**SEND CSV**

Saya akan menjelaskan alur proses button "Send CSV" secara lengkap dari awal sampai akhir:

1. Triggering Modal

* Ketika user mengklik button "Send CSV", Bootstrap JS akan mentrigger modal dengan ID "uploadModal" karena adanya atribut data-bs-toggle="modal" dan data-bs-target="#uploadModal"
* Modal yang ditampilkan berisi form untuk upload file CSV

1. Form Upload

* Modal menampilkan form dengan atribut:
  + method="post"
  + action="process.php"
  + enctype="multipart/form-data" (wajib untuk upload file)
* Input fields:
  + Hidden input dengan name="action" value="upload"
  + File input dengan id="csvFile" yang hanya menerima file .csv
* Form memiliki 2 button:
  + Close untuk menutup modal
  + Upload untuk submit form

1. Proses Upload (process.php)

* File diterima di process.php dan dilakukan beberapa validasi:
  + Pengecekan method POST
  + Validasi action "upload"
  + Pengecekan keberadaan file
  + Validasi tidak ada error saat upload
  + Validasi ekstensi file harus .csv

1. Pembacaan & Pemrosesan CSV

* Membuat koneksi database menggunakan class Database
* Membuka file CSV dengan fopen()
* Skip baris header dengan fgetcsv()
* Membaca data per baris:
  + Validasi jumlah kolom harus 12
  + Prepare statement SQL untuk insert/update
  + Data yang diinsert: nik, nama, gaji, hadir\_pusat, hadir\_proyek, konsumsi, lembur, tunjang\_lain, jkk, jkm, sehat, ptkp
  + Menggunakan ON DUPLICATE KEY UPDATE untuk update data jika NIK sudah ada
  + Menghitung jumlah sukses dan gagal

1. Database Operation

* Koneksi ke MySQL menggunakan PDO dengan parameter:
  + Host: localhost
  + User: rangga
  + Password: rangga
  + Database: avengers
  + Charset: utf8mb4

1. Response & Redirect

* Setelah proses selesai, set session alert dengan:
  + Success message jika berhasil
  + Error message jika gagal
* Redirect kembali ke home.php untuk menampilkan hasilnya

1. Security Features

* Menggunakan prepared statements untuk mencegah SQL injection
* Validasi file type
* Sanitasi input menggunakan fungsi sanitizeInput()
* Error handling dengan try-catch
* Session untuk menyimpan alert messages

Ini merupakan implementasi yang secure dan efisien untuk proses upload CSV dengan validasi yang baik dan error handling yang memadai.

**PROCESS.PHP**

Saya akan menjelaskan kode dan alur proses di process.php secara detail:

1. Inisialisasi Session dan Required Files

session\_start();

require\_once 'database.php'; // Class untuk koneksi database

require\_once 'functions.php'; // Helper functions

1. Validasi Request Method dan Action

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

if (isset($\_POST['action'])) {

if ($\_POST['action'] === 'upload' && isset($\_FILES['csvFile'])) {

* Cek request harus POST
* Cek keberadaan parameter 'action'
* Cek value action harus 'upload' dan file CSV harus ada

1. Validasi File Upload

if ($\_FILES['csvFile']['error'] !== UPLOAD\_ERR\_OK) {

$\_SESSION['alert'] = displayAlert('danger', 'File upload failed');

header('Location: home.php');

exit;

}

$file = $\_FILES['csvFile'];

if (pathinfo($file['name'], PATHINFO\_EXTENSION) !== 'csv') {

$\_SESSION['alert'] = displayAlert('danger', 'File must be a CSV');

header('Location: home.php');

exit;

}

* Cek tidak ada error saat upload
* Cek ekstensi file harus .csv
* Jika gagal, set alert error dan redirect ke home.php

1. Proses Database dan File

try {

$db = new Database();

$conn = $db->getConnection();

$handle = fopen($file['tmp\_name'], 'r');

fgetcsv($handle, 0, ';'); // Skip header row

* Inisialisasi koneksi database
* Buka file CSV
* Skip baris header

1. Proses Data CSV

$successCount = 0;

$failCount = 0;

while (($data = fgetcsv($handle, 0, ';')) !== false) {

if (count($data) !== 12) {

$failCount++;

continue;

}

* Inisialisasi counter success dan fail
* Loop setiap baris CSV
* Validasi jumlah kolom harus 12

1. Query Insert/Update

$sql = "INSERT INTO upah (nik, name, gaji, hadir\_pusat, hadir\_proyek, konsumsi,

lembur, tunjang\_lain, jkk, jkm, sehat, ptkp)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

ON DUPLICATE KEY UPDATE

name=VALUES(name), gaji=VALUES(gaji), hadir\_pusat=VALUES(hadir\_pusat),

hadir\_proyek=VALUES(hadir\_proyek), konsumsi=VALUES(konsumsi),

lembur=VALUES(lembur), tunjang\_lain=VALUES(tunjang\_lain),

jkk=VALUES(jkk), jkm=VALUES(jkm), sehat=VALUES(sehat),

ptkp=VALUES(ptkp)";

* Query untuk insert data baru atau update jika NIK sudah ada
* Menggunakan ON DUPLICATE KEY UPDATE untuk upsert operation

1. Execute Query

$stmt = $conn->prepare($sql);

try {

$stmt->execute([

trim($data[0]), // nik

trim($data[1]), // name

(int)$data[2], // gaji

(int)$data[3], // hadir\_pusat

(int)$data[4], // hadir\_proyek

(int)$data[5], // konsumsi

(int)$data[6], // lembur

(int)$data[7], // tunjang\_lain

(int)$data[8], // jkk

(int)$data[9], // jkm

(int)$data[10], // sehat

trim($data[11]) // ptkp

]);

$successCount++;

} catch (PDOException $e) {

$failCount++;

}

* Prepare statement untuk mencegah SQL injection
* Execute dengan array data yang sudah diformat:
  + String data di-trim()
  + Numeric data dikonversi ke integer
* Increment counter sesuai hasil

1. Finalisasi dan Response

fclose($handle);

$\_SESSION['alert'] = displayAlert(

'success',

"Upload completed: $successCount records successful, $failCount records failed"

);

} catch (Exception $e) {

$\_SESSION['alert'] = displayAlert('danger', 'Database error: ' . $e->getMessage());

}

* Tutup file CSV
* Set alert success dengan detail jumlah record berhasil/gagal
* Jika ada error database, set alert error

1. Redirect

header('Location: home.php');

exit;

* Redirect kembali ke home.php untuk menampilkan hasil

Fitur Keamanan yang Diterapkan:

1. Validasi file type
2. Prepared statements untuk SQL
3. Try-catch untuk error handling
4. Sanitasi data input
5. Type casting untuk data numerik
6. Session untuk message handling
7. Exit setelah redirect

Ini merupakan implementasi yang robust dengan:

* Error handling yang baik
* Keamanan yang memadai
* Feedback yang jelas ke user
* Support untuk bulk insert/update
* Optimasi performa dengan prepared statements

**FUNCTIONS.PHP**

Saya akan menjelaskan kode dan alur proses di functions.php secara detail:

1. Function sanitizeInput()

function sanitizeInput($input) {

return htmlspecialchars(trim($input));

}

Fungsi ini digunakan untuk membersihkan input dengan 2 langkah:

* trim(): Menghapus whitespace di awal dan akhir string
* htmlspecialchars(): Mengkonversi karakter khusus HTML menjadi entities untuk mencegah XSS attacks

Contoh penggunaan:

$name = sanitizeInput(" John Doe "); // Output: "John Doe"

$text = sanitizeInput("<script>alert('XSS')</script>");

// Output: &lt;script&gt;alert(&#039;XSS&#039;)&lt;/script&gt;

1. Function formatNumber()

function formatNumber($number) {

return number\_format($number, 0, ',', '.');

}

Fungsi ini memformat angka dengan:

* Parameter 1: Angka yang akan diformat
* Parameter 2: Jumlah desimal (0)
* Parameter 3: Pemisah desimal (',')
* Parameter 4: Pemisah ribuan ('.')

Contoh penggunaan:

$salary = formatNumber(1000000); // Output: "1.000.000"

$bonus = formatNumber(5500000); // Output: "5.500.000"

1. Function displayAlert()

function displayAlert($type, $message) {

return "

<div class='alert alert-{$type} alert-dismissible fade show' role='alert'>

{$message}

<button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'></button>

</div>";

}

Fungsi ini membuat alert Bootstrap dengan:

* Parameter $type: Tipe alert (success, danger, warning, info)
* Parameter $message: Pesan yang akan ditampilkan
* Menghasilkan HTML untuk alert yang bisa di-dismiss

Fitur alert:

* Class 'alert' sebagai base styling
* Class 'alert-{$type}' untuk warna sesuai tipe
* Class 'alert-dismissible' agar bisa ditutup
* Class 'fade show' untuk animasi
* Button close untuk menutup alert

Contoh penggunaan:

// Alert Success

echo displayAlert('success', 'Data berhasil disimpan');

// Output: <div class="alert alert-success alert-dismissible fade show"...>

// Alert Error

echo displayAlert('danger', 'Gagal mengunggah file');

// Output: <div class="alert alert-danger alert-dismissible fade show"...>

Implementasi dalam aplikasi:

1. Sanitizing Input

$name = sanitizeInput($\_POST['name']); // Membersihkan input nama

$nik = sanitizeInput($\_POST['nik']); // Membersihkan input NIK

1. Format Angka

$gaji = formatNumber($row['gaji']); // Format gaji: 5.000.000

$bonus = formatNumber($row['lembur']); // Format lembur: 1.500.000

1. Display Alert

// Success message

$\_SESSION['alert'] = displayAlert('success', 'Upload completed successfully');

// Error message

$\_SESSION['alert'] = displayAlert('danger', 'File upload failed');

Keuntungan penggunaan functions.php:

1. Reusability - fungsi bisa digunakan di berbagai file
2. Maintainability - perubahan cukup dilakukan di satu tempat
3. Security - membantu mencegah XSS attacks
4. Consistency - format angka dan alert yang konsisten
5. Clean code - mengurangi duplikasi kode

Best practices yang diterapkan:

1. Single Responsibility Principle - setiap fungsi memiliki satu tugas spesifik
2. DRY (Don't Repeat Yourself) - menghindari duplikasi kode
3. Security by design - sanitasi input secara default
4. Consistent formatting - format angka yang sesuai standar Indonesia
5. User feedback - sistem alert yang informatif

Functions.php merupakan helper file yang sangat penting dalam aplikasi ini, membantu standarisasi proses, meningkatkan keamanan, dan membuat kode lebih mudah di-maintain.

**DATABASE.PHP**

Saya akan menjelaskan kode dan alur proses di database.php secara detail:

1. Include Configuration

require\_once 'config.php';

Mengambil konstanta konfigurasi database:

* DB\_HOST: 'localhost'
* DB\_USER: 'rangga'
* DB\_PASS: 'rangga'
* DB\_NAME: 'avengers'
* DB\_CHARSET: 'utf8mb4'

1. Class Database

class Database {

private $conn;

* Membuat class Database untuk mengelola koneksi
* Property $conn bersifat private untuk enkapsulasi

1. Constructor Method

public function \_\_construct() {

try {

$this->conn = new PDO(

"mysql:host=" . DB\_HOST . ";dbname=" . DB\_NAME . ";charset=" . DB\_CHARSET,

DB\_USER,

DB\_PASS

);

$this->conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch(PDOException $e) {

error\_log("Database Connection Error: " . $e->getMessage());

throw new Exception("Database connection failed");

}

}

Alur konstruktor:

* Membuat koneksi PDO dengan parameter dari config.php
* Set atribut error mode ke ERRMODE\_EXCEPTION
* Jika gagal:
  + Log error ke error\_log sistem
  + Throw Exception untuk handling di level aplikasi

Detail parameter PDO:

* DSN string berisi:
  + Host: localhost
  + Database name: avengers
  + Character set: utf8mb4
* Username: rangga
* Password: rangga

1. Getter Method

public function getConnection() {

return $this->conn;

}

Method untuk mengakses koneksi:

* Public accessor untuk property private $conn
* Returns PDO connection object

Contoh penggunaan dalam aplikasi:

// Membuat instance Database

try {

$db = new Database();

$conn = $db->getConnection();

// Menggunakan koneksi

$stmt = $conn->prepare("SELECT \* FROM upah");

$stmt->execute();

} catch (Exception $e) {

// Handle error

}

Fitur keamanan yang diterapkan:

1. PDO untuk prepared statements
2. Error mode exception untuk error handling
3. Private property untuk enkapsulasi
4. Try-catch untuk exception handling
5. Error logging untuk debugging
6. UTF-8 character set untuk kompatibilitas

Best practices yang diimplementasi:

1. Singleton pattern - satu koneksi database
2. Separation of concerns - pemisahan konfigurasi
3. Encapsulation - property private
4. Error handling yang robust
5. Clean code principles
6. Security by design

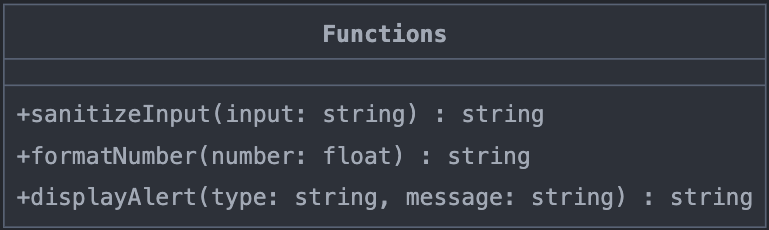
Keuntungan design ini:

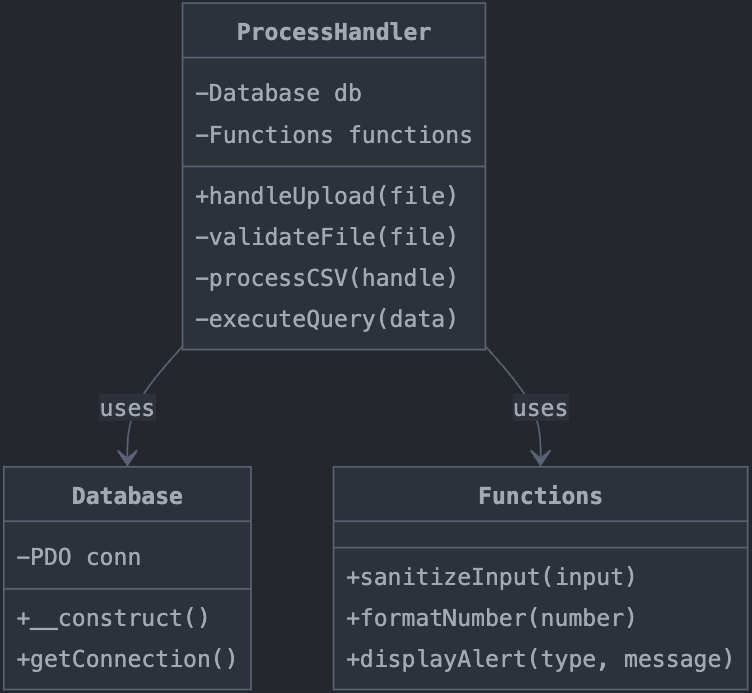
1. Maintainability - mudah diubah konfigurasi
2. Reusability - bisa digunakan di semua file
3. Security - implementasi prepared statements
4. Reliability - error handling yang baik
5. Debugging - logging error ke system

Implementasi koneksi database ini menggunakan:

1. PDO untuk:
   * Database abstraction
   * Prepared statements
   * Multiple database support
   * Exception handling
2. Modern PHP features:
   * Class dan OOP
   * Exception handling
   * Strict error reporting
   * Type hinting
3. Security features:
   * Parameter binding
   * Escape special characters
   * UTF-8 encoding
   * Error handling
4. Best practices:
   * Configuration separation
   * Encapsulation
   * Single responsibility
   * Clean code

Database.php merupakan komponen kritis yang menangani semua koneksi database dalam aplikasi dengan cara yang aman, efisien, dan mudah di-maintain.





A screenshot of a computer

Description automatically generated