**Creating a Display Interface IoT Web Dashboard**

*Andreika Luna Alghivari*

*Fakultas Vokasi, Universitas Brawijaya*

*Email: andreikaluna@student.ub.ac.id*

**Abstract**

This project develops a Laravel-based web dashboard that serves to visually display sensor data in the form of interactive charts using the Chart.js library. This dashboard is designed to make it easier for users to monitor sensor data in real-time and analyze trends through dynamic and responsive chart representations. In addition to visualization features, the dashboard also provides the ability to export data into Excel format by utilizing the Laravel Excel library, making it easier for users to perform further data processing or documentation. The system integrates Laravel's powerful backend with an interactive frontend, resulting in an efficient and user-friendly sensor data monitoring solution. This implementation is expected to support various IoT applications and web-based monitoring systems with complete data visualization and data management needs.

*Keywords: Web Dashboard, Internet of Things, Laravel, PlatformIO, Data Visualization, Data Export.*

**1. Introduction**

* 1. **Background**

Real-time monitoring of sensor data is crucial in various IoT applications. Presenting data through interactive charts simplifies analysis and decision-making. Combining Laravel as a robust backend with Chart.js for data visualization and Laravel Excel for data export creates an effective and user-friendly dashboard.

* 1. **Objective**

1. To develop a Laravel web dashboard to display sensor data in interactive charts using Chart.js.
2. To Provide a feature to export data to Excel format using Laravel Excel.
3. To Build a responsive and user-friendly interface for real-time sensor data monitoring.

**2. Methodology**

**2.1 Tools & Materials**

ESP32, Temperature and Humidity Sensor, USB Cable, PlatformIO and Visual Studio Code.

**2.2 Implementation Steps**

1. Open the chapter 12 project on the io platform in vscode
2. Open a terminal and run the composer require maatwebsite/excel command to install the export package
3. Create a new controller, GraphController, with the php artisan make:controller GraphController command.
4. Add a new routing in the routes/web.php file, namely

use App\Http\Controllers\GraphController;

Route::get('/', [GraphController::class, 'index'])->name('graph');

Route::get('/graph/export', [GraphController::class, 'exportToExcel'])->name('graph.export');

1. Then run the php command artisan make:export TransactionSensorExport --model=TransactionSensor on the terminal
2. Add the following code to the TransactionSensorExport file:

<?php

namespace App\Export;

use App\Models\TransactionSensor;

use Maatwebsite\Excel\Concept\FromCollection;

class TransactionSensorExport implements FromCollection

{

/\*\*

\* @back\Turn on\Support\Billing

\*/

public function collection()

{

return TransactionSensor::all();

}

}

1. Edit the web.php file in the routes folder to read as follows:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\GraphController;

Route::get('/', [GraphController::class, 'index'])->name('graph');

Route::get('/graph/export', [GraphController::class, 'exportToExcel'])->name('graph.export'); // Make sure this route exists

1. Create a graph.blade.php file in the resouces/views folder and add the code in the IoT module.
2. Run the program with the php artisan serve command.

**3. Results and Discussion**

**3.1 Experimental Results**

1. **Web Dashboard**

**A screenshot of a computer

Description automatically generated**

1. **Extract Excel Files**

A screenshot of a computer

Description automatically generated

The Laravel-based web dashboard has been successfully implemented and is fully operational. It effectively displays sensor data in interactive charts using Chart.js, providing clear and dynamic visualization. Additionally, the feature to export sensor data into Excel files using Laravel Excel has been successfully integrated and tested, allowing users to download the data easily for further analysis. This confirms that the dashboard meets the functional requirements for real-time data monitoring and data management.