

BAB III

PELAKSANAAN KERJA PROFESI

3.1 Bidang Kerja

Selama dua bulan, Pratikan terlibat dalam Kegiatan Kerja Profesi (KP) di PT. Tripatra, dengan peran utama sebagai pengembang web. Tugasnya meliputi pengembangan sebuah aplikasi berbasis website untuk pemantauan billing, yang memungkinkan pelacakan pekerjaan yang sedang berlangsung. Untuk menciptakan aplikasi ini, Pratikan memanfaatkan Visual Studio Code sebagai perangkat lunak utama, serta memilih framework Laravel. Sebelum memulai proses pembangunan aplikasi, Pratikan memperdalam pengetahuannya mengenai Laravel, didukung oleh pembimbing kerja di kantor. Fokus utama Pratikan selama di PT. Tripatra adalah merancang aplikasi billing monitoring, yang mencakup pembuatan interface, basis data, dan berbagai fitur yang akan diintegrasikan ke dalam aplikasi website tersebut.

3.2 Pelaksanaan Kerja

Pratikan menjalani beberapa fase dalam menjalankan pekerjaan sebagai bagian dari pelaksanaan profesi mereka. Tahapan awal melibatkan persiapan kerja profesi, di mana pratikan mengajukan surat permohonan kepada perusahaan terkait. Selanjutnya, surat konfirmasi diterima oleh pratikan dari perusahaan, yang menandai langkah berikutnya dalam proses pelaksanaan kerja profesi.

Pelaksanaan kerja profesi ini, pratikan mendapatkan pekerjaan berupa pembuatan aplikasi CRUD, mengaplikasikan Frond End, membuat skema ERD dan mengaplikasikan Back End. Berikut adalah time linanya:

Kegiatan	Oktober		November				Desember	
	W3	W4	W1	W2	W3	W4	W1	W2
Perkenalan								
Pembuatan CRUD								
Analisis Kebutuhan								
FrontEnd Billing Monitoring								
Skema ERD								
BackEnd Billing Monitoring								

Table diatas merupakan time line yang dikerjakan oleh Pratikan selama kerja profesi di PT. Tripatra. Berikut merupakan rincian yang dilakukan pratikan selama magang di PT. Tripatra.

3.2.1 Perkenalan

Awal masuk kerja profesi, Pratikan diperkenalkan terhadap lingkungan sekitar. Dimulai dari semua staff devisi sampai head of system solusion. Pada perkenalan ini juga Pratikan mulai berbaur pada semua orang pada devisi *It System Solusion*.

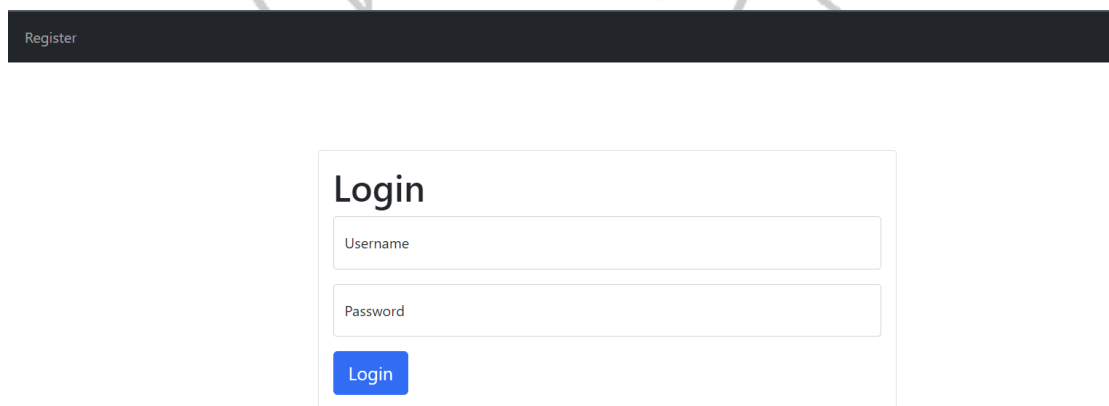
3.2.2 Pembuatan CRUD

Setelah perkenalan, Pratikan mendapatkan sebuah tugas yaitu pembuatan aplikasi CRUD. Aplikasi CRUD itu sendiri merupakan aplikasi *create, read, update* dan *delete*. Pratikan membuat aplikasi CRUD agar mempermudah ketika pembuatan aplikasi *billing monitoring*. Konserp

aplikasi CRUD yang dibuat kali ini seperti *inventory*. Setiap user mempunyai *inventory* yang berbeda-beda.

3.2.2.1 Tampilan Aplikasi CRUD

Pada aplikasi CRUD, mempunyai fitur *login*, *register*, *list*, tambah dan logout. Berikut tampilan aplikasi CRUD.



The image shows a screenshot of a web application interface. At the top, there is a dark navigation bar with the word 'Register' in white text. Below the navigation bar, the main content area is white. In the center, there is a 'Login' form. The form has a title 'Login' in bold black text. Below the title, there are two input fields: 'Username' and 'Password'. Below the 'Password' field, there is a blue button with the text 'Login' in white.

Gambar 3. 1 Tampilan Aplikasi CRUD

Pada gambar 3.1, merupakan tampilan aplikasi CRUD pada halaman *login*. Pada halaman ini user dimohon untuk memasukkan *username* dan *password*. Jika user tidak mempunyai sebuah akun, maka user bisa mengakses fitur *register* pada *navbar*.

Login

Register

Nama

Username

Email

Password

Tambah

Gambar 3. 2 Tampilan Register Aplikasi CRUD

Pada gambar 3.2, merupakan tampilan *register* pada aplikasi CRUD. Pada halaman ini user dapat mendaftarkan sebuah akun dengan cara mengisi nama, *username*, *email* dan *password*.

List [Tambah](#) [Logout](#) Hai, white

List

No	Item	Jumlah	Aksi
1	Contoh	1	Ubah Hapus

Gambar 3. 3 Tampilan List Aplikasi CRUD

Pada gambar 3.3, merupakan tampilan *list* pada aplikasi CRUD. Pada halaman ini aplikasi akan menampilkan apa saja yang sudah *user* inputkan. Pada halaman ini terdapat juga 2 (dua) buah fitur yaitu *ubah* dan *hapus*. Fitur tersebut mempermudah user untuk mengubah ataupun menghapus apa saja yang sudah diinputkan oleh *user*.

List Tambah Logout Hai, white

Tambah

Item

Jumlah

Gambar 3. 4 Tampilan Tambah Aplikasi CRUD

Pada gambar 3.4, merupakan halaman tambah pada aplikasi CRUD. Pada halaman ini *user* dapat menambah sebuah barang. Halaman ini menyajikan 2 (dua) buah *field*, yaitu item dan jumlah.

3.2.3 Analisis Kebutuhan

Sebelum Pratikan membuat sebuah aplikasi *invoice monitoring*, pratikan menganalisis kebutuhan terlebih dahulu. Menganalisis kebutuhan ini bertujuan agar mempermudah pekerjaan yang akan dikerjakan oleh Pratikan.

3.2.3.1 Spesifikasi *Hardware*

Spesifikasi perangkat keras atau *hardware* yang digunakan dalam pengembangan aplikasi *billing monitoring* sebagai berikut :

- a. *Processor* Intel Core i3-11155G4
- b. RAM 8GB

3.2.3.2 Spesifikasi *Software*

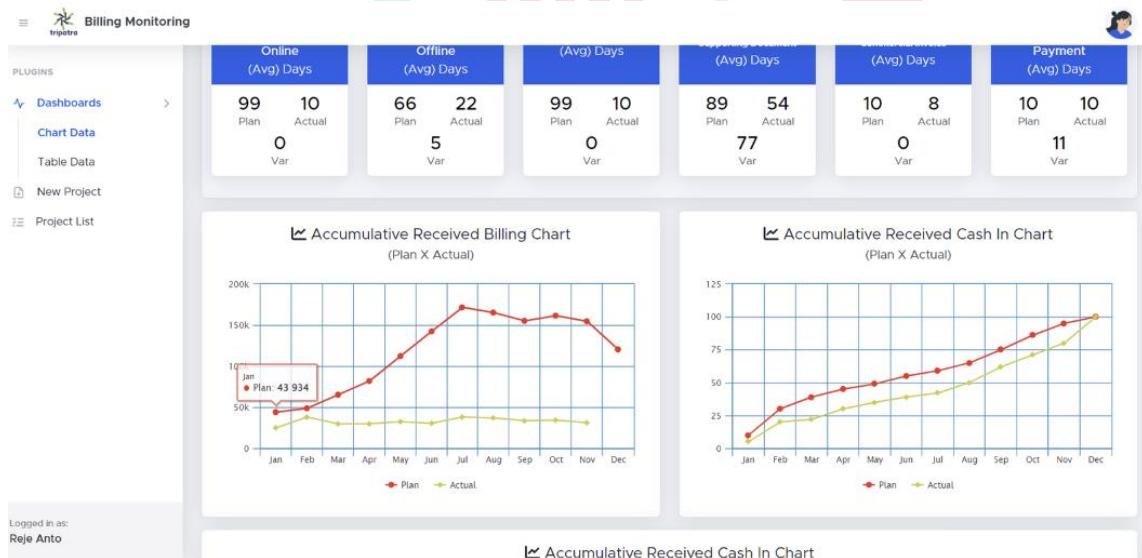
Spesifikasi perangkat lunak atau *software* yang digunakan pratikan dalam pengembangan aplikasi *billing monitoring* sebagai berikut :

- a. Sistem operasi : Microsoft Windows 11
- b. Web browser

- c. Visual Studio Code
- d. Laravel
- e. Composer
- f. Figma
- g. Navicat
- h. PostgreSQL
- i. PHP

3.2.4 FrontEnd *Billing Monitoring*

Pada aplikasi *billing monitoring*, Pratikun membuat sebuah tampilan yang dapat mempermudah pengguna dalam menggunakan aplikasi tersebut. Aplikasi *billing monitoring* mempunyai sebuah navbar yaitu dashboard - chart data, dashboard - table data, *new project*, *project list* dan *project list - milestone* Berikut merupakan tampilan yang telah dibuat pada aplikasi *billing monitoring*:



Gambar 3. 5 Tampilan Dashboard Chart Aplikasi *Billing Monitoring*

Gambar 3.5 merupakan tampilan dashboard – chart data pada aplikasi *billing monitoring*. Pada halaman tersebut *website* akan menampilkan sebuah informasi berupa angka dan *grafik* yang terhubung oleh database. *Website* menampilkan *progress claim online*, *progress claim offline*, *performa invoice*, *readiness of supporting document*, *submission of commercial invoice* dan *received payment*. Untuk tampilan *grafik* juga menampilkan *accumulative received billing chart* dan *accumulative received cash in chart*.

Billing Monitoring Table
Thursday · 7 December 2023

Year: All | Service Type: All | Project: All

Show 10 entries | Search: []

Service	Project	Scenario	Jan-23	Feb-23	Mar-23	Apr-23	Mei-23	Jun-23	Jul-23	Aug-23	Sep-23	Okt-23	Nov-23	Des-23	Action
Engineering	TPPI	Actual	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	[edit] [delete]
Engineering	TPPI	Forecast	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	204.987	[edit] [delete]
ASL	Salak	Plan	-	-	-	-	-	-	-	-	-	-	-	-	[edit] [delete]

Showing 1 to 10 of 12 entries | Previous 1 2 Next

Gambar 3. 6 Tampilan Dashboard Table Aplikasi Billing Monitoring

Gambar 3.6 merupakan tampilan halaman pada dashboard – table. Pada halaman tersebut *website* menampilkan sebuah table yang terhubung pada database. Halaman tersebut juga mempunyai sebuah fitur *edit* dan *delete*.

The screenshot displays a three-step wizard for creating a new project. Step 1, 'Project Setup', includes the following fields:

- Service Type:** A dropdown menu with the placeholder text 'Open this select menu'.
- Project Name:** A text input field.
- PIC (Cost Control Team):** A dropdown menu with the placeholder text 'Open this select menu'.
- Project Periode:** A date range selector with the placeholder text 'Select date range...'.
- Total Project Value:** A numeric input field containing '0'.
- Currency:** A numeric input field containing '0'.

Step 3, 'Billing Plan', shows a progress indicator for 'Billing Plan Data (0) 0%' and a 'Submit' button. A message below the progress indicator states: 'There is no billing plan data added.'

Gambar 3. 7 Tampilan New Project Aplikasi Billing Monitoring

Gambar 3.7 merupakan tampilan pada *new project*. Pada *new project* Pratikan menggunakan tampilan *wizard*. *Wizard* mempunyai sebuah fungsi yaitu menampilkan beberapa *step* untuk mengisi suatu *form*. Pada halaman ini terdapat 3 *step*, yaitu *project setup*, *project informations* dan *billing plan*. Pada *step project setup* terdapat sebuah *field* yaitu *service type*, *project name*, *PIC*, *project periode*, *total project value* dan *currency*. Untuk *field total project value* dan *currency* user hanya dapat mengisi sebuah angka dan titik.

Step 2

Billing Scheme

Pilih ▼

Contract Type

Pilih ▼

Description

Billing Value

Billing % Cummulative Billing %

0% 0%

Completion Date

mm/dd/yyyy 📅

Previous Next

Gambar 3. 8 Tampilan New Project Aplikasi Billing Monitoring

Gambar 3.8 merupakan tampilan *new project* pada *step 2* yaitu *project informations*. Pada halaman tersebut *website* akan menampilkan sebuah *form* dengan nama *billing scheme*, *contract type*, *description*, *billing value*, *billing %*, *cumulative billing %* dan *completion date*. Pada *form billing value user* hanya dapat mengisi sebuah angka dan titik. *Billing %* dan *cumulative billing % field* dibuat menjadi *disable* karena *field* tersebut terisi otomatis melalui perhitungan pada *billing value* dan *total project value*.

Step 3

Progress Claim Online (By System)

How many day(s) form submission to approval

Submission Date Approval Date

mm/dd/yyyy mm/dd/yyyy

Progress Claim Online (Hardcopy Document)

How many day(s) form submission to approval

Submission Date Approval Date Upload to System

mm/dd/yyyy mm/dd/yyyy mm/dd/yyyy

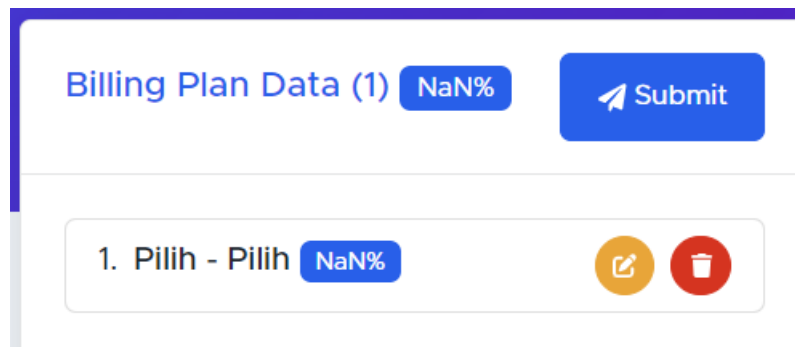
Proforma Invoice

How many day(s) form submission to approval

Submission Date Approval Date

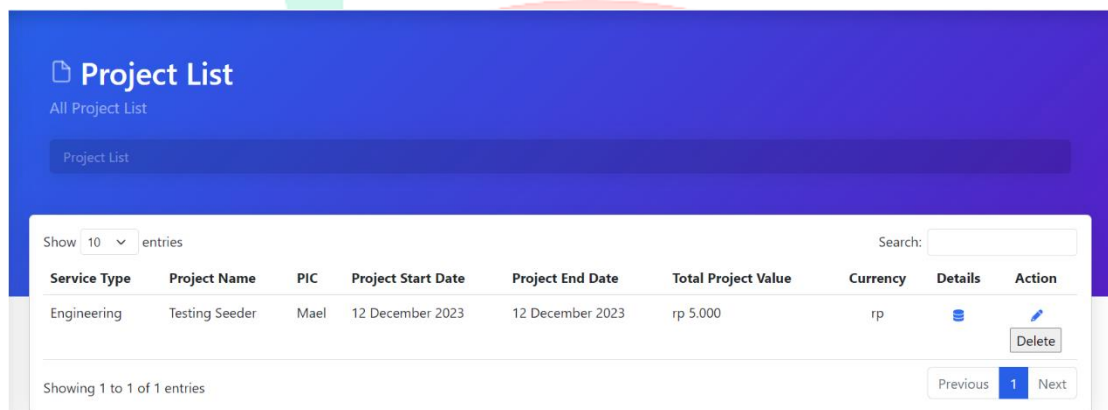
Gambar 3. 9 Tampilan New Project Aplikasi Billing Monitoring

Gambar 3.9 merupakan tampilan *new project* pada *step 3* yaitu *billing plan*. Pada halaman ini *website* menampilkan sebuah *field* yaitu *progress claim online*, *progress claim offline*, *proforma invoice*, *upload proforma invoice in company's system*, *readiness of supporting document*, *submission of commercial invoice* dan *received payment*. Halaman ini *user* dapat tidak mengisi salah satu maupun semua *form* yang ada.



Gambar 3. 10 Tampilan Billing Plan Data Aplikasi Billing Monitoring

Gambar 3.10 berada di halaman *new project*. *Billing plan data* ini berfungsi untuk menampilkan list-list yang sudah diisi *user* pada *step 2* dan *3*. Pada fitur ini, *user* dapat mengedit apa saja yang sudah diisi dan dapat menghapus.



Gambar 3. 11 Tampilan Project List Aplikasi Billing Monitoring

Gambar 3.11 merupakan tampilan pada halaman *project list*. Pada halaman ini *user* dapat melihat apa saja yang sudah mereka isi. Pada halaman ini terhubung pada *database*. Halaman tersebut menampilkan *project* yang diisi pada *step 1* (satu). Terdapat 3 (tiga) buah fitur pada halaman ini, yaitu *edit*, *hapus* dan *detail*.

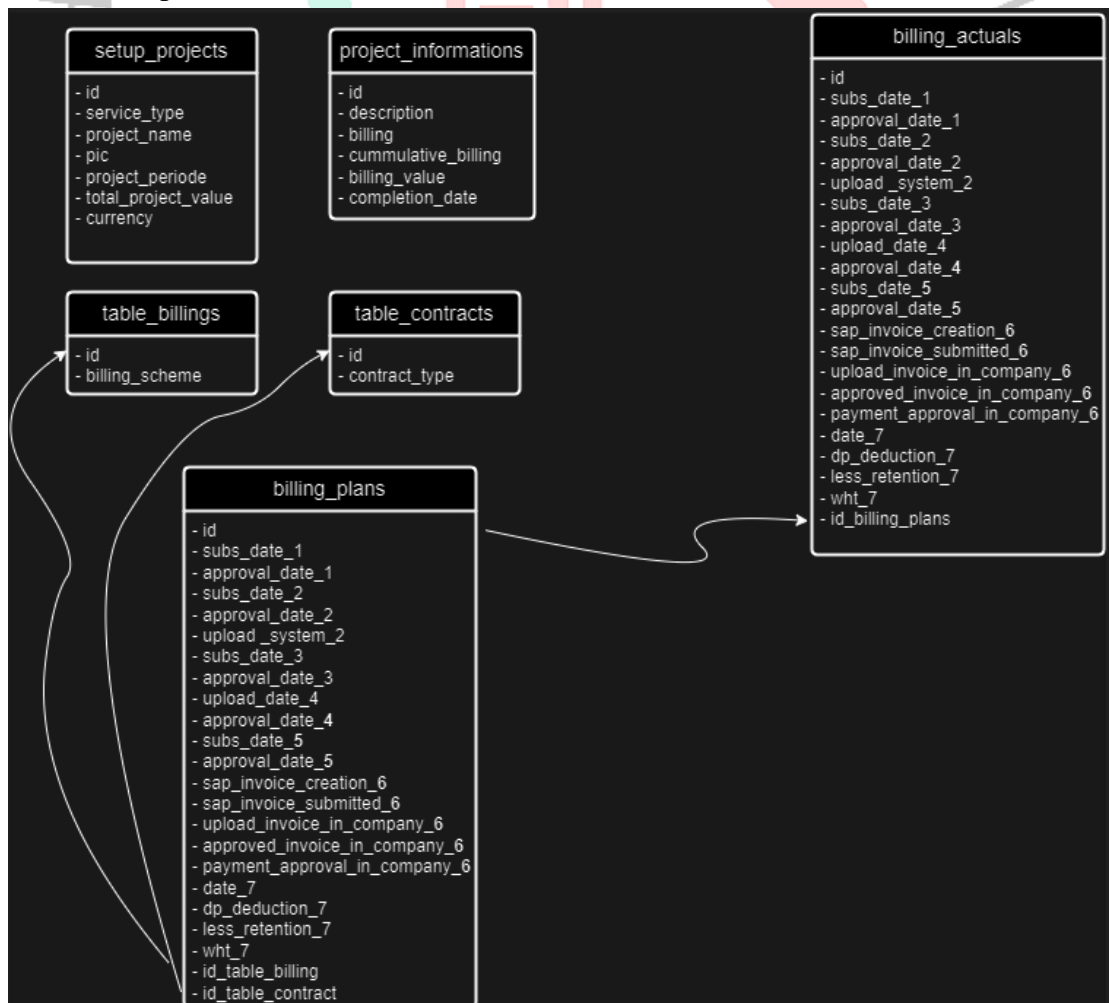
Plan Actual Actual										
Billing Scheme							Progress Claim Online (by System)			Action
Contract Type	Description	Billing Perc.	Billing Cum. Perc.	Billing Value	Currency	Submission Date	Approval Date	Submissi		
Showing 0 to 0 of 0 entries										

Gambar 3.12 Tampilan Detail Project List Aplikasi Billing Monitoring

Gambar 3.12 merupakan halaman pada *detail* di *project list*. Pada halaman ini *website* akan menampilkan yang sudah diisi oleh *user* pada *step* 2 dan 3. Pada halaman ini terdapat banyak *table*.

3.2.5 Skema ERD

Dalam pembuatan *website billing monitoring*, Pratkan membuat sebuah skema *ERD*. Dimana skema ini lah yang saling terhubung satu sama lain pada *table* lain. Berikut *skema ERD*:



Gambar 3.13 Skema ERD Aplikasi Billing Monitoring

Gambar 3.13 merupakan skema *ERD* pada aplikasi *billing monitoring*. Pada skema ini, Pratkan membutuhkan setidaknya 6 table. Beberapa table saling terhubung satu sama lain. Hal ini disebabkan untuk hasil yang sempurna. Fungsi dari skema *ERD* ini untuk membuat sebuah rancangan database agar pembuat dapat memahami struktur dari database tersebut.

3.2.6 BackEnd *Billing Monitoring*

Aplikasi *billing monitoring* mempunyai sebuah fungsi *backend*. *Backend* berfungsi sebagai penghubung antara *website* dengan sebuah database. Berikut merupakan *backend* dari aplikasi *billing monitoring*.

```
public function store(Request $request)
{
    DB::beginTransaction();
    try {
        // 1. Store data project setup
        $project_setup_id = $this->storeProjectSetup($request);
        // 2. Store data project information
        $project_information_ids = $this->storeProjectInformation($request);
        // 3. Store data billings
        $billing_plan_ids = $this->storeBillingPlan($request);
        // 4. map id into single array for storing to trxProjectBilling
        $data = [];
        $arr_length = count($project_information_ids);
        for ($i = 0; $i < $arr_length; $i++) {
            array_push($data, [
                'project_setup_id' => $project_setup_id,
                'project_information_ids' => $project_information_ids[$i],
                'billing_plan_ids' => $billing_plan_ids[$i],
                'created_by' => Auth::user()->id,
                'updated_by' => Auth::user()->id,
            ]);
        }
        // 5. Store data trx
        $this->storeTrxProjectBilling($data);
        DB::commit();
    } catch (\Throwable $th) {
        DB::rollBack();
        return json_encode("Error : " . $th->getMessage());
    }

    return json_encode('success');
}
```

Gambar 3. 14 Function Store Aplikasi Billing Monitoring

Gambar 3.14 merupakan sebuah *function store* pada *controllers-form*. Pada *function store* mempunyai sebuah fungsi untuk menangani penyimpanan data terkait beberapa table. Pertama dimulainya sebuah transaksi *database*. Setelah itu menyimpan data mengembalikan serangkaian id pada setiap *store*. Jika terjadinya sebuah kesalahan ketika memasukkan input, maka akan muncul sebuah pesan “Error : ” pada aplikasi.

```
private function storeProjectSetup(Request $request)
{
    $project_periode = explode("-", $request->project_setup['project_periode']);
    $start_date = new Carbon($project_periode[0]);
    $end_date = new Carbon($project_periode[1]);

    // 1. Store data project setup
    $project_setups_id = ProjectSetup::create([
        'currency' => $request->project_setup['currency'],
        'pic' => $request->project_setup['pic'],
        'project_name' => $request->project_setup['project_name'],
        'service_type' => $request->project_setup['service_type'],
        'start_date' => $start_date->toDateString(),
        'end_date' => $end_date->toDateString(),
        'total_project_value' => $request->project_setup['total_project_value'],
    ]);

    return $project_setups_id->id;
}
```

Gambar 3. 15 Function *storeProjectSetup* Aplikasi Billing Monitoring

Gambar 3.15 merupakan sebuah *function storeProjectSetup* pada *controllers-form*. Pada *function* ini berfungsi untuk memasukkan sebuah data yang sudah diinput oleh user kedalam sebuah *database*. Pada *function* ini memerlukan sebuah “Carbon” yang berfungsi untuk memisahkan input tanggal menjadi *start_date* dan *end_date*. *Function* ini terdapat sebuah fungsi untuk menghubungkan sebuah *field* pada *frontend* dan *database*.

```

private function storeBillingPlan(Request $request)
{
    $billing_plans = $request->billing_plan;
    $billing_plan_ids = [];
    foreach ($billing_plans as $bp) {
        $billing_plan = Billings::create([
            'plan_date_pcos_submission' => $bp['plan_date_pcos_submission'],
            'plan_date_pcos_approval' => $bp['plan_date_pcos_approval'],
            'plan_date_pcohd_submission' => $bp['plan_date_pcohd_submission'],
            'plan_date_pcohd_approval' => $bp['plan_date_pcohd_approval'],
            'plan_date_pcohd_upload_system' => $bp['plan_date_pcohd_upload_system'],
            'plan_date_pi_submission' => $bp['plan_date_pi_submission'],
            'plan_date_pi_approval' => $bp['plan_date_pi_approval'],
            'plan_date_pics_upload' => $bp['plan_date_pics_upload'],
            'plan_date_pics_approval' => $bp['plan_date_pics_approval'],
            'plan_date_rsd_submission' => $bp['plan_date_rsd_submission'],
            'plan_date_rsd_approval' => $bp['plan_date_rsd_approval'],
            'plan_date_sci_sap_invoice_creation' => $bp['plan_date_sci_sap_invoice_creation'],
            'plan_date_sci_sap_invoice_submitted' => $bp['plan_date_sci_sap_invoice_submitted'],
            'plan_date_sci_upload_invoice' => $bp['plan_date_sci_upload_invoice'],
            'plan_date_sci_approved_invoice' => $bp['plan_date_sci_approved_invoice'],
            'plan_date_sci_payment_approval' => $bp['plan_date_sci_payment_approval'],
            'plan_date_received_payment' => $bp['plan_date_received_payment'],
            'plan_dp_deduction' => $bp['plan_dp_deduction'],
            'plan_less_retention' => $bp['plan_less_retention'],
            'plan_wht' => $bp['plan_wht']
        ]);
        array_push($billing_plan_ids, $billing_plan->id);
    }
    return $billing_plan_ids;
}

```

Gambar 3. 16 Function BillingPlan Aplikasi Billing Monitoring

Gambar 3.16 merupakan sebuah *function BillingPlan* pada *controller-form*. Pada gambar di atas merupakan sebuah fungsi untuk menghubungkan *frontend* pada *form* ke *database*. Hal ini bertujuan agar yang diinputkan oleh user dapat dimasukan secara otomatis kedalam *database*.

```

private function storeTrxProjectBilling(array $data)
{
    foreach ($data as $data) {
        TrxProjectBilling::create([
            'project_setups_id' => $data['project_setup_id'],
            'project_informations_id' => $data['project_information_ids'],
            'billings_id' => $data['billing_plan_ids'],
            'created_by' => $data['created_by'],
            'updated_by' => $data['updated_by'],
        ]);
    }
}

```

Gambar 3. 17 Function store TrxProjectBilling Aplikasi Billing Monitoring

Gambar 3.17 merupakan sebuah *function storeTrxProjectBilling* pada *controller-form*. Pada *function* mempunyai fungsi yaitu mengambil sebuah id dari table lain yang akan bertujuan untuk menampilkan sebuah tampilan berdasarkan id yang sudah terhubung.

```

class AuthController extends Controller
{
    //
    public function index()
    {
        $data = ['title' => 'Login'];
        return view('layouts.guest', $data);
    }

    public function redirect()
    {
        return Socialite::driver('azure')->redirect();
    }
}

```

Gambar 3. 18 Function Index dan Redirect Aplikasi Billing Monitoring

Gambar 3.18 merupakan sebuah *function index* dan *redirect* pada *AuthController*. *Function* ini mengindikasikan bahwa menggunakan *Laravel Socialite* untuk mengimplementasikan otentikasi dengan penyedia layanan *Azure Active Directory*. Pada *function* ini menyediakan pengalihan ke halaman login *Azure AD* untuk mengotentikasikan pengguna.


```

public function callback()
{
    try {
        $user_ad = Socialite::driver('azure')->stateless()->user();
        $email = strtolower($user_ad->email);
    } catch (Exception $e) {
        return redirect()->back();
    }
    $success = [
        'Id' => $user_ad->user['id'],
        'Name' => $user_ad->user['displayName'],
        'Email' => $user_ad->user['mail'],
        'JobTitle' => $user_ad->user['jobTitle'],
    ];
    $user = User::where('email', $email)->first();
    // Jika Tidak ada user
    if (!$user) {
        // Create user baru
        $user = User::updateOrCreate(
            ['email' => $email],
            [
                'name' => $user_ad->name,
                'email' => $email,
                'ad_id' => $user_ad->id,
                'job_title' => $user_ad->user['jobTitle'],
                'email_verified_at' => now(),
                'role_id' => 1
            ]
        );
    }
    Auth()->login($user, true);
    session()->regenerate();
    session()->put('email', $email);
    return redirect(route('dashboard.index'));
}

```

Gambar 3. 19 Function callback Aplikasi Billing Monitoring

Gambar 3.19 merupakan sebuah *function callback* pada *AuthController*. *Function* ini bertujuan untuk menggunakan *Socialite* untuk mendapatkan data pengguna setelah proses otentikasi dengan *Azure*. Jika terjadi sebuah kesalahan pada otentikasi, *user* akan diarahkan kembali ke halaman login. Setelah itu terdapat proses data pengguna *Azure* disimpan dalam variable “*\$success*” untuk digunakan lebih lanjut. Setelah itu sistem akan mencari data *user* berdasarkan alamat email dari *database*. Setelah ditemukannya data *user* maka akan *user* akan diarahkan ke halaman *dashboard*.

```

public function logout(Request $request)
{
    auth('web')->logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();
    return redirect('/');
}

```

Gambar 3. 20 Function logout Aplikasi Billing Monitoring

Gambar 3.20 merupakan sebuah *function* *logout*. Pada *function* ini bertujuan untuk mengakhiri sesi dari *website* dan mengeluarkan *account*.

```

$data_project_setup = [
    [
        'service_type' => 'Engineering',
        'project_name' => 'Seeder',
        'pic' => 'reje',
        'start_date' => '2023-12-7',
        'end_date' => '2023-12-30',
        'total_project_value' => '1000',
        'currency' => 'usd'
    ],
    [
        'service_type' => 'Engineering',
        'project_name' => 'Testing Seeder',
        'pic' => 'Mael',
        'start_date' => '2023-12-12',
        'end_date' => '2023-12-12',
        'total_project_value' => '5000',
        'currency' => 'rp'
    ],
];
foreach ($data_project_setup as $data) {
    ProjectSetup::create([
        'service_type' => $data['service_type'],
        'project_name' => $data['project_name'],
        'pic' => $data['pic'],
        'start_date' => $data['start_date'],
        'end_date' => $data['end_date'],
        'total_project_value' => $data['total_project_value'],
        'currency' => $data['currency'],
    ]);
}

```

Gambar 3. 21 Seeder ProjectSetup Aplikasi Billing Monitoring

Gambar 3.21 merupakan sebuah *seeder* untuk ProjectSetup. *Seeder* ini berfungsi untuk mengisi secara otomatis tanpa melakukan sebuah proses input yang terdapat pada *form*. Pada gambar diatas menunjukkan bahwa ProjectSetup diisi secara otomatis dan menghasilkan 2 buat row.

```
$data_project_informations = [
  [ ...
  ],
  [
    'billing_scheme' => 'Milestone',
    'contract_type' => 'Lumpsum',
    'description' => 'seeder',
    'billing_value' => '5000',
    'billing_percentage' => '',
    'billing_cummulative_percentage' => '',
    'completion_date' => '2023-12-29'
  ],
];
foreach ($data_project_informations as $data) {
  ProjectInformations::create(
    [
      'billing_scheme' => $data['billing_scheme'],
      'contract_type' => $data['contract_type'],
      'description' => $data['description'],
      'billing_value' => $data['billing_value'],
      'billing_percentage' => $data['billing_percentage'],
      'billing_cummulative_percentage' => $data['billing_cummulative_percentage'],
      'completion_date' => $data['completion_date'],
    ]
  );
}
```

Gambar 3. 22 *Seeder ProjectInformations Aplikasi Billing Monitoring*

Gambar 3.22 merupakan sebuah *seeder* untuk ProjectInformations. *Seeder* ini berfungsi untuk mengisi secara otomatis tanpa melakukan sebuah proses input yang terdapat pada *form*. Pada gambar diatas menunjukkan bahwa ProjectInformations diisi secara otomatis dan menghasilkan 2 buat row.

```

$data_billings = [
  [ ...
  ],
  [
    'plan_date_pcos_submission' => '2023-12-27', 'plan_date_pcos_approval' => '2023-12-27',
    'plan_date_pcohd_submission' => '2023-12-27', 'plan_date_pcohd_approval' => '2023-12-27',
    'plan_date_pcohd_upload_system' => '2023-12-27', 'plan_date_pi_submission' => '2023-12-27',
    'plan_date_pi_approval' => '2023-12-27', 'plan_date_pics_upload' => '2023-12-27',
    'plan_date_pics_approval' => '2023-12-27', 'plan_date_rsd_submission' => '2023-12-27',
    'plan_date_rsd_approval' => '2023-12-27', 'plan_date_sci_sap_invoice_creation' => '2023-12-27',
    'plan_date_sci_sap_invoice_submitted' => '2023-12-27', 'plan_date_sci_upload_invoice' => '2023-12-27',
    'plan_date_sci_approved_invoice' => '2023-12-27', 'plan_date_sci_payment_approval' => '2023-12-27',
    'plan_date_received_payment' => '2023-12-27', 'plan_dp_deduction' => '12',
    'plan_less_retention' => '11', 'plan_wht' => '10'
  ],
];
foreach ($data_billings as $data) {
  Billings::create(
    [
      'plan_date_pcos_submission' => $data['plan_date_pcos_submission'], 'plan_date_pcos_approval' => $data['plan_date_pcos_approval'],
      'plan_date_pcohd_submission' => $data['plan_date_pcohd_submission'], 'plan_date_pcohd_approval' => $data['plan_date_pcohd_approval'],
      'plan_date_pcohd_upload_system' => $data['plan_date_pcohd_upload_system'], 'plan_date_pi_submission' => $data['plan_date_pi_submission'],
      'plan_date_pi_approval' => $data['plan_date_pi_approval'], 'plan_date_pics_upload' => $data['plan_date_pics_upload'],
      'plan_date_pics_approval' => $data['plan_date_pics_approval'], 'plan_date_rsd_submission' => $data['plan_date_rsd_submission'],
      'plan_date_rsd_approval' => $data['plan_date_rsd_approval'], 'plan_date_sci_sap_invoice_creation' => $data['plan_date_sci_sap_invoice_creation'],
      'plan_date_sci_sap_invoice_submitted' => $data['plan_date_sci_sap_invoice_submitted'], 'plan_date_sci_upload_invoice' => $data['plan_date_sci_upload_invoice'],
      'plan_date_sci_approved_invoice' => $data['plan_date_sci_approved_invoice'], 'plan_date_sci_payment_approval' => $data['plan_date_sci_payment_approval'],
      'plan_date_received_payment' => $data['plan_date_received_payment'], 'plan_dp_deduction' => $data['plan_dp_deduction'],
      'plan_less_retention' => $data['plan_less_retention'], 'plan_wht' => $data['plan_wht'],
    ]
  );
}

```

Gambar 3. 23 Seeder Billings Aplikasi Billing Monitoring

Gambar 3.23 merupakan sebuah *seeder* untuk Billings. *Seeder* ini berfungsi untuk mengisi secara otomatis tanpa melakukan sebuah proses input yang terdapat pada *form*. Pada gambar diatas menunjukkan bahwa Billings diisi secara otomatis dan menghasilkan 2 buat row.

```

$data_trxprojectbilling = [
  [
    'project_setups_id' => 1,
    'project_informations_id' => 1,
    'billings_id' => 1,
    'created_by' => 1,
    'updated_by' => 1
  ],
];
foreach ($data_trxprojectbilling as $data) {
  TrxProjectBilling::create([
    'project_setups_id' => $data['project_setups_id'],
    'project_informations_id' => $data['project_informations_id'],
    'billings_id' => $data['billings_id'],
    'created_by' => $data['created_by'],
    'updated_by' => $data['updated_by'],
  ]);
}

```

Gambar 3. 24 Seeder TrxProjectBilling Aplikasi Billing Monitoring

Gambar 3.24 merupakan sebuah *seeder* untuk TrxProjectBilling. *Seeder* ini berfungsi untuk mengisi secara otomatis tanpa melakukan sebuah proses input yang terdapat pada *form*. Pada gambar diatas menunjukkan bahwa TrxProjectBilling diisi secara otomatis dan menghasilkan 2 buat row.

```

class TrxProjectBilling extends Model
{
    use HasFactory;
    protected $guarded = ['id'];

    public function getAllMilestonesByProject($project_setups_id)
    {
        return $this->select(
            'trx_project_billings.id as id_trx',
            'project_setups.currency',
            'project_informations.billing_scheme',
            'project_informations.contract_type',
            'project_informations.description',
            'project_informations.billing_value',
            'project_informations.billing_percentage',
            'project_informations.billing_cumulative_percentage',
            'project_informations.completion_date',
            'billings.*'
        )
        ->join('project_setups', 'project_setups.id', '=', 'trx_project_billings.project_setups_id')
        ->join('project_informations', 'project_informations.id', '=', 'trx_project_billings.project_informations_id')
        ->join('billings', 'billings.id', '=', 'trx_project_billings.billings_id')
        ->where('project_setups_id', $project_setups_id)
        ->get();
    }
}

```

Gambar 3. 25 Model TrxProjectBilling Aplikasi Billing Monitoring

Gambar 3.25 merupakan model dari TrxProjectBilling. Pada model ini terdapat sebuah *function* getAllMilestonesByProject. *Function* ini berfungsi untuk menghubungkan antara table billing_monitoring, project_information dan project_setup. Menghubungkan semua table dengan cara mengambil id dari masing-masing table.

```

<script>
    function hanyaAngka(event) {
        var angka = (event.which) ? event.which : event.keyCode
        if (angka != 46 && angka > 31 && (angka < 48 || angka > 57))
            return false;
        return true;
    }
</script>

```

Gambar 3. 26 Script hanyaAngka Aplikasi Billing Monitoring

Gambar 3.26 merupakan script. Fungsi dari script tersebut adalah suatu *field* yang menggunakan “hanyaAngka” tidak dapat menginput huruf tetapi hanya bisa memasukkan sebuah angka dan titik saja.

```
let setProjectSetupData = () => {
  if (!validateRequiredFields()) {
    alert('Mohon isi semua form.');
```

```
    $('#wizard2-tab').prop('disabled', true)
    return;
  }

  project_setup = {
    service_type: $('#service_type').val(),
    project_name: $('#project_name').val(),
    pic: $('#pic').val(),
    project_periode: $('#project_periode').val(),
    total_project_value: $('#total_project_value').val(),
    currency: $('#currency').val(),
  };
  $('#wizard2-tab').prop('disabled', false)
  document.getElementById('wizard2-tab').click()
};
```

Gambar 3. 27 Script SerProjectSetupData Aplikasi Billing Monitoring

Gambar 3.27 merupakan script. Fungsi dari script tersebut adalah untuk mewajibkan *user* mengisi pada *project_setup* atau *step 1*. Jika *user* belum mengisi satupun maka system akan melarang *user* untuk melanjutkan *step* berikutnya.

Gambar 3.28 merupakan sebuah *migrations database*. Pada gambar diatas merupakan sebuah *migrations database* dengan nama table *billings*. Migrations ini bertujuan agar mempermudah Pratkan dalam membuat sebuah database.

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('billings', function (Blueprint $table) {
            $table->id();
            $table->date('plan_date_pcos_submission')->nullable();
            $table->date('plan_date_pcos_approval')->nullable();
            $table->date('plan_date_pcohd_submission')->nullable();
            $table->date('plan_date_pcohd_approval')->nullable();
            $table->date('plan_date_pcohd_upload_system')->nullable();
            $table->date('plan_date_pi_submission')->nullable();
            $table->date('plan_date_pi_approval')->nullable();
            $table->date('plan_date_pics_upload')->nullable();
            $table->date('plan_date_pics_approval')->nullable();
            $table->date('plan_date_rsd_submission')->nullable();
            $table->date('plan_date_rsd_approval')->nullable();
            $table->date('plan_date_sci_sap_invoice_creation')->nullable();
            $table->date('plan_date_sci_sap_invoice_submitted')->nullable();
            $table->date('plan_date_sci_upload_invoice')->nullable();
            $table->date('plan_date_sci_approved_invoice')->nullable();
            $table->date('plan_date_sci_payment_approval')->nullable();
            $table->date('plan_date_received_payment')->nullable();
            $table->string('plan_dp_deduction')->nullable();
            $table->string('plan_less_retention')->nullable();
            $table->string('plan_wht')->nullable();
            $table->string('plan_original_payment')->nullable();
            $table->string('plan_gross_payment')->nullable();
            $table->string('plan_add_vat')->nullable();
            $table->string('plan_net_payment')->nullable();
            $table->date('actual_date_pcos_submission')->nullable();
            $table->date('actual_date_pcos_approval')->nullable();
            $table->date('actual_date_pcohd_submission')->nullable();
        });
    }
}

```

Gambar 3. 28 Migrations Billings Aplikasi Billing Monitoring

Pada table billings terdapat “nullable”. Fungsi dari nullable adalah jika suatu table tidak diisi maka program maupun aplikasi tetap dapat berjalan. Database yang digunakan pratikan pada pembuatan aplikasi *billing monitoring* adalah *pseudocode*.

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('project_informations', function (Blueprint $table) {
            $table->id();
            $table->string('billing_scheme')->nullable();
            $table->string('contract_type')->nullable();
            $table->string('description');
            $table->string('billing_value')->nullable();
            $table->string('billing_percentage');
            $table->string('billing_cummulative_percentage')->nullable();
            $table->string('completion_date')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('project_informations');
    }
};

```

Gambar 3. 29 Migrations *project_informations* Aplikasi Billing Monitoring

Gambar 3.29 merupakan sebuah *migrasion database*. Pada gambar tersebut table yang dibuat dengan nama *project_informations*. Pada table ini terdapat 9 *field*. Table tersebut dibuat dengan beberapa *field* yaitu *id*, *billing_scheme*, *contract_type*, *description*, *billing_value*, *billing_percentage*, *billing_cummulative_percentage* dan *completion_date*. Table *project_informations* menggunakan “nullable” seperti table sebelumnya, dikarenakan user dapat tidak mengisi suatu form pada new project. Fungsi “*public function down*” berfungsi agar jika table *database* ini ingin dihapus maka dapat terhapus melalui *source code*.


```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('project_setups', function (Blueprint $table) {
            $table->id();
            $table->string('service_type');
            $table->string('project_name');
            $table->string('pic')->nullable();
            $table->date('start_date');
            $table->date('end_date');
            $table->string('total_project_value');
            $table->string('currency');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('project_setups');
    }
};

```

Gambar 3. 30 Migrations *project_setups* Aplikasi Billing Monitoring

Gambar 3.30 merupakan sebuah *migrasion database*. Pada gambar tersebut table yang dibuat dengan nama *project_setups*. Table tersebut dibuat dengan beberapa *field* yaitu *id*, *service_type*, *project_name*, *pic*, *start_date*, *end_date*, *total_project_value* dan *currency*.

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('trx_project_billings', function (Blueprint $table) {
            $table->id();
            $table->foreignId('project_setups_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->foreignId('project_informations_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->foreignId('billings_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->integer('created_by');
            $table->integer('updated_by');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('trx_project_billings');
    }
};

```

Gambar 3. 31 Migrations *trx_project_billings* Aplikasi Billing Monitoring

Gambar 3.31 merupakan sebuah *migrasion database*. Pada gambar tersebut table yang dibuat dengan nama *trx_project_billings*. Table tersebut dibuat dengan beberapa *field* yaitu *id*, *project_setups_id*, *project_informations_id*, *billings_id*, *created_by* dan *update_by*. Table ini berfungsi untuk mengambil id pada table lain.

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('billing_forecasts', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->date('fc_date_pcos_submission')->nullable();
            $table->date('fc_date_pcos_approval')->nullable();
            $table->date('fc_date_pcohd_submission')->nullable();
            $table->date('fc_date_pcohd_approval')->nullable();
            $table->date('fc_date_pcohd_upload_system')->nullable();
            $table->date('fc_date_pi_submission')->nullable();
            $table->date('fc_date_pi_approval')->nullable();
            $table->date('fc_date_pics_upload')->nullable();
            $table->date('fc_date_pics_approval')->nullable();
            $table->date('fc_date_rsd_submission')->nullable();
            $table->date('fc_date_rsd_approval')->nullable();
            $table->date('fc_date_sci_sap_invoice_creation')->nullable();
            $table->date('fc_date_sci_sap_invoice_submitted')->nullable();
            $table->date('fc_date_sci_upload_invoice')->nullable();
            $table->date('fc_date_sci_approved_invoice')->nullable();
            $table->date('fc_date_sci_payment_approval')->nullable();
            $table->date('fc_date_received_payment')->nullable();
            $table->string('fc_dp_deduction')->nullable();
            $table->string('fc_less_retention')->nullable();
            $table->string('fc_wht')->nullable();
        });
    }
};

```

Gambar 3. 32 Migrations Billings_forecasts Aplikasi Billing Monitoring

Gambar 3.32 merupakan *migrations database*. Pada *migrations* ini penamaan dari table adalah *billing_forecasts*. Table ini menggunakan “*nullable*” yang berfungsi agar ketika sesuatu table kosong atau ketika tidak diisi oleh *user* pada form input, maka *database* dapat berjalan dengan lancar.

Gambar 3.33 merupakan *migrations database*. Pada *migrations* ini table diberi nama *session*. Table ini terdiri beberapa field yaitu *id*, *user_id*, *ip_adres*, *user_agent*, *payload* dan *last_activity*.

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('sessions');
    }
};

```

Gambar 3. 33 Migrations Session Aplikasi Billing Monitoring

Gambar 3.34 merupakan sebuah migrasion database. Pada gambar tersebut table yang dibuat dengan nama `trx_project_billings_forecasts`. Table tersebut dibuat dengan beberapa field yaitu `id`, `project_setups_id`, `project_informations_id`, `billings_id`, `billings_forecasts_id`, `created_by` dan `update_by`. Table ini berfungsi untuk mengambil id pada table lain.

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('trx_project_billing_forecasts', function (Blueprint $table) {
            $table->id();
            $table->foreignId('project_setups_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->foreignId('project_informations_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->foreignId('billings_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->foreignId('billing_forecasts_id')->constrained()->onDelete('cascade')->onUpdate('cascade');
            $table->integer('created_by');
            $table->integer('updated_by');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('trx_project_billing_forecasts');
    }
};

```

Gambar 3. 34 Migrations *trx_project_billings* Aplikasi Billing Monitoring

Gambar 3.35 merupakan sebuah migration database. Pada database ini table diberi nama *billing_forecasts*. Table ini berfungsi untuk menjadi wadah dari sebuah fitur input forecast. Table ini memiliki sebuah “nullable” yang berfungsi sebagai ketika kita tidak mengisi table tersebut maka tidak terjadi sebuah error system.

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('billings_forecasts', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->date('fc_date_pcos_submission')->nullable();
            $table->date('fc_date_pcos_approval')->nullable();
            $table->date('fc_date_pcohd_submission')->nullable();
            $table->date('fc_date_pcohd_approval')->nullable();
            $table->date('fc_date_pcohd_upload_system')->nullable();
            $table->date('fc_date_pi_submission')->nullable();
            $table->date('fc_date_pi_approval')->nullable();
            $table->date('fc_date_pics_upload')->nullable();
            $table->date('fc_date_pics_approval')->nullable();
            $table->date('fc_date_rsd_submission')->nullable();
            $table->date('fc_date_rsd_approval')->nullable();
            $table->date('fc_date_sci_sap_invoice_creation')->nullable();
            $table->date('fc_date_sci_sap_invoice_submitted')->nullable();
            $table->date('fc_date_sci_upload_invoice')->nullable();
            $table->date('fc_date_sci_approved_invoice')->nullable();
            $table->date('fc_date_sci_payment_approval')->nullable();
            $table->date('fc_date_received_payment')->nullable();
            $table->string('fc_dp_deduction')->nullable();
            $table->string('fc_less_retention')->nullable();
            $table->string('fc_wht')->nullable();
        });
    }
};

```

Gambar 3. 35 Migrations Billings_forecasts Aplikasi Billing Monitoring

```

Route::get('/', [AuthController::class, 'index'])->name('login');
Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
Route::get('/mdp', [MasterDataProjectController::class, 'index'])->name('master.data.project')->middleware('auth');
Route::prefix('auth')->group(function () {
    Route::post('/', [AuthController::class, 'redirect'])->name('auth');
    Route::get('/callback', [AuthController::class, 'callback'])->name('auth.callback');
});
Route::get('/dashboard', [DashboardController::class, 'index'])->name('dashboard.index')->middleware('auth');
Route::get('/dashboard_1', [DashboardController::class, 'table'])->name('dashboard.table_dashboard')->middleware('auth');
Route::get('/project-list', [ProjectListController::class, 'projectList'])->name('project.list')->middleware('auth');
Route::get('/project-list/{id}', [ProjectListController::class, 'milestoneProject'])->name('project.milestone')->middleware('auth');
Route::resource('/project-setup', ProjectSetupController::class)->middleware(['auth']);
Route::get('/edit', [ProjectListController::class, 'EditProject'])->name('edit.edit_project')->middleware('auth');
Route::get('/edit-milestone', [ProjectListController::class, 'EditMilestone'])->name('edit.edit_milestone')->middleware('auth');
Route::delete('/project-list/{id}', [ProjectListController::class, 'destroy']);
// Route::delete('/project-list/{id}', [ProjectListController::class, 'destroy_milestone']);

```

Gambar 3. 36 Tampilan Routes - Web

Gambar 3.36 merupakan tampilan pada routes/web. Pada halaman ini berfungsi untuk menjalankan beberapa halaman maupun sebuah fungsi pada halaman pemrograman. *Routes* yang digunakan pada aplikasi ini ada sebanyak 14 *routes*. Fungsi “POST” adalah untuk mengirimkan data yang sudah di diisi ke database. Fungsi dari “GET” adalah untuk mengakses suatu halaman atau mengirimkan suatu permintaan pencarian. Fungsi “DELETE” digunakan untuk menghapus sebuah database.

```

let editBillingList = (id) => {
    id_data_for_update = id
    let billing_plan_data = data['billing_plan'][id]
    let project_information_data = data['project_information'][id]
    btnUpdateToggle(true)

    $('#billing_scheme').val(project_information_data['billing_scheme'])
    $('#contract_type').val(project_information_data['contract_type'])
    $('#description').val(project_information_data['description'])
    $('#billing_value').val(project_information_data['billing_value'])
    resetDrawPercentage()
    $('#billing_value').data('billing_percentage', project_information_data['billing_percentage'])
    $('#billing_percentage').attr('placeholder', project_information_data['billing_percentage'] + '%')
    $('#completion_date').val(project_information_data['completion_date'])

    $('#plan_date_pcos_submission').val(billing_plan_data['plan_date_pcos_submission'])
    $('#plan_date_pcos_approval').val(billing_plan_data['plan_date_pcos_approval'])
    $('#plan_date_pcohd_submission').val(billing_plan_data['plan_date_pcohd_submission'])
    $('#plan_date_pcohd_approval').val(billing_plan_data['plan_date_pcohd_approval'])
    $('#plan_date_pcohd_upload_system').val(billing_plan_data['plan_date_pcohd_upload_system'])
    $('#plan_date_pi_submission').val(billing_plan_data['plan_date_pi_submission'])
    $('#plan_date_pi_approval').val(billing_plan_data['plan_date_pi_approval'])
    $('#plan_date_pics_upload').val(billing_plan_data['plan_date_pics_upload'])
    $('#plan_date_pics_approval').val(billing_plan_data['plan_date_pics_approval'])
    $('#plan_date_rsd_submission').val(billing_plan_data['plan_date_rsd_submission'])
    $('#plan_date_rsd_approval').val(billing_plan_data['plan_date_rsd_approval'])
    $('#plan_date_sci_sap_invoice_creation').val(billing_plan_data['plan_date_sci_sap_invoice_creation'])
}

```

Gambar 3. 37 editBillingList

Gambar 3.37 merupakan memberikan sebuah perintah untuk mengedit pada *billing plan card list*. Fungsi ini untuk mempermudah user

jika terdapat kesalahan maupun ingin mengecek kembali apa saja yang sudah diinputkan.

3.3 Kendala yang Dihadapi

Selama masa Kerja Profesi di PT.TRIPATRA, Pratikan menghadapi beberapa tantangan saat mengembangkan aplikasi billing monitoring. Kendala-kendala tersebut meliputi:

1. *Framework* yang digunakan adalah *Laravel*, sehingga pratikan memerlukan sedikit waktu untuk mempelajari mengenai *framework* tersebut.
2. Terdapat sebuah fitur form yang dimana dapat diisi berkali-kali sebelum dilakukan submit. Pratikan mempunyai kendala dalam hal tersebut.
3. Kesibukan pratikan dalam hal tugas akhir sehingga menghambat pratikan dalam membuat projek yang sedang dilakukan

3.4 Cara Mengatasi Kendala

Untuk menanggulangi hambatan yang ditemui selama proses Kerja Profesi dan pembuatan aplikasi website billing monitoring, Pratikan telah menemukan beberapa pendekatan efektif:

1. Pratikan mencari solusi dengan cara mempelajari *framework Laravel* di internet
2. Mengatasi fitur mengisi berkali-kali dengan cara pratikan mencari di internet dan bertanya kepada mentor mengenai fitur tersebut.
3. Pratikan mengerjakan tugas akhir dan kerja profesi secara pararel atau diwaktu yang bersamaan untuk mencegah terjadinya keterlambatan.

3.5 Pembelajaran yang Diperoleh dari Kerja Profesi

Dalam melaksanakan Kerja Profesi, Pratikan berhasil mengakumulasi beberapa pengalaman yang sangat bermanfaat sebagai persiapan untuk memasuki lingkungan kerja profesional. Pengalaman-pengalaman ini mencakup :

1. Kemampuan Pratikan dalam beradaptasi ditingkatkan melalui pembelajaran komunikasi yang efektif dengan mentor dan rekan kerja.
2. Pratikan dapat mengetahui dengan jelas seperti apa dunia kerja itu.
3. Pratikan mendapatkan ilmu-ilmu yang tidak dipelajari di kampus.

