaanController();
```

```
    private Koneksi koneksi = new Koneksi();
```

```
    private int selectedIdPermintaan = -1; // ID permintaan yang dipilih
```

```
    public FormPermintaan()
```

```
    {
```

```
        InitializeComponent();
```

```
    }
```

```
private void FormPermintaan_Load(object sender, EventArgs e)
{
    LoadPermintaanPending();
}

private void LoadPermintaanPending()
{
    if (permintaanController == null)
    {
        MessageBox.Show("permintaanController belum diinisialisasi!", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    DataTable dt = permintaanController.GetPermintaanPending();

    if (dt == null)
    {
        MessageBox.Show("Gagal mengambil data permintaan!", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);

        return;
    }

    if (dt.Rows.Count > 0)
    {
        dataGridViewDataPermintaan.DataSource = dt;

        dataGridViewDataPermintaan.Columns[0].HeaderText = "ID Permintaan";
        dataGridViewDataPermintaan.Columns[1].HeaderText = "Nama Barang";
        dataGridViewDataPermintaan.Columns[2].HeaderText = "Jumlah";
        dataGridViewDataPermintaan.Columns[3].HeaderText = "Status";
        dataGridViewDataPermintaan.Columns[4].HeaderText = "Tanggal Permintaan";
    }
}
```

```

        dataGridViewDataPermintaan.Columns[5].HeaderText = "User";
    }

    else
    {

        MessageBox.Show("Tidak ada permintaan pending.", "Informasi",
        MessageBoxButtons.OK, MessageBoxIcon.Information);

        dataGridViewDataPermintaan.DataSource = null;

    }
}

```

Penjelasan :

Kode di atas adalah FormPermintaan, yang digunakan buat menampilkan daftar permintaan dengan status "Pending". Saat form dimuat (FormPermintaan_Load), metode LoadPermintaanPending() dijalankan buat mengambil data dari PermintaanController. Kalau permintaanController belum diinisialisasi atau query gagal, muncul pesan error. Kalau ada data, hasilnya ditampilkan di DataGridView dengan header yang sesuai (ID Permintaan, Nama Barang, Jumlah, Status, Tanggal Permintaan, User). Kalau nggak ada data, muncul pesan "Tidak ada permintaan pending" dan tabel dikosongkan.

SOURCE CODE

```

private void dataGridViewDataPermintaan_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridViewDataPermintaan.Rows[e.RowIndex];

        selectedIdPermintaan = Convert.ToInt32(row.Cells[0].Value); // Simpan ID permintaan

        // Debugging untuk memastikan ID terbaca

        MessageBox.Show($"ID Permintaan Dipilih: {selectedIdPermintaan}", "Debug",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

```
// Tombol aktif jika ada data yang dipilih

buttonSetuju.Enabled = true;

buttonTolak.Enabled = true;

}

}
```

Penjelasan :

Kode di atas menangani event klik pada DataGridView (dataGridViewDataPermintaan_CellClick). Saat pengguna klik baris tertentu, ID permintaan dari kolom pertama disimpan ke selectedIdPermintaan. Untuk memastikan ID terbaca dengan benar, ditampilkan MessageBox sebagai debugging. Jika ada data yang dipilih, tombol Setuju (buttonSetuju) dan Tolak (buttonTolak) diaktifkan (Enabled = true), memungkinkan admin untuk memproses permintaan.

SOURCE CODE

```
private void buttonSetuju_Click(object sender, EventArgs e)
{
    if (selectedIdPermintaan == -1)
    {
        MessageBox.Show("Pilih permintaan terlebih dahulu!", "Peringatan",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);

        return;
    }

    if (permintaanController.UpdateStatusPermintaan(selectedIdPermintaan, "Disetujui"))
    {
        MessageBox.Show("Permintaan berhasil disetujui.", "Sukses", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

        LoadPermintaanPending(); // Refresh daftar permintaan
    }
    else
    {
        MessageBox.Show("Gagal menyetujui permintaan.", "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
```



```

    }
}

private void buttonTolak_Click(object sender, EventArgs e)
{
    if (selectedIdPermintaan == -1)
    {
        MessageBox.Show("Pilih permintaan terlebih dahulu!", "Peringatan",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);

        return;
    }

    if (permintaanController.UpdateStatusPermintaan(selectedIdPermintaan, "Ditolak"))
    {
        MessageBox.Show("Permintaan berhasil ditolak.", "Sukses", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

        LoadPermintaanPending(); // Refresh daftar permintaan
    }
    else
    {
        MessageBox.Show("Gagal menolak permintaan.", "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

```


Penjelasan :

Kode di atas menangani aksi menyetujui atau menolak permintaan saat tombol Setuju (buttonSetuju_Click) atau Tolak (buttonTolak_Click) ditekan.

- Cek apakah ada permintaan yang dipilih (selectedIdPermintaan == -1). Kalau belum, muncul peringatan.
- Jika ada, metode UpdateStatusPermintaan() di PermintaanController dipanggil untuk memperbarui status ke "Disetujui" atau "Ditolak".
- Kalau berhasil, muncul pesan sukses, lalu daftar permintaan di-refresh dengan LoadPermintaanPending().

	<ul style="list-style-type: none"> Kalau gagal, muncul pesan error. <p>Kode ini memastikan admin bisa memproses permintaan dengan mudah dan daftar selalu diperbarui.</p>
--	--

- Output Program**
Halaman Data Barang



Barang

Transaksi

Laporan

Logout

Gudangin

Data Barang

ID	Nama Barang	Kategori	Stok
2	Sapu el	Peralatan Kebersi...	100
3	Kompor	Peralatan Dapur	88
4	AC	Elektronik	100
5	Rice Cooker	Peralatan Dapur	100
8	Blender	Pelaratn Dapur	100
10	Vacuum Cleaner	Pelaratn Kebersi...	100

Search

Search

Input Data Barang

Nama Barang

Kategori

Stok

Tombol Aksi

Tambah Edit

Hapus Clear

Display

Halaman Login

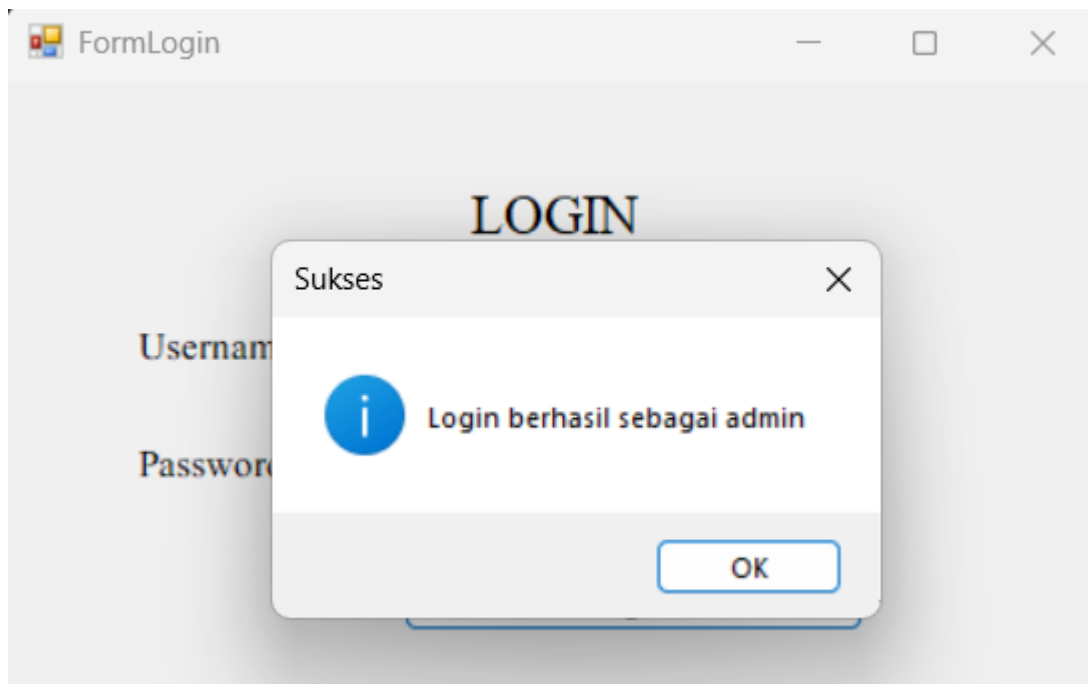
FormLogin

LOGIN

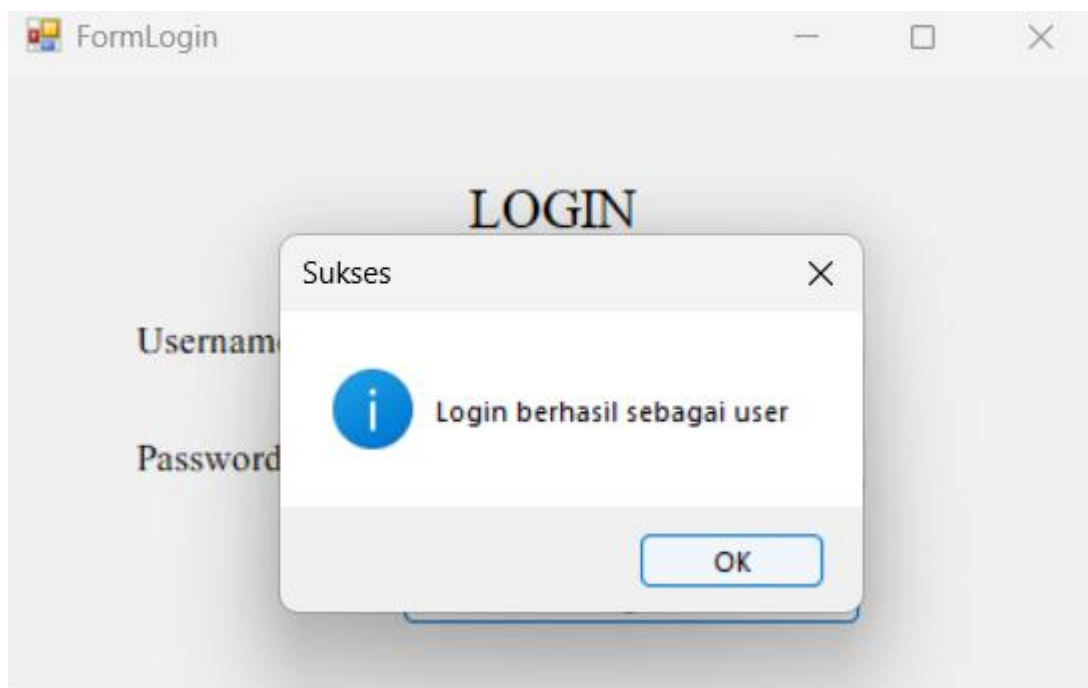
Username

Password


Login Admin



Login User



Halaman Pesanan User ada Dashboard Admin



Barang

Transaksi

Pesanan

Logout

Gudangin


Data Barang

	ID Permintaan	Nama Barang	Jumlah	Status	Tanggal Permintaan	Us
▶	6	Dispenser Air	10	pending	17/02/2025 18:45	tok
	4	Kompore	5	pending	17/02/2025 18:45	tok
*						

Search

Tombol Aksi

Halaman Data Barang pada Dashboard User



Barang

Status Pesanan

Logout

Gudangin

Data Barang

ID	Nama Barang	Kategori	Stok
2	Sapu el	Peralatan Kebersi...	100
3	Kompor	Peralatan Dapur	88
4	AC	Elektronik	100
5	Rice Cooker	Peralatan Dapur	100
8	Blender	Pelaratian Dapur	100
10	Vacuum Cleaner	Pelaratian Kebersi...	100

Search

Search

Input Data Barang


Nama Barang

Kategori

Jumlah

Pesan

Halaman Status Pesanan Pada User



Barang

Status Pesanan

Logout

Gudangin

ID Permintaan	Nama Barang	Jumlah	Status	Tanggal Permintaan
4	Kompor	5	pending	17/02/2025 18:45

Search

Search

Input Data Barang

Nama Barang

Kategori

Jumlah

Batalkan Pesanan

PENUTUP

Aplikasi Gudangin merupakan sistem manajemen inventory barang yang dirancang untuk mempermudah pengguna dalam mengelola stok barang, permintaan pesanan, serta proses persetujuan dan pembatalan permintaan. dengan konsep MVC (Model-View-Controller) untuk memisahkan logika bisnis, antarmuka pengguna, dan pengolahan data agar lebih terstruktur dan mudah dikem.

Melalui fitur CRUD (Create, Read, Update, Delete), pengguna dapat menambah, memperbarui, dan menghapus data barang, serta mengelola permintaan pesanan. Proses autentikasi pengguna memastikan hanya pengguna terdaftar yang bisa mengakses sistem, dan admin memiliki hak untuk menyetujui atau menolak permintaan pesanan. Data disajikan dalam bentuk DataGridView, memungkinkan pengguna melihat informasi barang dan pesanan dengan jelas.

Dari pengembangan ini, dapat disimpulkan bahwa Gudangin memberikan solusi efektif dalam mengelola inventory barang secara digital, mengurangi kesalahan pencatatan manual, serta meningkatkan efisiensi dalam pengelolaan stok dan permintaan barang. Ke depannya, aplikasi ini bisa dikembangkan lebih lanjut dengan fitur seperti notifikasi stok habis, laporan otomatis, serta integrasi dengan cloud storage untuk meningkatkan fungsionalitasnya.