CH5 - THORIQULHAQ

by Thoriqulhaq Jibril Al Qudsy

CHAPTER 5

IMPLEMENTATION AND TESTING

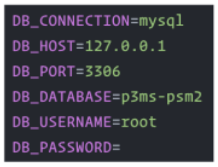
5.1 Introduction

This chapter describes the process of implementing and testing the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan which has been built in detail and presents examples of important program code snippets written in the development of this system. The processes involved are the installation of the software and hardware to develop the system, the coding activities, the types of tests performed and the results. The development process for the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan is based on determinations in the early stages of development such as system architecture design and database design. But there are also changes and additions made to the initial decision that has been set. This is to perform system repairs. Even though changes and additions have been made, the goals, objectives and user needs have not changed and further development has been carried out to meet the specified aspects.

5.2 Coding of System Main Functions

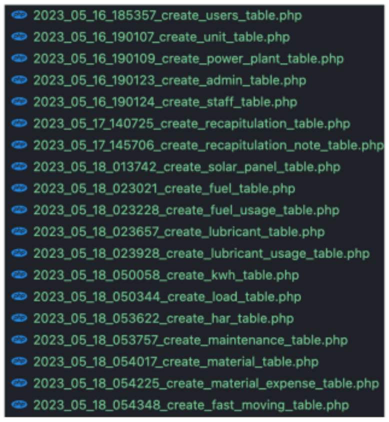
First, before going further into the details of the functions in the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan, this system uses a MySQL database where all data is placed in one database called p3ms-psm2 as shown in figure 5.1 which we can monitor through the database client named TablePlus. The definitions of field names, data types and primary keys for tables are determined through the database migration feature which is an innate feature of the Laravel framework itself as shown in figure 5.2. In addition, relationships between tables are also established.

After everything regarding the database has been settled, then the functions on the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan are made with the aim of later being able to help the process of running the system such as connecting the interface with the database that we have prepared previously also helping process the required input and output. These functions will later be found in the controller of each existing module and later the controller will have access to the Model so that it can access the database. The existence of this Model is actually not mandatory, but Laravel itself has an ORM feature called Eloquent and can only be accessed by creating a model for each table so we need to prepare it first. In this program there are 5 controllers that handle various functions as shown in figure 5.3 to 5.20 below.



```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=p3ms-psm2
DB_USERNAME=root
DB_PASSWORD=
```

Figure 5.1 Database Configuration in .ENV File



```
2023_05_16_185357_create_users_table.php
2023_05_16_190107_create_unit_table.php
2023_05_16_190109_create_power_plant_table.php
2023_05_16_190123_create_admin_table.php
2023_05_16_190124_create_staff_table.php
2023_05_17_140725_create_recapitulation_table.php
2023_05_17_145706_create_recapitulation_note_table.php
2023_05_18_013742_create_solar_panel_table.php
2023_05_18_023021_create_fuel_table.php
2023_05_18_023228_create_fuel_usage_table.php
2023_05_18_023657_create_lubricant_table.php
2023_05_18_023928_create_lubricant_usage_table.php
2023_05_18_050058_create_kwh_table.php
2023_05_18_050344_create_load_table.php
2023_05_18_053622_create_har_table.php
2023_05_18_053757_create_maintenance_table.php
2023_05_18_054017_create_material_table.php
2023_05_18_054225_create_material_expense_table.php
2023_05_18_054348_create_fast_moving_table.php
```

Figure 5.2 Database Migration File in ‘migration’ Folder

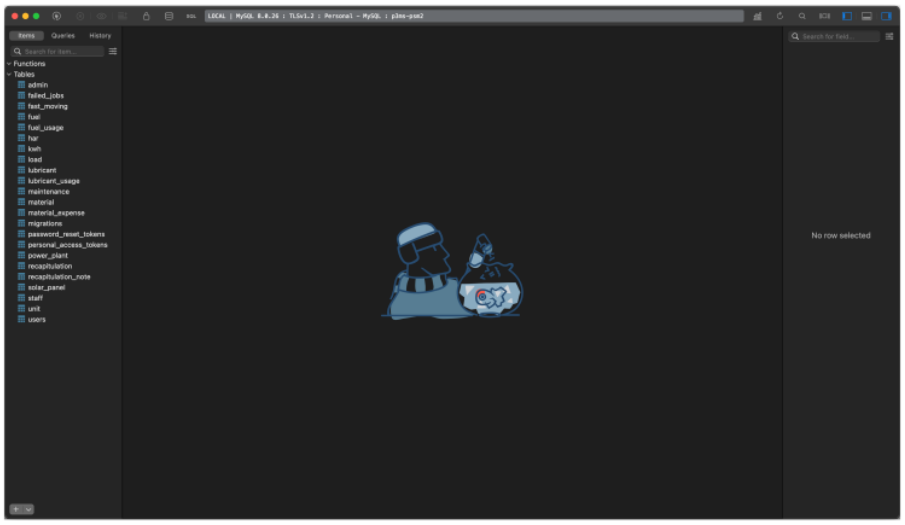
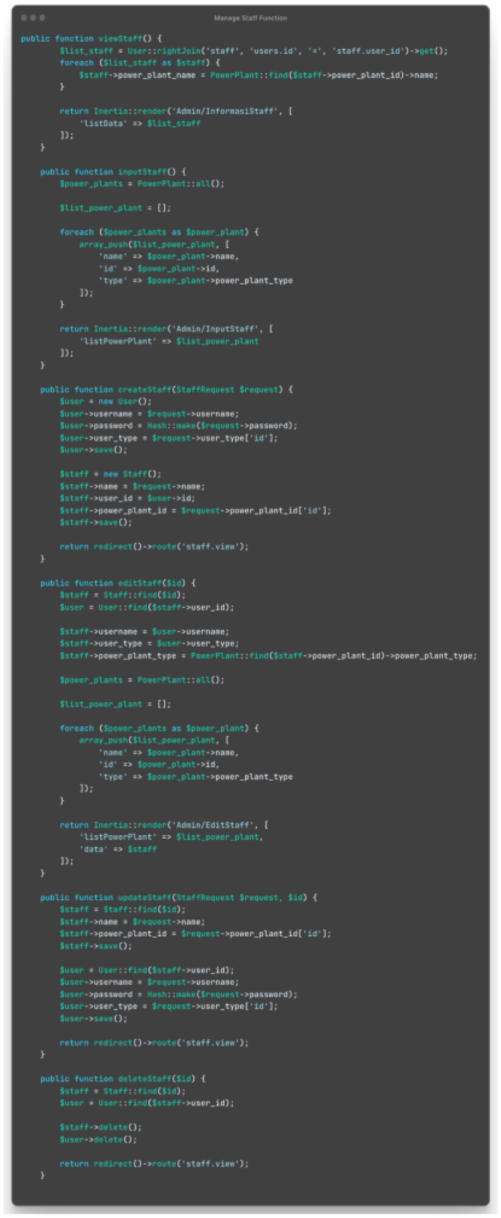


Figure 5.3 TablePlus as Database Client

5.2.1 Account Controller

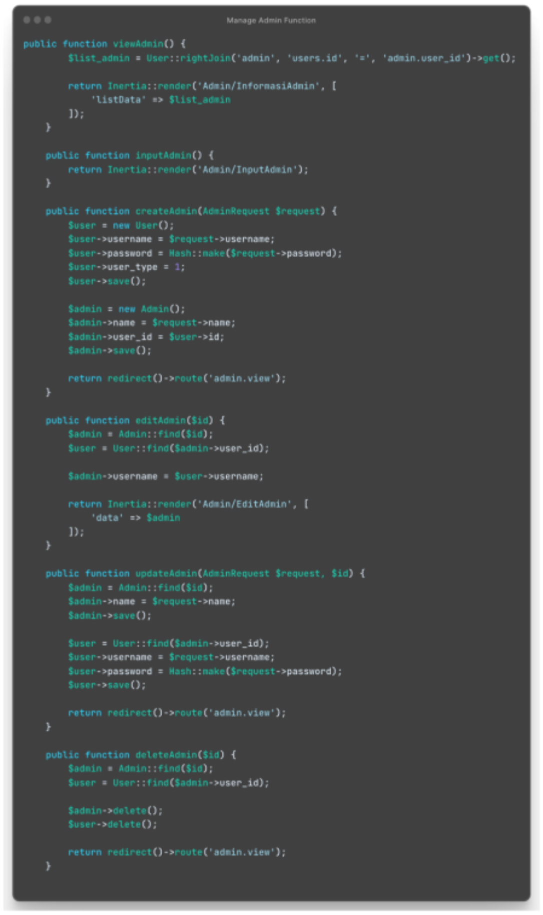
5.2.1.1 Manage Staff Function



```php
public function viewStaff() {
        $list_staff = User::rightJoin('staff', 'users.id', '=', 'staff.user_id')->get();
        foreach ($list_staff as $staff) {
            $staff->power_plant_name = PowerPlant::find($staff->power_plant_id)->name;
        }

        return Inertia::render('Admin/InformasiStaff', [
            'listData' => $list_staff
        ]);
    }

    public function inputStaff() {
        $power_plants = PowerPlant::all();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Admin/InputStaff', [
            'listPowerPlant' => $list_power_plant
        ]);
    }

    public function createStaff(StaffRequest $request) {
        $user = new User();
        $user->username = $request->username;
        $user->password = Hash::make($request->password);
        $user->user_type = $request->user_type['id'];
        $user->save();

        $staff = new Staff();
        $staff->name = $request->name;
        $staff->user_id = $user->id;
        $staff->power_plant_id = $request->power_plant_id['id'];
        $staff->save();

        return redirect()->route('staff.view');
    }

    public function editStaff($id) {
        $staff = Staff::find($id);
        $user = User::find($staff->user_id);

        $staff->username = $user->username;
        $staff->user_type = $user->user_type;
        $staff->power_plant_type = PowerPlant::find($staff->power_plant_id)->power_plant_type;

        $power_plants = PowerPlant::all();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Admin/EditStaff', [
            'listPowerPlant' => $list_power_plant,
            'data' => $staff
        ]);
    }

    public function updateStaff(StaffRequest $request, $id) {
        $staff = Staff::find($id);
        $staff->name = $request->name;
        $staff->power_plant_id = $request->power_plant_id['id'];
        $staff->save();

        $user = User::find($staff->user_id);
        $user->username = $request->username;
        $user->password = Hash::make($request->password);
        $user->user_type = $request->user_type['id'];
        $user->save();

        return redirect()->route('staff.view');
    }

    public function deleteStaff($id) {
        $staff = Staff::find($id);
        $user = User::find($staff->user_id);

        $staff->delete();
        $user->delete();

        return redirect()->route('staff.view');
    }
```

Figure 5.4 Manage Staff Function in Account Controller

5.2.1.2 Manage Admin Function

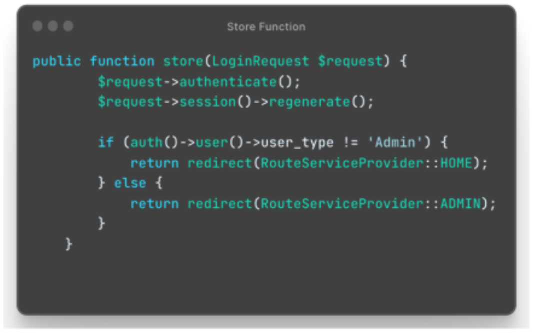


```php
public function viewAdmin() {
        $list_admin = User::rightJoin('admin', 'users.id', '=', 'admin.user_id')->get();

        return Inertia::render('Admin/InformasiAdmin', [
            'listData' => $list_admin
        ]);
    }

    public function inputAdmin() {
        return Inertia::render('Admin/InputAdmin');
    }

    public function createAdmin(AdminRequest $request) {
        $user = new User();
        $user->username = $request->username;
        $user->password = Hash::make($request->password);
        $user->user_type = 1;
        $user->save();

        $admin = new Admin();
        $admin->name = $request->name;
        $admin->user_id = $user->id;
        $admin->save();

        return redirect()->route('admin.view');
    }

    public function editAdmin($id) {
        $admin = Admin::find($id);
        $user = User::find($admin->user_id);

        $admin->username = $user->username;

        return Inertia::render('Admin/EditAdmin', [
            'data' => $admin
        ]);
    }

    public function updateAdmin(AdminRequest $request, $id) {
        $admin = Admin::find($id);
        $admin->name = $request->name;
        $admin->save();

        $user = User::find($admin->user_id);
        $user->username = $request->username;
        $user->password = Hash::make($request->password);
        $user->save();

        return redirect()->route('admin.view');
    }

    public function deleteAdmin($id) {
        $admin = Admin::find($id);
        $user = User::find($admin->user_id);

        $admin->delete();
        $user->delete();

        return redirect()->route('admin.view');
    }
```

Figure 5.5 Manage Admin Function in Account Controller

5.2.2 Authentication Controller

5.2.2.1 Store Function

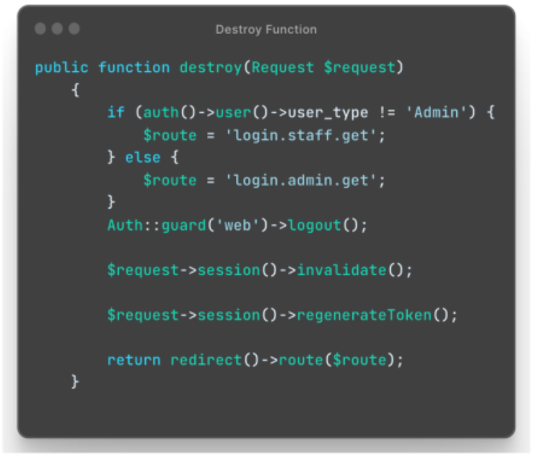


Store Function

```
public function store(LoginRequest $request) {
        $request->authenticate();
        $request->session()->regenerate();

        if (auth()->user()->user_type != 'Admin') {
            return redirect(RouteServiceProvider::HOME);
        } else {
            return redirect(RouteServiceProvider::ADMIN);
        }
    }
```

Figure 5.6 Store Function in Authentication Controller

5.2.2.2 Destroy Function



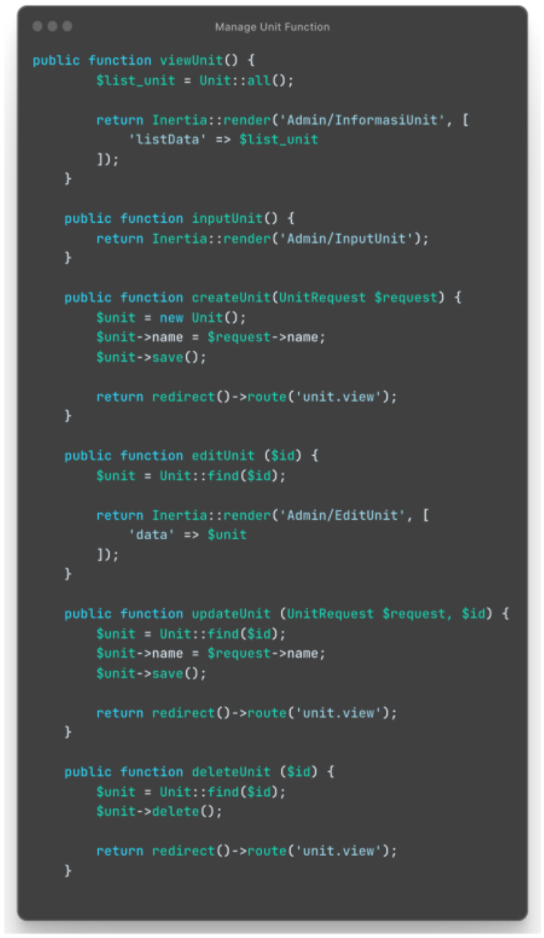
Destroy Function

```
public function destroy(Request $request)
    {
        if (auth()->user()->user_type != 'Admin') {
            $route = 'login.staff.get';
        } else {
            $route = 'login.admin.get';
        }
        Auth::guard('web')->logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect()->route($route);
    }
```

Figure 5.7 Destroy Function in Authentication Controller

5.2.3 Master Data Controller

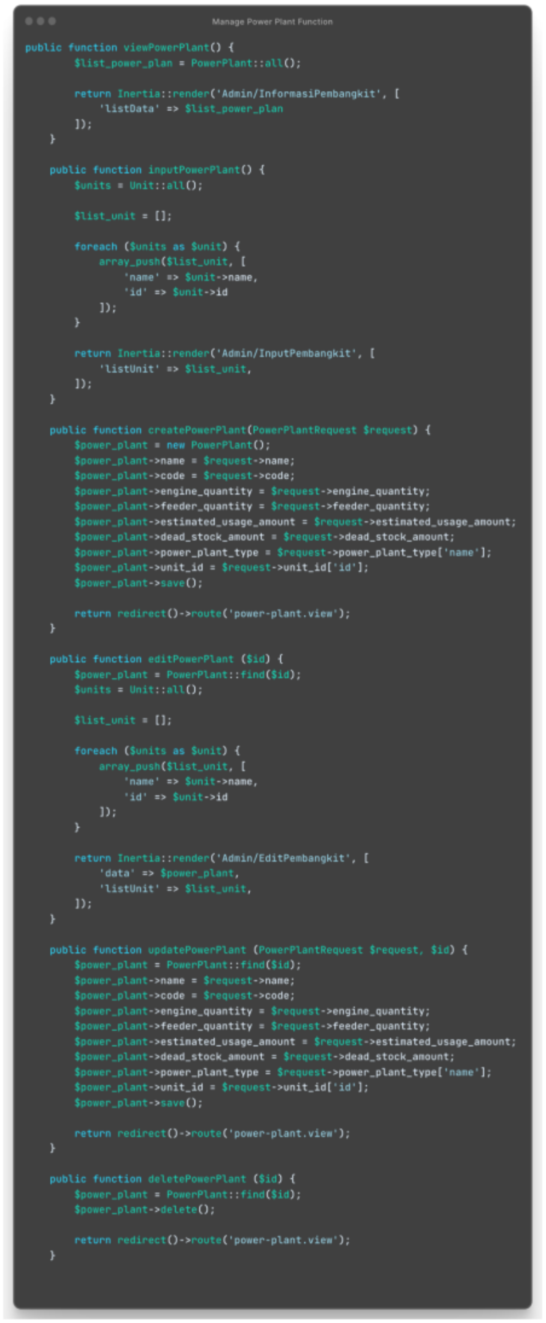
5.2.3.1 Manage Unit Function



```
public function viewUnit() {
        $list_unit = Unit::all();

        return Inertia::render('Admin/InformasiUnit', [
            'listData' => $list_unit
        ]);
    }

    public function inputUnit() {
        return Inertia::render('Admin/InputUnit');
    }

    public function createUnit(UnitRequest $request) {
        $unit = new Unit();
        $unit->name = $request->name;
        $unit->save();

        return redirect()->route('unit.view');
    }

    public function editUnit ($id) {
        $unit = Unit::find($id);

        return Inertia::render('Admin/EditUnit', [
            'data' => $unit
        ]);
    }

    public function updateUnit (UnitRequest $request, $id) {
        $unit = Unit::find($id);
        $unit->name = $request->name;
        $unit->save();

        return redirect()->route('unit.view');
    }

    public function deleteUnit ($id) {
        $unit = Unit::find($id);
        $unit->delete();

        return redirect()->route('unit.view');
    }
```

Figure 5.8 Manage Unit Function in Master Data Controller

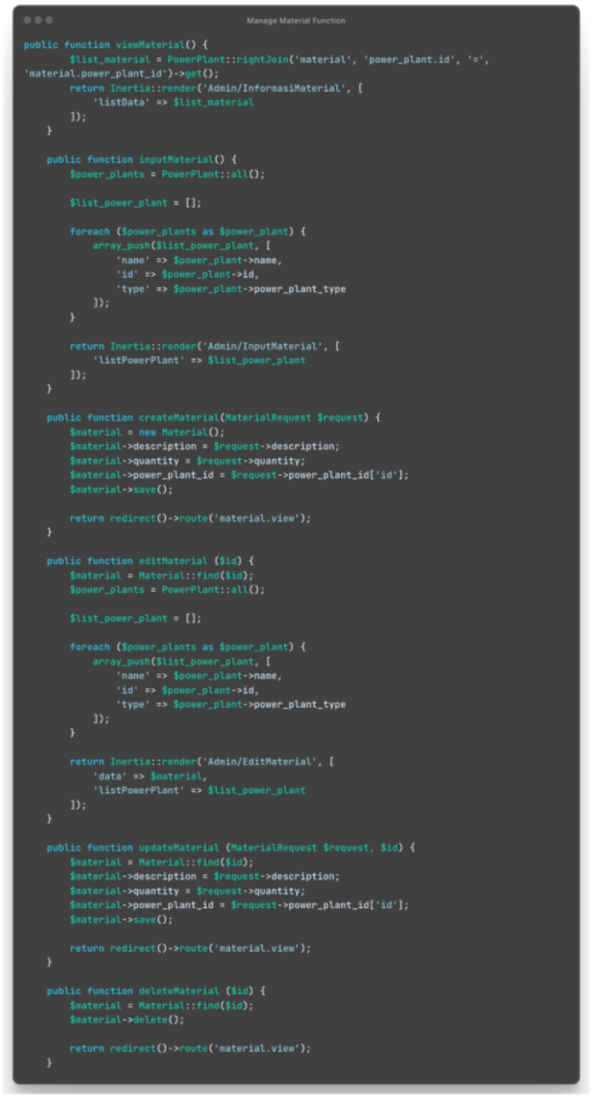
5.2.3.2 Manage Power Plant Function



```php
public function viewPowerPlant() {
        $list_power_plan = PowerPlant::all();

        return Inertia::render('Admin/InformasiPembangkit', [
            'listData' => $list_power_plan
        ]);
    }

    public function inputPowerPlant() {
        $units = Unit::all();

        $list_unit = [];

        foreach ($units as $unit) {
            array_push($list_unit, [
                'name' => $unit->name,
                'id' => $unit->id
            ]);
        }

        return Inertia::render('Admin/InputPembangkit', [
            'listUnit' => $list_unit,
        ]);
    }

    public function createPowerPlant(PowerPlantRequest $request) {
        $power_plant = new PowerPlant();
        $power_plant->name = $request->name;
        $power_plant->code = $request->code;
        $power_plant->engine_quantity = $request->engine_quantity;
        $power_plant->feeder_quantity = $request->feeder_quantity;
        $power_plant->estimated_usage_amount = $request->estimated_usage_amount;
        $power_plant->dead_stock_amount = $request->dead_stock_amount;
        $power_plant->power_plant_type = $request->power_plant_type['name'];
        $power_plant->unit_id = $request->unit_id['id'];
        $power_plant->save();

        return redirect()->route('power-plant.view');
    }

    public function editPowerPlant ($id) {
        $power_plant = PowerPlant::find($id);
        $units = Unit::all();

        $list_unit = [];

        foreach ($units as $unit) {
            array_push($list_unit, [
                'name' => $unit->name,
                'id' => $unit->id
            ]);
        }

        return Inertia::render('Admin/EditPembangkit', [
            'data' => $power_plant,
            'listUnit' => $list_unit,
        ]);
    }

    public function updatePowerPlant (PowerPlantRequest $request, $id) {
        $power_plant = PowerPlant::find($id);
        $power_plant->name = $request->name;
        $power_plant->code = $request->code;
        $power_plant->engine_quantity = $request->engine_quantity;
        $power_plant->feeder_quantity = $request->feeder_quantity;
        $power_plant->estimated_usage_amount = $request->estimated_usage_amount;
        $power_plant->dead_stock_amount = $request->dead_stock_amount;
        $power_plant->power_plant_type = $request->power_plant_type['name'];
        $power_plant->unit_id = $request->unit_id['id'];
        $power_plant->save();

        return redirect()->route('power-plant.view');
    }

    public function deletePowerPlant ($id) {
        $power_plant = PowerPlant::find($id);
        $power_plant->delete();

        return redirect()->route('power-plant.view');
    }
```

Figure 5.9 Manage Power Plant Function in Master Data Controller

5.2.3.3 Manage Material Function

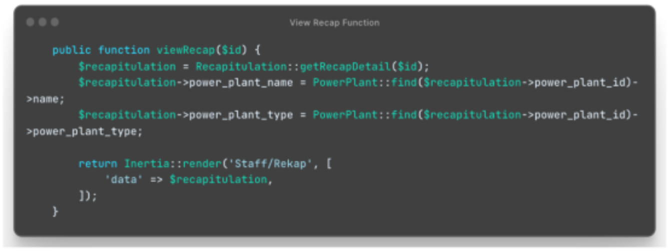


```php
public function viewMaterial() {
        $list_material = PowerPlant::rightJoin('material', 'power_plant.id', '=',
'material.power_plant_id')->get();
        return Inertia::render('Admin/InformasiMaterial', [
            'listData' => $list_material
        ]);
    }

    public function inputMaterial() {
        $power_plants = PowerPlant::all();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Admin/InputMaterial', [
            'listPowerPlant' => $list_power_plant
        ]);
    }

    public function createMaterial(MaterialRequest $request) {
        $material = new Material();
        $material->description = $request->description;
        $material->quantity = $request->quantity;
        $material->power_plant_id = $request->power_plant_id['id'];
        $material->save();

        return redirect()->route('material.view');
    }

    public function editMaterial ($id) {
        $material = Material::find($id);
        $power_plants = PowerPlant::all();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Admin/EditMaterial', [
            'data' => $material,
            'listPowerPlant' => $list_power_plant
        ]);
    }

    public function updateMaterial (MaterialRequest $request, $id) {
        $material = Material::find($id);
        $material->description = $request->description;
        $material->quantity = $request->quantity;
        $material->power_plant_id = $request->power_plant_id['id'];
        $material->save();

        return redirect()->route('material.view');
    }

    public function deleteMaterial ($id) {
        $material = Material::find($id);
        $material->delete();

        return redirect()->route('material.view');
    }
```

Figure 5.10 Manage Material Function in Master Data Controller

5.2.4 Recapitulation Controller

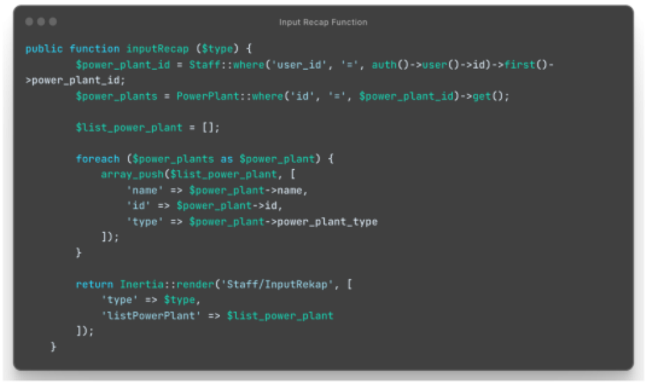
5.2.4.1 View Recap Function



```
    public function viewRecap($id) {
        $recapitulation = Recapitulation::getRecapDetail($id);
        $recapitulation->power_plant_name = PowerPlant::find($recapitulation->power_plant_id)-
>name;
        $recapitulation->power_plant_type = PowerPlant::find($recapitulation->power_plant_id)-
>power_plant_type;

        return Inertia::render('Staff/Rekap', [
            'data' => $recapitulation,
        ]);
    }
```

Figure 5.11 View Recap Function in Recapitulation Controller

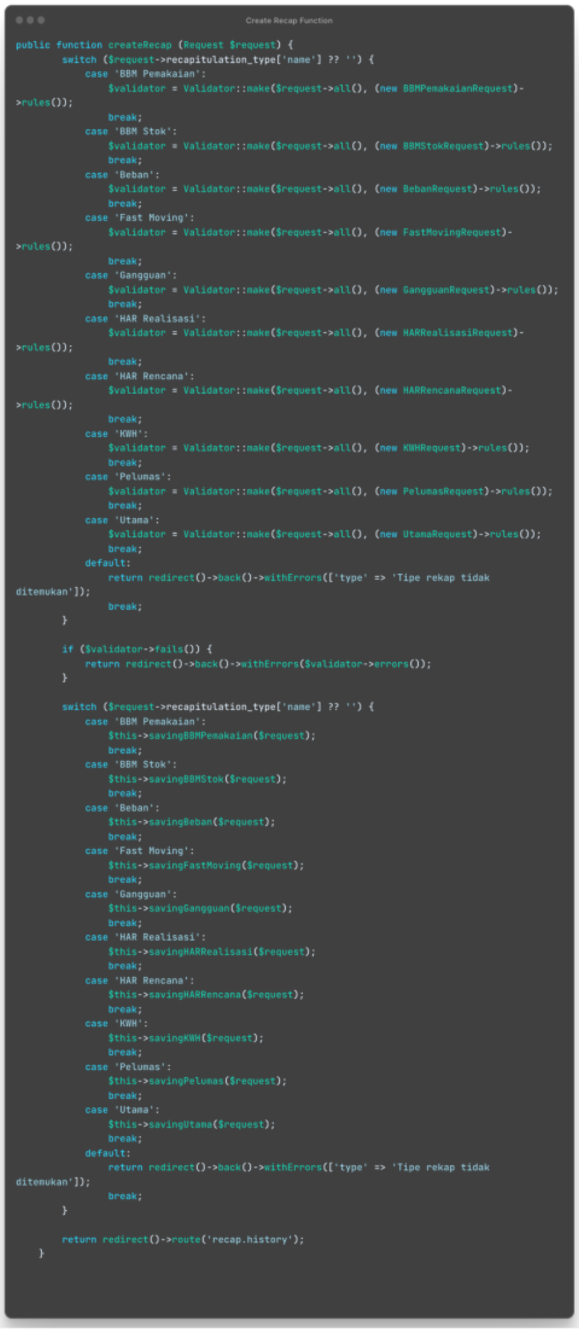
5.2.4.2 Input Recap Function



```
public function inputRecap ($type) {
        $power_plant_id = Staff::where('user_id', '=', auth()->user()->id)->first()-
>power_plant_id;
        $power_plants = PowerPlant::where('id', '=', $power_plant_id)->get();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Staff/InputRekap', [
            'type' => $type,
            'listPowerPlant' => $list_power_plant
        ]);
    }
```

Figure 5.12 Input Recap Function in Recapitulation Controller

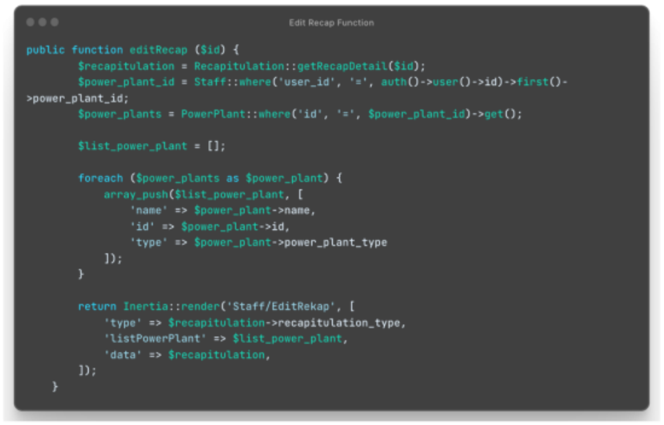
5.2.4.3 Create Recap Function



```
public function createRecap (Request $request) {
        switch ($request->recapitulation_type['name'] ?? '') {
            case 'BBM Pemakaian':
                $validator = Validator::make($request->all(), (new BBMPemakaianRequest)-
>rules());
                break;
            case 'BBM Stok':
                $validator = Validator::make($request->all(), (new BBMStokRequest)->rules());
                break;
            case 'Beban':
                $validator = Validator::make($request->all(), (new BebanRequest)->rules());
                break;
            case 'Fast Moving':
                $validator = Validator::make($request->all(), (new FastMovingRequest)-
>rules());
                break;
            case 'Gangguan':
                $validator = Validator::make($request->all(), (new GangguanRequest)->rules());
                break;
            case 'HAR Realisasi':
                $validator = Validator::make($request->all(), (new HARRealisasiRequest)-
>rules());
                break;
            case 'HAR Rencana':
                $validator = Validator::make($request->all(), (new HARRencanaRequest)-
>rules());
                break;
            case 'KWH':
                $validator = Validator::make($request->all(), (new KWHRequest)->rules());
                break;
            case 'Pelumas':
                $validator = Validator::make($request->all(), (new PelumasRequest)->rules());
                break;
            case 'Utama':
                $validator = Validator::make($request->all(), (new UtamaRequest)->rules());
                break;
            default:
                return redirect()->back()->withErrors(['type' => 'Tipe rekap tidak
ditemukan']);
                break;
        }

        if ($validator->fails()) {
            return redirect()->back()->withErrors($validator->errors());
        }

        switch ($request->recapitulation_type['name'] ?? '') {
            case 'BBM Pemakaian':
                $this->savingBBMPemakaian($request);
                break;
            case 'BBM Stok':
                $this->savingBBMStok($request);
                break;
            case 'Beban':
                $this->savingBeban($request);
                break;
            case 'Fast Moving':
                $this->savingFastMoving($request);
                break;
            case 'Gangguan':
                $this->savingGangguan($request);
                break;
            case 'HAR Realisasi':
                $this->savingHARRealisasi($request);
                break;
            case 'HAR Rencana':
                $this->savingHARRencana($request);
                break;
            case 'KWH':
                $this->savingKWH($request);
                break;
            case 'Pelumas':
                $this->savingPelumas($request);
                break;
            case 'Utama':
                $this->savingUtama($request);
                break;
            default:
                return redirect()->back()->withErrors(['type' => 'Tipe rekap tidak
ditemukan']);
                break;
        }

        return redirect()->route('recap.history');
    }
```

Figure 5.13 Create Recap Function in Recapitulation Controller

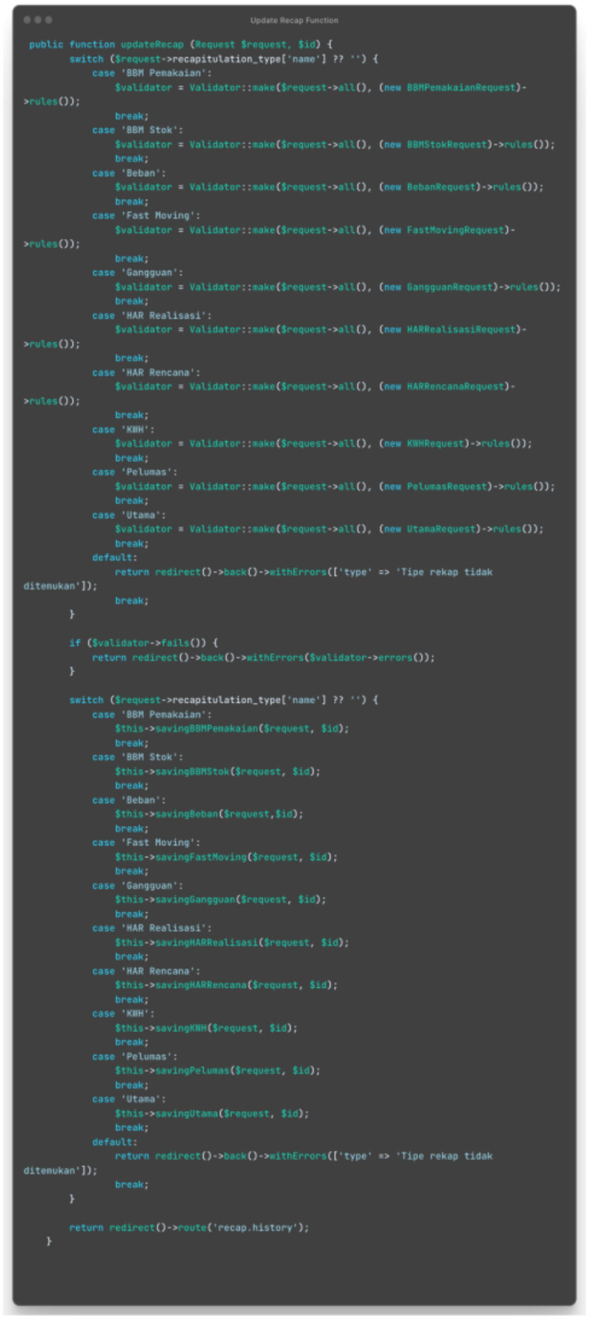
5.2.4.4 Edit Recap Function



```php
public function editRecap ($id) {
        $recapitulation = Recapitulation::getRecapDetail($id);
        $power_plant_id = Staff::where('user_id', '=', auth()->user()->id)->first()-
>power_plant_id;
        $power_plants = PowerPlant::where('id', '=', $power_plant_id)->get();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Staff/EditRekap', [
            'type' => $recapitulation->recapitulation_type,
            'listPowerPlant' => $list_power_plant,
            'data' => $recapitulation,
        ]);
    }
```

Figure 5.14 Edit Recap Function in Recapitulation Controller

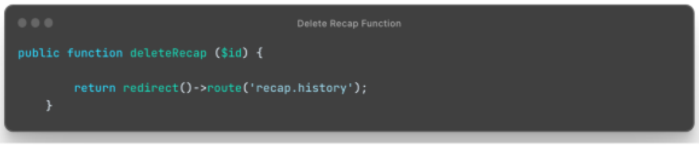
5.2.4.5 Update Recap Function



```
 public function updateRecap (Request $request, $id) {
        switch ($request->recapitulation_type['name'] ?? '') {
            case 'BBM Pemakaian':
                $validator = Validator::make($request->all(), (new BBMPemakaianRequest)-
>rules());
                break;
            case 'BBM Stok':
                $validator = Validator::make($request->all(), (new BBMStokRequest)->rules());
                break;
            case 'Beban':
                $validator = Validator::make($request->all(), (new BebanRequest)->rules());
                break;
            case 'Fast Moving':
                $validator = Validator::make($request->all(), (new FastMovingRequest)-
>rules());
                break;
            case 'Gangguan':
                $validator = Validator::make($request->all(), (new GangguanRequest)->rules());
                break;
            case 'HAR Realisasi':
                $validator = Validator::make($request->all(), (new HARRealisasiRequest)-
>rules());
                break;
            case 'HAR Rencana':
                $validator = Validator::make($request->all(), (new HARRencanaRequest)-
>rules());
                break;
            case 'KWH':
                $validator = Validator::make($request->all(), (new KWHRequest)->rules());
                break;
            case 'Pelumas':
                $validator = Validator::make($request->all(), (new PelumasRequest)->rules());
                break;
            case 'Utama':
                $validator = Validator::make($request->all(), (new UtamaRequest)->rules());
                break;
            default:
                return redirect()->back()->withErrors(['type' => 'Tipe rekap tidak
ditemukan']);
                break;
        }

        if ($validator->fails()) {
            return redirect()->back()->withErrors($validator->errors());
        }

        switch ($request->recapitulation_type['name'] ?? '') {
            case 'BBM Pemakaian':
                $this->savingBBMPemakaian($request, $id);
                break;
            case 'BBM Stok':
                $this->savingBBMStok($request, $id);
                break;
            case 'Beban':
                $this->savingBeban($request,$id);
                break;
            case 'Fast Moving':
                $this->savingFastMoving($request, $id);
                break;
            case 'Gangguan':
                $this->savingGangguan($request, $id);
                break;
            case 'HAR Realisasi':
                $this->savingHARRealisasi($request, $id);
                break;
            case 'HAR Rencana':
                $this->savingHARRencana($request, $id);
                break;
            case 'KWH':
                $this->savingKWH($request, $id);
                break;
            case 'Pelumas':
                $this->savingPelumas($request, $id);
                break;
            case 'Utama':
                $this->savingUtama($request, $id);
                break;
            default:
                return redirect()->back()->withErrors(['type' => 'Tipe rekap tidak
ditemukan']);
                break;
        }

        return redirect()->route('recap.history');
    }
```

Figure 5.15 Update Recap Function in Recapitulation Controller

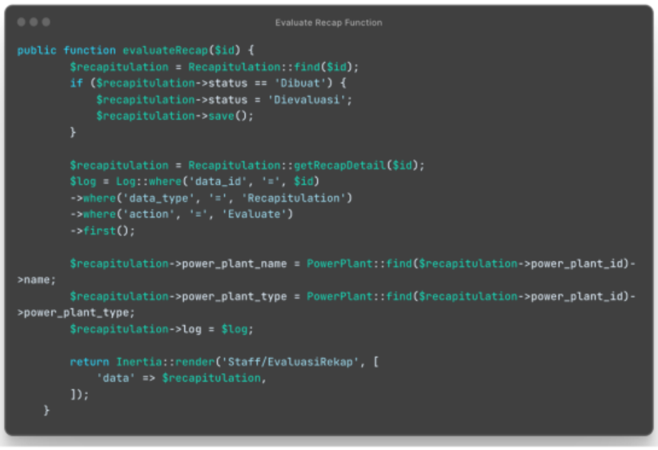
5.2.4.6 Delete Recap Function



```
public function deleteRecap ($id) {

        return redirect()->route('recap.history');
    }
```

Figure 5.16 Delete Recap Function in Recapitulation Controller

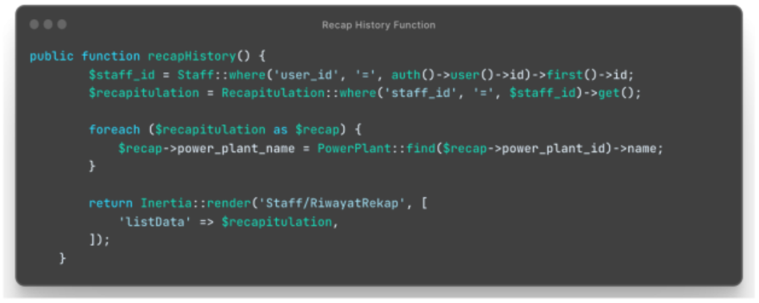
5.2.4.7 Evaluate Recap Function



```
public function evaluateRecap($id) {
        $recapitulation = Recapitulation::find($id);
        if ($recapitulation->status == 'Dibuat') {
            $recapitulation->status = 'Dievaluasi';
            $recapitulation->save();
        }

        $recapitulation = Recapitulation::getRecapDetail($id);
        $log = Log::where('data_id', '=', $id)
        ->where('data_type', '=', 'Recapitulation')
        ->where('action', '=', 'Evaluate')
        ->first();

        $recapitulation->power_plant_name = PowerPlant::find($recapitulation->power_plant_id)-
>name;
        $recapitulation->power_plant_type = PowerPlant::find($recapitulation->power_plant_id)-
>power_plant_type;
        $recapitulation->log = $log;

        return Inertia::render('Staff/EvaluasiRekap', [
            'data' => $recapitulation,
        ]);
    }
```

Figure 5.17 Evaluate Recap Function in Recapitulation Controller

5.2.4.8 Recap History Recap Function



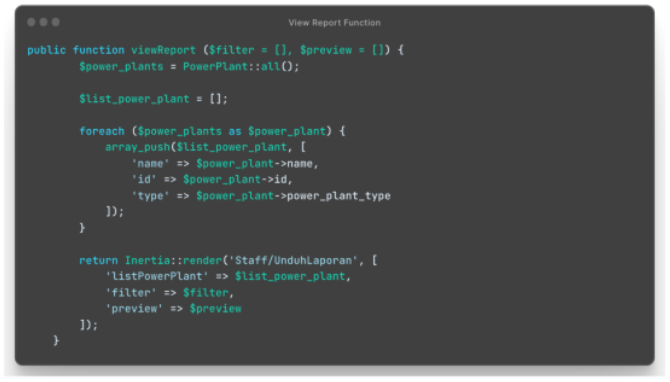
Recap History Function

```
public function recapHistory() {
        $staff_id = Staff::where('user_id', '=', auth()->user()->id)->first()->id;
        $recapitulation = Recapitulation::where('staff_id', '=', $staff_id)->get();

        foreach ($recapitulation as $recap) {
            $recap->power_plant_name = PowerPlant::find($recap->power_plant_id)->name;
        }

        return Inertia::render('Staff/RiwayatRekap', [
            'listData' => $recapitulation,
        ]);
    }
```

Figure 5.18 Recap History Function in Recapitulation Controller

5.2.5 Report Controller

5.2.5.1 View Report Function

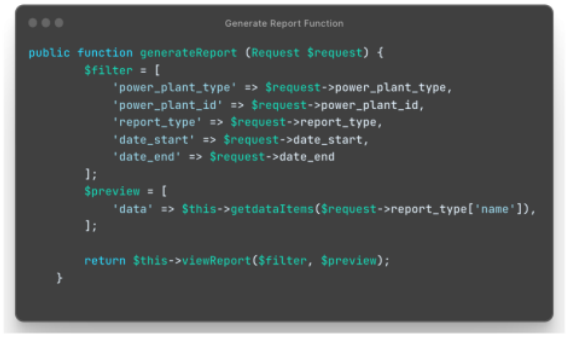


View Report Function

```
public function viewReport ($filter = [], $preview = []) {
        $power_plants = PowerPlant::all();

        $list_power_plant = [];

        foreach ($power_plants as $power_plant) {
            array_push($list_power_plant, [
                'name' => $power_plant->name,
                'id' => $power_plant->id,
                'type' => $power_plant->power_plant_type
            ]);
        }

        return Inertia::render('Staff/UnduhLaporan', [
            'listPowerPlant' => $list_power_plant,
            'filter' => $filter,
            'preview' => $preview
        ]);
    }
```

Figure 5.19 View Report Function in Report Controller

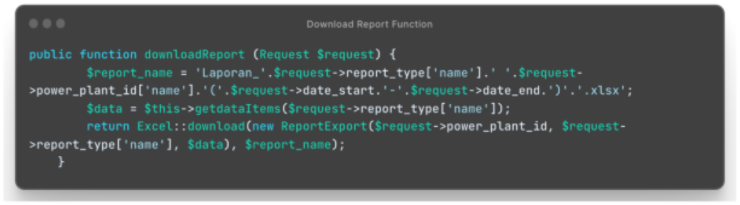
5.2.5.2 Generate Report Function



```php
public function generateReport (Request $request) {
        $filter = [
            'power_plant_type' => $request->power_plant_type,
            'power_plant_id' => $request->power_plant_id,
            'report_type' => $request->report_type,
            'date_start' => $request->date_start,
            'date_end' => $request->date_end
        ];
        $preview = [
            'data' => $this->getdataItems($request->report_type['name']),
        ];

        return $this->viewReport($filter, $preview);
    }
```

Figure 5.20 Generate Report Function in Report Controller

5.2.5.3 Export Report Function



```php
public function downloadReport (Request $request) {
        $report_name = 'Laporan_'.$request->report_type['name'].' '.$request-
>power_plant_id['name'].'('.$request->date_start.'-'.$request->date_end.')'.'.xlsx';
        $data = $this->getdataItems($request->report_type['name']);
        return Excel::download(new ReportExport($request->power_plant_id, $request-
>report_type['name'], $data), $report_name);
    }
```

Figure 5.21 Export Report Function in Report Controller

5.3 Interfaces of System Main Functions

The user interface is an important factor in system development because it functions as an intermediary between the user and the system. User-friendly and easy-to-understand interfaces play an important role in production as they indirectly reduce costs by reducing training time for users. The development of the user interface for this system is carried out using the VueJS Frontend Framework with the help of InertiaJS for integration with the Laravel Framework. We also use PrimeVue, a UI component library for Vue.js which uses TailwindCSS, a utility-first CSS Framework for styling purposes.

5.4 Testing

After the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan was developed, system testing was carried out to test and identify errors. Testing is an important phase in the development of any system so that it can identify code problems before it is fully deployed. System testing is to identify system improvements that need to be made and to identify whether the system being developed is in accordance with the goals and objectives of system development.

5.4.1 Black box Testing

Black box testing is carried out at the module interface. This test does not touch on the performance of the program but emphasizes that the output produced must match the user's needs. This test attempts to identify errors such as incorrect or missing functionality, interface errors, structural or database access errors, presentation errors, and startup or termination errors.

5.4.1.1 System Flow

In black box testing, the system flow refers to the sequence of inputs and outputs within the module or software being tested. It outlines how the system processes inputs and generates corresponding outputs. The specific steps and logic involved in the system flow are not visible to the tester, as they are considered internal to the module. The focus of black box testing is on evaluating the correctness of the outputs produced based on given inputs, without considering the internal workings of the system.

5.4.1.2 Input Output Verification

In black box testing, input-output verification aims to ensure that the output produced by the module matches the expected output for a given set of inputs. Test cases are designed based on the requirements and specifications of the system. The tester provides inputs to the module and checks whether the outputs generated are correct and meet the desired expectations. This verification process helps identify errors such as missing or incorrect functionality, interface issues, and other discrepancies between expected and actual outputs.

5.4.1.3 Error Messages

During black box testing, error messages play an essential role in providing feedback to the user or tester when an error or exceptional condition occurs within the system. Error messages should be informative, clear, and concise, helping users or testers understand the cause of the error and take appropriate action. Black box testing includes verifying that error messages are displayed correctly, contain relevant information, and guide users in resolving the issue. This ensures that users receive meaningful feedback when errors occur, improving the overall user experience and assisting in troubleshooting.

5.4.2 White box Testing

White box testing is a detailed check of the internal program i.e. about the logic flow. White-box testing will test all the logical results of a program whether they are correct or vice versa. Usually involves testing the loop portion of the program to detect errors. User testing was performed to determine if users find the system easy to use while system testing was done to verify the absence of the error in the system. The results of the tests obtained based on the average user satisfaction with the system. A user testing has been conducted to test this system. Table X.X shows the result of the testing.

5.4.3 User Testing

System testing encompasses both user testing and system testing. User testing aims to evaluate whether users can easily use the system, while system testing verifies the absence of errors in the system. The test results are obtained based on the average user satisfaction with the system. A user testing process was conducted specifically for this system, and Table 5.1 displays the testing outcome.

Table 5.1 Testing Result

User Acceptance Testing Result		
No	Question	Average Answer
1	Is the system easy to understand and use?	4/5
2	Is the system interface designed to make it easy for users?	4/5
3	Is the main menu of the system helpful?	4/5
4	Is the system responsive in a timely manner?	3.5/5
5	Is the security of the stored data trustworthy?	3/5
6	Is the system fully utilized?	4/5

The results of the user testing indicate that consumers are content with the system and perceive it as user-friendly, meeting their requirements. Several modules of the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan underwent testing, as documented in Table 5.2 to Table 5.5. The primary goal of this testing phase was to identify any errors that may arise during the user acceptance of the system.

Table 5.2 Admin create a new user

Tester Name	:	Aji (PT PLN (Persero) UP3 Pamekasan engineer staff)
Date	:	24 June 2023
Module	:	Manage Account Module

Instruction	**Predicted Result**	**Result**
1. Login as admin. 2. Choose button manage user account for ‘Staff’. 3. Clicks on the ’Tambah’ button to create new staff acount. 4. Input all required fields on the recapitulation form. 5. Click on the “Simpan” button 6. Sign out.	1. Able to input all required information. 2. System able to give error message for miss or wrong input information. 3. Able to view the current new staff account.	Success

Table 5.3 Operator staff create a new recapitulation

Tester Name	:	Putri (PT PLN (Persero) UP3 Pamekasan staff)
Date	:	24 June 2023
Module	:	Manage Recapitulation Module

Instruction	**Predicted Result**	**Result**
1. Login as operator staff. 2. Choose recapitulation type to be inputted. 3. Clicks on the button of selected recapitulation type to create. 4. Input all required fields on the recapitulation form. 5. Click on the “Simpan” button 6. Sign out.	4. Able to input all required information. 5. System able to give error message for miss or wrong input information. 6. Able to view the current new recapitulation.	Success

Table 5.4 PIC staff evaluate recapitulation

Tester Name	:	Avicenna (PT PLN (Persero) UP3 Pamekasan engineer supervisor)	
Date	:	24 June 2023	
Module	:	Manage Recapitulation Module	
Instruction		**Predicted Result**	**Result**
1. Login as operator staff. 2. Click on the "Permintaan Rekap" navbar button. 3. Click the "Evaluasi" button on one of the recapitulations. 4. Start evaluating the recapitulation whether approved or rejected. 5. Save Evaluation Result. 6. Sign out.		1. Able to view selected recapitulation that need to be evaluated. 2. Able to make correction if there is minor mistake. 3. Able to give the recapitulation evaluation status. 4. Able to update the material stock quantity if the recapitulation type chosen is relate with material.	Success

Table 5.5 PIC staff download report

Tester Name	:	Avicenna (PT PLN (Persero) UP3 Pamekasan engineer supervisor)	
Date	:	24 June 2023	
Module	:	Manage Report Module	
Instruction		**Predicted Result**	**Result**
1. Login as operator staff. 2. Click on the "Unduh Laporan" navbar button. 3. Click the "Evaluasi" button on one of the recapitulations. 4. Input all required fields on the recapitulation form to filter data. 5. Click the "Unduh" button on one of the preview side. 6. Sign out.		1. System show report preview before able to be downloaded. 2. System able to query the data for the report based on the filter inputted. 3. Able to download the report as an excel file	Success

5.5 Chapter Summary

This chapter explains thoroughly the implementation stage of the Power Plants Performance Monitoring System (P3MS) for PT PLN (Persero) UP3 Pamekasan which has explained the use of the software used. Implementation of the system has gone through several processes. The testing phase is a critical stage to identify any errors that occur. Faults are identified and corrected to ensure the end product of the system functions properly. Through testing, system developers can ensure that the system being developed successfully meets user needs and achieves goals. In fact, through testing as well, system development can increase the effectiveness of the system.

CH5 - THORIQULHAQ

ORIGINALITY REPORT

14%	2%	0%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Taylor's Education Group Student Paper	10%
2	Submitted to Universiti Teknologi Malaysia Student Paper	3%
3	su-plus.strathmore.edu Internet Source	1%

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	On		