SKRIPSI

ANALISIS PEMANFAATAN LARAVEL DEBUGBAR DALAM MEMPERMUDAH PENGERJAAN ISSUE PADA APLIKASI PUSAT PENGEMBANGAN SUMBER DAYA MANUSIA (PPSDM)

Disusun Oleh:

N a m a	: Dendy Surya Darmawan
NIM	: 17523078

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS PEMANFAATAN LARAVEL DEBUGBAR DALAM MEMPERMUDAH PENGERJAAN ISSUE PADA APLIKASI PUSAT PENGEMBANGAN SUMBER DAYA MANUSIA (PPSDM)

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a	: Dendy Surya Darmawan
NIM	: 17523078

Yogyakarta, 13 Juli 2021

Pembimbing,

(Andhik Budi Cahyono, S.T., M.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS PEMANFAATAN LARAVEL DEBUGBAR DALAM MEMPERMUDAH PENGERJAAN ISSUE PADA APLIKASI PUSAT PENGEMBANGAN SUMBER DAYA MANUSIA PPSDM (PPSDM)

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 5 Agustus 2021

Tim Penguji

Ketua Penguji

Andhik Budi Cahyono, S.T., M.T.

Anggota 1

Moh. Idris, S.Kom., M.Kom

Anggota 2

Sri Mulyati, S.Kom., M.Kom

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Dendy Surya Darmawan
NIM	:	17523078

Tugas akhir dengan judul:

ANALISIS PEMANFAATAN LARAVEL DEBUGBAR DALAM PENGERJAAN ISSUE PADA APLIKASI PUSAT PENGEMBANGAN SUMBER DAYA MANUSIA (PPSDM)

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 13 Juli 2021

(Dendy Surya Darmawan)

HALAMAN PERSEMBAHAN

Alhamdulillah, segala puji dan syukur kepada Allah SWT karena berkat rahmat, karunia dan hidayah-Nya, sehingga penulis dapat menyelesaikan laporan tugas akhir magang selama 6 bulan di perusahaan PT. Javan Cipta Solusi.

Laporan tugas akhir ini penulis persembahkan kepada orang - orang yang sudah membantu dalam mendukung, menyemangati, memberikan arahan dan masukkan selama proses penulisan sebagai berikut:

1. Kedua orang tua yang saya cintai dan banggakan, laporan ini dipersembahkan kepada sosok yang sangat penting bagi penulis dalam melewati berbagai fase kehidupan. Jasa dan pengorbanan yang mereka telah diberikan tak terhitung lagi di mulai dari membangun karakter dan prinsip diri, ajaran agama, membiayai segala fasilitas demi kesuksesan termasuk biaya perkuliahan saat ini di Universitas Islam Indonesia. Tidak ada yang bisa penulis sampaikan selain rasa bersyukur kepada sang Maha Pencipta semesta atas kehadiran mereka berdua dan berharap masih diberikan banyak waktu untuk terus bersama penulis kelak beberapa tahun kedepan.
2. Bapak Andhik Budi Cahyono, S.T., M.T sebagai dosen pembimbing tugas akhir yang telah memberikan banyak arahan dan bimbingan yang sangat berarti bagi penulis dalam menyelesaikan tugas akhir ini.
3. Rekan dari Ini Grup sebagai tempat untuk mengeluarkan berbagai keluh kesah serta meminta tanggapan selama proses penyusunan laporan tugas akhir.

HALAMAN MOTO

“Bila telah diperjuangkan dengan sungguh-sungguh baik hasilnya sukses atau gagal, sesungguhnya semangat perjuangan itu adalah kesuksesan tersendiri”

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Alhamdullilahi Robbil ‘Alamin rasa syukur selalu dipanjatkan kepada allah SWT karena atas rahmat, taufiq dan hidayah-Nya, penulis dapat menyelesaikan laporan akhir magang yang berjudul “Analisis Pemanfaatan Laravel Debugbar Dalam Pengerjaan Issue Pada Aplikasi Pusat Pengembangan Sumber Daya Manusia (PPSDM)”.

Adapun laporan akhir magang ini juga sekaligus sebagai tugas akhir sebagai syarat untuk menyelesaikan pendidikan dengan jenjang Strata 1 (S1) pada jurusan Informatika Universitas Islam Indonesia. Penulis mengucapkan terima kasih dan hormat yang sebesar-besarnya kepada pihak yang sudah membantu selama penyusunan laporan tugas akhir ini, selain itu juga penulis berterima kasih kepada pihak lain yang sudah memberikan ilmu, pengalaman dan nasihat yang luar biasa dalam selama masa perkuliahan di jurusan Informatika. Adapun pihak-pihak tersebut sebagai berikut:

1. Allah SWT, atas segala limpahan rahmat dan hidayah-Nya, memberikan penulis kekuatan, kesabaran, serta harapan agar tetap terus berusaha melewati berbagai rintangan yang datang hingga penulis akhirnya dapat menyelesaikan laporan tugas akhir dengan lancar.
2. Kedua orang tua dari penulis yang selalu mendukung, menyemangati dan mendoakan yang terbaik agar dipermudahkan dalam berbagai urusan.
3. Bapak Hendrik, S.T., M.Eng., selaku Ketua Jurusan Informatika Universitas Islam Indonesia
4. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika Program Sarjana Universitas Islam Indonesia
5. Bapak Andhik Budi Cahyono S.T., M.T selaku dosen pembimbing tugas akhir yang telah memberikan banyak arahan dan bimbingan yang sangat berarti bagi penulis dalam menyelesaikan laporan tugas akhir ini.
6. Bapak Anandia Muhammad Yudhistira selaku *supervisor* atau pembimbing lapangan selama magang di PT. Javan Cipta Solusi
7. Rekan-rekan *Senior* dan *Junior Enginner* pada Tim Engineer PT. Javan Cipta Solusi yang sudah memberikan ilmu-ilmu dan pengalaman baru selama magang.
8. Rekan-rekan tim PPSDM yang sudah memberikan arahan dan bimbingan selama pengerjaan tugas perbaikan issue aplikasi PPSDM.

9. Mas Novi Aditya, sebagai *Junior programmer* yang sering membantu saya ketika mengalami kesulitan terkait baris kode maupun hal teknis terkait *coding* lainnya selama magang.
10. Mas Qisthi Ramadhani, salah seorang *Senior Programmer* yang juga sering membantu menjelaskan mekanisme pengerjaan tugas, memberikan masukkan dan saran dari kode yang dihasilkan serta menambah ilmu dan wawasan baru dari sudut pandang tertentu selama bergabung dalam proyek PPSDM.
11. Teman-teman Ini Grup, sebagai tempat curhat atau mengeluarkan keluh kesah serta meminta masukkan atau tanggapan terkait penyusunan laporan tugas akhir.
12. Semua pihak lain yang tidak bisa disebutkan, yang sudah membantu saya melewati masa perkuliahan hingga sampai ke proses penyusunan laporan tugas akhir ini.

Semoga semua pihak yang telah membantu penulis selama penyusunan laporan tugas akhir maupun selama proses perkuliahan mendapatkan pahala yang berlimpah serta diberikan kemudahan berbagai urusan selalu dari Allah SWT.

Penulis meminta maaf apabila selama penyusunan laporan tugas akhir ini terdapat banyak kesalahan teknis seperti tata bahasa, tata tulis, ejaan dan pungtuasi maupun isi berupa materi yang kurang luas dan lengkap.

Penulis berharap semoga laporan tugas akhir ini dapat bermanfaat bagi para pembaca serta sebagai sumber referensi untuk penulisan karya ilmiah dalam penelitian lain.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 13 Juli 2021

(Dendy Surya Darmawan)

SARI

Dalam pengerjaan *issue* (*bug* dan *error*) pada website aplikasi Pusat Pengembangan Sumber Daya Manusia (PPSDM) bidang Pengadaan Barang dan Jasa (PBJ), *programmer* memerlukan informasi yang mudah dan cepat untuk seperti pencarian nama fail *views*, nama *controller*,nama *route*, dan nama *method controller*. Akan tidak efisien apabila *programmer* melakukan pencarian dengan cara manual. *Packages Laravel Debugbar* merupakan solusi yang digunakan oleh programmer proyek PPSDM karena menyediakan ringkasan informasi sebuah halaman mulai dari fail *views, route, model, query* dan jenis *collector* lain. *Packages* tersebut menampilkan sebuah panel *overlay* di bagian bawah halaman website. *Packages* tersebut dalam penelitian ini digunakan untuk menangani pengerjaan issue proyek PPSDM yaitu mempermudah pencarian nama fail *views*, pencarian nama *route* dan *controller*, serta melakukan *print out* sebuah *variable*. Pemanfaatan tersebut kemudian dievaluasi untuk mengetahui apakah *packages* tersebut benar-benar dapat mempermudah pengerjaan *issue* proyek PPSDM. Parameter evaluasi diambil dari subkarakteristik function *suitability* yaitu *function completeness, function correctness, function appropriateness* dan subkarakteristik dari *performance efficiency* yaitu *time behavior* dan *resource utilization* dari model ISO/IEC 25010. Berdasarkan hasil evaluasi yang telah dilakukan, diketahui bahwa *packages* Laravel Debugbar dapat mempermudah dalam pencarian nama fail *views*, pencarian nama *route* dan *controller*, serta melakukan *print out* sebuah *variable*.

Kata kunci: *Laravel Debugbar,* Analisis, ISO/IEC 25010.

GLOSARIUM

Laravel Debugbar	sebuah *packages* yang mengintegrasikan antara PHP Debugbar dengan Laravel.
Laravel	sebuah *framework* aplikasi web gratis dengan bahasa pemrograman PHP yang menggunakan konsep Model-View-Controller (MVC).
Packages	sebuah *source code* atau kode program yang berdiri sendiri dengan fungsi tertentu.
ISO/IEC 25010	standar Internasional untuk mengevaluasi kualitas perangkat lunak, versi lanjutan dari ISO/IEC 9126.
Functional Completeness	kemampuan sebuah sistem atau perangkat lunak untuk memberikan fungsi yang dapat mencakup semua yang ditentukan dan tujuan dari pengguna.
Functional Correctness	kemampuan sebuah sistem atau perangkat lunak untuk menyediakan hasil yang benar dengan tingkat presisi yang diperlukan.
Functional Appropriateness	kemampuan sebuah sistem atau perangkat lunak dengan fungsi yang ada mampu memfasilitasi pencapaian tugas dan tujuan tertentu.
Time Behavior	tingkatan waktu respon dan waktu proses serta tingkat keluaran dari sebuah sistem atau perangkat lunak ketika menjalankan fungsinya memenuhi persyaratan.
Resource Utilization	tingkatan jumlah dan jenis sumber daya yang digunakan oleh sistem atau perangkat lunak ketika menjalankan fungsinya memenuhi persyaratan.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latarbelakang

Dalam pengembangan aplikasi berbagai masalah bisa muncul di luar kendali dari tim proyek, pengguna atau klien diharapkan dapat memberikan *feedback* semaksimal mungkin untuk meningkatkan kualitas dari pengembangan aplikasi. Masalah-masalah tersebut biasa disebut dengan istilah *issue*. Adapun jenis *issue* yang dilaporkan dari pengguna atau klien adalah *bug* dan *error* ketika mengakses halaman tertentu.

Pada pengerjaan sebuah *issue* proyek pengembangan sebuah aplikasi, *programmer* tentu harus dapat memahami dengan cepat maksud serta ekspektasi *issue* yang telah dikerjakan, memperkirakan alur pengerjaan serta penulisan kode, mengetahui *boundary* atau batasan kode yang akan digunakan, serta dapat mengimplementasikan kode tersebut dengan kualitas kode yang baik dan mengikuti standar proyek. Kemampuan atau *skills* yang dimiliki *programmer* seiring waktu akan bertambah dengan intensitas pengerjaan berbagai jenis *issue* yang diberikan. Semakin banyak issue yang dikerjakan maka tentu semakin tinggi tantangan yang akan datang serta menimbulkan persepsi dari *programmer* bahwa ada kemungkinan kegagalan dalam mengerjakan *issue* tersebut. Beberapa penelitian mengungkapkan bahwa tantangan dalam mengerjakan sesuatu hal secara positif dapat meningkatkan motivasi dalam diri. Wilkie's (2016) dalam penelitiannya, menemukan bahwa siswa yang menerima tantangan lebih dalam *issue* matematika maka secara positif akan memberikan dampak dua sisi yaitu (kesenangan, keterikatan, ketertarikan) dan manfaat pembelajaran (Wilkie, 2016). Selain itu dampak lain yang ditemukan adalah secara efektif meningkatkan motivasi siswa, performa pembelajaran, dan rasa kepuasan dalam pembelajaran (Hung et al., 2015). Dengan kata lain *programmer* memang akan menghadapi tantangan yang sulit, namun dampak yang didapatkan selama mengerjakan sebuah *issue* hingga menyelesaikannya tentunya akan bermanfaat untuk peningkatan kualitas dan kemampuan dalam bekerja.

Meningkatkan kemampuan dengan menambah intensitas dalam mengerjakan *issue* memang menjadi kewajiban bagi *programmer*, karena dari penugasan tersebut akan mendapatkan lebih banyak pengalaman serta menyerap lebih banyak ilmu. Namun kemampuan yang dimiliki oleh setiap *programmer* masih belum cukup. *Programmer* perlu menggunakan *tools* yang relevan dan tepat agar membantu dalam bekerja. Ada berbagai macam *tools* yang membantu dalam pengembangan sebuah perangkat lunak atau aplikasi dengan fungsi-fungsi tertentu seperti

communication team, database environment, code editor, task collaboration tools, version control systems dan masih banyak *tools* lain. Salah satu *tools* yang digunakan dalam proyek yang diangkat pada laporan tugas akhir ini yaitu aplikasi PPSDM (Pusat Pengembangan Sumber Daya manusia) bidang PBJ (Pengadaan Barang/Jasa) adalah *packages Laravel Debugbar*.

Laravel Debugbar merupakan sebuah *packages Laravel* yang membantu *programmer* dengan mengumpulkan data-data menggunakan kelas PHP yaitu DataCollector yang kemudian menjadi sebuah informasi yang relevan, valid, akurat dan lengkap yang dapat digunakan dalam proses *debugging* atau menulis kode. Cara kerja hingga keputusan yang dibuat oleh *programmer* salah satunya berpengaruh karena informasi yang baik dan berkualitas. Beberapa pemanfaatan *packages* ini dalam proses pengerjaan *issue* pada proyek PPSDM yaitu memberikan kemudahan dalam pencarian nama fail *views*, *route* dan *controller* yang digunakan serta juga dapat melakukan *print out* sebuah *variable* sebagai pengganti *function* umum di Laravel yaitu *dump and die* atau *dd()*. *Packages* ini berbentuk sebuah panel dibagian bawah layar aplikasi, dengan masing-masing tab sesuai dengan *collectors* yang telah disediakan packages ini seperti *views*, *messages*, *exceptions*, *query*, *route*, *models*, *cache* dan *collector* lainnya.

Pada laporan tugas ini dibahas tentang analisis yaitu sebuah evaluasi atau pengujian yang dilakukan untuk membuktikan hipotesis pemanfaatan *packages Laravel Debugbar* dapat mempermudah dalam pengerjaan *issue* proyek PPSDM. Parameter yang digunakan untuk mengevaluasi diambil dari dimensi *product quality model* ISO/IEC 25010. Parameter yang digunakan ada 5 yang dibagi menjadi 3 subkarakteristik dari *function suitability* yaitu *functional completeness*, *functional correctness* dan *functional appropriateness*. Sedangkan 2 subkarakteristik selanjutnya dari *performance efficiency* yaitu *time behavior* dan *resource utilization*. Model ISO/IEC 25010 merupakan versi lanjutan dari ISO/IEC 9126 yang telah direvisi secara teknis menjadi standar internasional terbaru dalam mengevaluasi kualitas perangkat lunak.

1.2 Ruang Lingkup Magang

Program magang dilaksanakan selama 6 bulan di mulai sejak 1 September 2020 hingga 28 Februari di PT. Javan Cipta Solusi dengan posisi sebagai *programmer* PHP. Perusahaan ini bergerak dibidang Teknologi Informasi penyedia jasa yang membantu dalam melayani Businnes Optimization, Web/Mobile App Development, Data Analytics, Product Development, dan Managed Services (*PT Javan Cipta Solusi: Home*, 2019). Adapun aktivitas yang dilakukan selama magang di proyek ini adalah sebagai berikut.

a. Pengerjaan tugas dengan jenis *issue Bug*

b. Pengerjaan tugas dengan jenis *issue Error*
c. Pengerjaan tugas *Sonarqube*
d. Pengerjaan tugas Otorisasi menu dan hak akses media gambar pada *role user* tertentu
e. Pengerjaan tugas *migration* dan *seeder database*
f. Pengerjaan tugas penambahan fitur *download* excel
g. Pengerjaan tugas penambahan fitur baru *user competencies*

1.3 Tujuan

Tujuan dari penulisan laporan tugas akhir tentang analisis pemanfaatan *packages Laravel Debugbar* dalam mempermudah pengerjaan *issue* pada proyek PPSDM sebagai berikut.

a. Untuk mengetahui apakah subkarakteristik *function suitability* dan *performance efficiency* pada model ISO/IEC 25010 dapat menjadi parameter evaluasi dalam menilai pemanfaatan *packages Laravel Debugbar* dalam pengerjaan *issue* proyek PPSDM.
b. Untuk mengetahui hasil evaluasi berdasarkan subkarakteristik *function suitability* dan *performance efficiency* pada model ISO/IEC 25010 bahwa pemanfaatan packages Laravel Debugbar memberikan kemudahan dalam pengerjaan *issue* dalam proyek PPSDM.

1.4 Manfaat

Manfaat dari penulisan tugas akhir pemanfaatan *packages Laravel Debugbar* pada tahapan *debugging* program dalam pengerjaan issue perbaikan bug sebagai berikut.

a. Memberikan gambaran evaluasi pemanfaatan *packages Laravel Debugbar* dalam penyelesaian *issue* proyek PPSDM berdasarkan subkarakteristik *function suitability* dan *performance efficiency* pada ISO/IEC 25010.
b. Memberikan gambaran hasil analisis dari evaluasi pemanfaatan *packages Laravel Debugbar* dalam mempermudah penyelesaian *issue* proyek PPSDM berdasarkan subkaratersitik *function suitability* dan *peformance efficiency* pada ISO/IEC 25010.
c. Memberikan gambaran terkait tahapan-tahapan dalam penyelesaian sebuah *issue* oleh *programmer* dalam proyek PPSDM dengan menggunakan *packages Laravel Debugbar*.

BAB II
DASAR TEORI

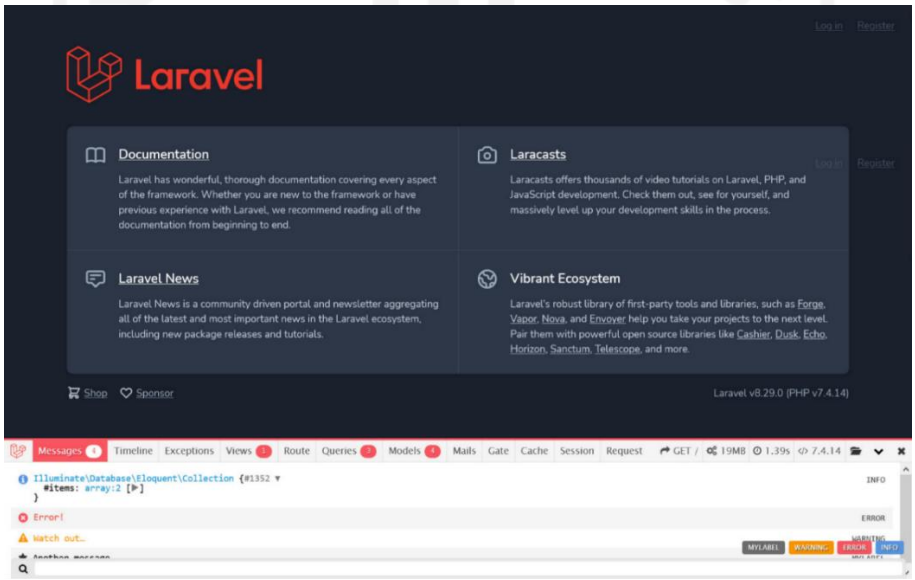
2.1 Laravel Debugbar dan Penggunaannya

Laravel Debugbar merupakan sebuah *packages* dari PHP Debugbar yang terintegrasi dengan *framework* Laravel (Heuvel, 2021). Packages ini dibuat oleh seorang *Web Developer* sekaligus *Co-Founder* dan *Lead Developer* FruitCake asal Belanda yakni Barry van de Heuvel. *Packages Laravel Debugbar* menggunakan salah satu kelas PHP yaitu DataCollectors, kelas ini berfungsi untuk mengumpulkan data-data dari kode yang membangun halaman yang diakses menjadi sebuah informasi baru. Proses instalasi dari *packages Laravel Debugbar* ini menggunakan composer dengan notasi perintah pada Gambar 2.1.

```
1      composer require barryvdh/laravel-debugbar --dev
2
```

Gambar 2.1 Notasi instalasi *packages Laravel Debugbar*

Setelah instalasi selesai, *packages* ini akan berjalan setiap mengakses halaman pada aplikasi. *Packages Laravel Debugbar* akan menampilkan informasi yang sudah dikumpulkan berbentuk panel *overlay* dibagian bawah *website* dengan masing-masing *tab* memiliki jenis *collectors* tertentu seperti pada Gambar 2.2.



Gambar 2.2 Tampak Packages Laravel Debugbar

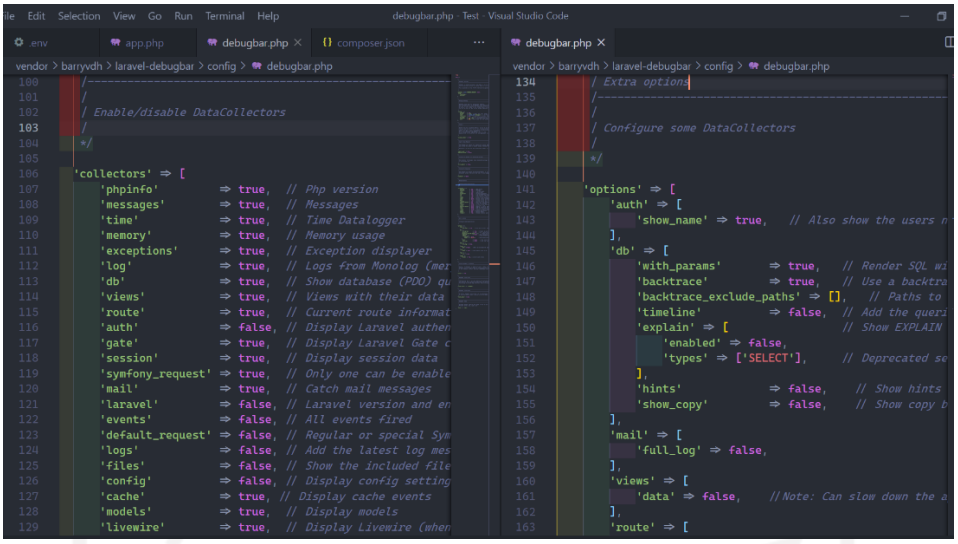
Adapun *default collectors* yang ada pada *packages Laravel Debugbar* ini adalah sebagai berikut.

- PhpInfoCollector, menampilkan versi PHP yang sedang digunakan aplikasi.
- MessagesCollector, menampilkan pesan *method* seperti *error*, info, *warning*, dan juga dapat melakukan kostumisasi pesan, pada umumnya MessageCollector digunakan untuk proses *debugging* yaitu melakukan *print statement* sebuah *variable* menggantikan fungsi var_dump() atau dd().
- TimeDataCollector, menampilkan waktu yang dibutuhkan untuk *load* semua data di halaman(*booting* dan *application).*
- *MemoryCollector,* menampilkan penggunaan *memory* yang dihabiskan untuk menjalankan aplikasi.
- *ExceptionsCollector,* menampilkan *error exception*(perubahan alur program dari kondisi normal ke kondisi tertentu). Ketika aplikasi sedang berjalan, error ini tidak akan menyebabkan *crash* atau aplikasi berhenti melainkan aplikasi tersebut akan berjalan tidak sesuai dengan ekspektasi, *exception* biasanya terjadi ketika adanya transaksi data ke *database*.

Packages *Laravel Debugbar* juga memiliki *collector* lain seperti *views, messages, exceptions, query, route, models, cache* dan masih banyak *collector* lainnya seperti yang ditunjukan pada Gambar 2.2. Untuk konfigurasi *packages Laravel Debugbar* terdapat dalam fail ‘debugbar.php’. Pada fail tersebut berisi konfigurasi yang dapat disesuaikan dengan kebutuhan seperti menambahkan atau mematikan *collectors* tertentu, pemilihan tema seperti *light* dan *dark*, konfigurasi tambahan untuk beberapa *collectors* tertentu dan lain sebagainya dapat dilihat pada Gambar 2.3.

Penggunaan *Packages* ini bermanfaat bagi *programmer* karena memberikan informasi yang relevan dengan halaman yang diakses, mudah dipahami dan cara pengoperasian mudah serta dapat mengefisiensi kerja. Maksud efisiensi di sini yaitu dapat menghemat waktu karena tidak perlu mencari *resource* kode yang membangun halaman tersebut, dengan begitu tenaga yang dikeluarkan juga sedikit.

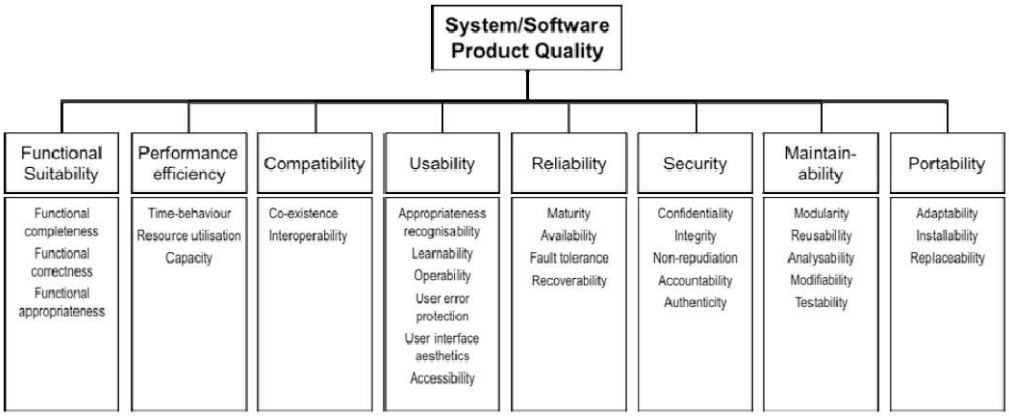
Untuk para *developer* perlu menjadi catatan bahwa *packages Laravel Debugbar* hanya digunakan pada lingkup *development saja,* karena akan memperlambat kinerja dari aplikasi dengan melakukan *request* semua data *collectors* setiap melakukan *refresh* halaman. Apabila penggunaan *packages* ini melambat pada lingkup *development* disarankan untuk mematikan beberapa *collectors.*



Gambar 2.3 Konfigurasi *Packages Laravel Debugbar*

2.2 Model ISO/IEC 25010

ISO/IEC merupakan standar dunia internasional dalam mengevaluasi atau menilai kualitas sebuah perangkat lunak yang dikembangkan oleh *International Organization for Standarization and International Electronical Commission* disingkat dengan ISO/IEC. Adapun versi yang digunakan pada tugas akhir ini adalah versi ISO/IEC 25010 edisi pertama tahun 2011, pengembangan dari versi ISO/IEC 9126 tahun 2001 yang telah direvisi secara teknis, menjadi standar internasional terbaru dan relevan untuk menguji *software* dan sistem komputer yang dikembangkan. ISO/IEC 25010 memiliki 8 karakteristik yang menjadi tolak ukur dalam mengevaluasi kualitas perangkat lunak berdasarkan dimensi *product quality* yaitu *functional suitability*, *performance efficiency*, *compatibility*, *usability*, *reliability*, *security*, *maintainability*, dan portability seperti pada Gambar 2.4 (ISO/IEC, 2011).



Gambar 2.4 Model *Product Quality* berdasarkan *ISO/IEC 25010:2011*

Dalam laporan tugas akhir ini, parameter yang digunakan untuk mengevaluasi pemanfaatan *packages Laravel Debugbar* ada 5 yaitu *functional completeness, functional correctness, functional appropriateness, time behavior* dan *resource utilization.* Subkarakteristik tersebut merupakan turunan dari karakteristik *function suitability* dan *performance efficiency* dari model ISO/IEC 25010 dapat dilihat pada Gambar 2.4. Adapun data yang digunakan untuk mengevaluasi pemanfaatan *packages* ini dengan 5 parameter tersebut adalah berdasarkan asumsi peneliti selama pengerjaan *issue* proyek PPSDM menggunakan *packages* ini. Setelah melakukan evaluasi kemudian ditarik kesimpulan untuk membuktikan kebenaran dari dugaan peneliti yaitu *packages Laravel Debugbar* dapat mempermudah pengerjaan *issue* dalam proyek PPSDM.

Karakteristik yang digunakan pada laporan tugas akhir ini hanya 2 yaitu *function suitability* dan *performance efficiency* dari model ISO/IEC 25010 dengan total 5 subkarakteristik sebagai parameter evaluasi pemanfaatan *packages Laravel Debugbar.* Hal ini dengan alasan karena 2 karakteristik tersebut memiliki dampak yang dirasakan oleh penulis yaitu dari segi fungsionalitas(*function suitability*) yang diberikan *packages* ini membuat peforma dan efisiensi (*performance efficiency*) kerja *programmer* meningkat selama pengerjaan *issue* proyek PPSDM.

2.2.1 Parameter Functional Completeness

Functional Completeness yaitu kemampuan sebuah sistem atau perangkat lunak untuk memberikan fungsi yang dapat mencakup semua yang ditentukan dan tujuan dari pengguna(ISO/IEC, 2011). Maksud dari parameter ini adalah untuk menilai apakah *packages* ini yang memiliki fungsi yang diinginkan oleh pengguna khususnya yaitu menampilkan ringkasan informasi berupa nama fail *views, route* dan *controller* yang digunakan serta dapat melakukan *print out* sebuah *variable* untuk pengecekan kode yang ditulis.

Adapun tahapan-tahapan dalam melakukan pengujian parameter ini adalah dengan melihat semua jenis *collector* yang disediakan oleh *packages* ini berupa sebuah *tab,* kemudian mengakses *tab* yang diinginkan dalam hal ini adalah *tab views, route* dan *messages.* Untuk *tab messages* sebagai *output* dari hasil *print out variable* dilakukan penilaian sebagai pembuktian untuk Barry vd. Heuvel, selaku pembuat *packages Laravel Debugbar* bahwa benar *packages* ini dapat melakukan pengecekan hasil *variable* dari kode yang ditulis dengan melempar *variable* data ke notasi pada Gambar 3.13. Apabila 3 *tab* tersebut menampilkan informasi sesuai kebutuhan yang diinginkan oleh *programmer,* selanjutnya adalah melakukan penilaian dengan bobot dan skor dari asumsi peneliti selama penggunaan *packages* ini pada pengerjaan *issue* proyek PPSDM sebagai berikut.

a. Sangat Baik (nilai 3), *packages* ini dapat menampilkan informasi yang diinginkan oleh *programmer* yaitu untuk melakukan pencarian nama fail *views, route* dan *controller* serta dapat juga melakukan pengecekan hasil kode yang sedang ditulis dengan *print out variable.*
b. Baik (nilai 2), *packages* ini memiliki fungsi menampilkan ringkasan informasi halaman terbentuk, namun bukan menjadi perhatian utama bahwa *packages* ini dapat menggantikan fungsi yang sudah ada pada *Laravel* seperti pengecekan hasil kode yang ditulis dengan melempar *variable* data menggunakan notasi pada Gambar 3.13 dan mencetaknya pada *tab messages* yang sebenarnya dapat menggunakan perintah *dd().*
c. Kurang Baik (nilai 1), *packages* ini tidak dapat menampilkan ringkasan informasi halaman berupa nama fail *views, route* dan *controller* yang digunakan serta tidak dapat melempar *variable* data ke notasi pada Gambar 3.13 untuk melakukan pengecekan hasil kode yang sedang ditulis pada *tab messages*.

2.2.2 Parameter Functional Correctness

Functional Correctness yaitu kemampuan sebuah sistem atau perangkat lunak untuk menyediakan hasil yang benar dengan tingkat presisi yang diperlukan(ISO/IEC, 2011). Maksudnya dari parameter ini adalah untuk menilai apakah *packages* ini menampilkan ringkasan informasi terbentuknya halaman dengan benar khususnya untuk pencarian nama fail *views, route* dan *controller* yang digunakan serta hasil dari pengecekan kode yang ditulis dengan melempar *variable* data menggunakan notasi pada Gambar 3.13 sama jika menggunakan *function* umum *dd()*.

Adapun tahapan-tahapan dalam melakukan pengujian parameter ini adalah dengan mengakses nama fail *views, route* dan *controller* yang digunakan halaman, sedangkan hasil pengecekan *variable* data yang ada pada *tab messages* harus sama apabila menggunakan *function dd().* Apabila informasi yang ditampilkan pada *tab* terbukti valid maka dilakukan penilaian dengan bobot skor dan pernyataan berdasarkan asumsi peneliti selama menggunakan *packages* ini selama pengerjaan *issue* proyek PPSDM sebagai berikut.

a. Sangat Baik (nilai 3), *packages* ini menampilkan ringkasan informasi halaman terbentuk yang terbukti valid apabila mengakses sesuai nama fail *views, route* atau *controller* yang ada pad *tab packages* dengan fail kode program yang ada pada aplikasi.

b. Cukup (nilai 2), *packages* ini menampilkan ringkasan informasi halaman terbentuk yang terbukti valid apabila mengakses fail pada *tab* dengan fail kode program aplikasi, namun bukan menjadi perhatian utama untuk hasil pengecekan kode yang ditulis karena apabila *packages* ini dapat melakukan fungsi yang sama dengan function *dd()* maka sudah pasti *variable* data tersebut terbukti valid.
c. Kurang (nilai 1), *packages* ini menampilkan data yang tidak sesuai atau tidak valid khususnya pada *tab views, route* dan hasil pengecekan *variable* data pada *tab messages* .

2.2.3 Parameter Functional Appropriateness

Functional Appropriateness yaitu kemampuan sebuah sistem atau perangkat lunak dengan fungsi yang ada mampu memfasilitasi pencapaian tugas dan tujuan tertentu(ISO/IEC, 2011). Maksudnya dari parameter ini adalah untuk menilai apakah *packages* ini memfasilitasi pencapaian tugas dan tujuan tertentu dalam hal ini berkaitan dengan pencarian nama fail *views, route* dan *controller*, serta dalam melakukan *print out variable.*

Adapun tahapan-tahapan dalam melakukan pengujian parameter ini adalah dengan melakukan pengecekan pada *tab views, route* dan *messages* untuk melihat adanya informasi tambahan yang dalam mempermudah pencarian nama fail *views, route* dan *controller.* Sedangkan dalam melakukan *print out variable* perlu mencoba beberapa *syntax* tertentu untuk memastikan ada pemanfaatan tertentu yang dapat mempermudah *programmer*. Apabila ada informasi spesifik yang mempermudah dalam pencarian nama fail *views, route* dan *controller,* sedangkan pada saat melakukan *print out variable* terbukti ada pemanfaatan tertentu, maka dilakukan penilaian dengan bobot skor dan pernyataan berdasarkan asumsi peneliti selama menggunakan *packages* ini selama pengerjaan *issue* proyek PPSDM sebagai berikut.

a. Sangat Baik (nilai 3), *packages* ini menampilkan informasi spesifik dan berpengaruh secara signifikan dengan *programmer* khususnya dalam pencarian nama fail *views, route* dan *controller* yang digunakan. Untuk *tab messages* memfasilitasi *programmer* seperti *shortcut syntax* untuk menulis *print out variable* agar lebih mudah digunakan daripada *syntax* kode Gambar 3.13 yang terlalu panjang.
b. Cukup (nilai 2), *packages* ini mampu menampilkan informasi spesifik dan lengkap namun tidak secara signifikan berpengaruh langsung dengan kebutuhan yang diinginkan oleh *programmer* terkait pemanfaatan dalam pencarian nama fail *views, route* dan *controller* pada *tab views* dan *route.* Sedangkan melakukan *print out*

variable pada *tab messages* memang tidak ada fasilitas yang dapat mempermudah *programmer* dalam menulis namun *packages* dapat memberikan fasilitas seperti *print out* beberapa *variable* sekaligus.

c. Kurang (nilai 1), *packages* ini menampilkan informasi ringkasan sebuah halaman terbentuk dalam hal ini yaitu pencarian nama fail *views, route* dan *controller* serta melakukan *print out variable,* namun justru tidak memfasilitasi untuk memberikan informasi yang lebih spesifik atau lengkap untuk mempermudah *programmer* khususnya pada *tab views* dan *route*. Sedangkan melakukan *print out variable* pada *tab messages* tidak ada fasilitas yang dapat mempermudah *programmer* dalam menulis *syntax.*

2.2.4 Parameter Time Behavior

Time Behavior yaitu tingkatan waktu respon dan waktu proses serta tingkat keluaran dari sebuah sistem atau perangkat lunak ketika menjalankan fungsinya memenuhi persyaratan (ISO/IEC, 2011). Maksudnya parameter ini adalah untuk menilai apakah *packages* ini membutuhkan waktu respon dan waktu proses tertentu untuk menjalankan fungsinya dalam hal ini yaitu pencarian nama fail *views, route* dan *controller* yang digunakan halaman serta waktu yang dibutuhkan ketika melakukan *print out variable* menggunakan *packages* ini.

Adapun tahapan-tahapan dalam melakukan pengujian parameter ini adalah dengan mencatat waktu proses hingga menampikan sebuah halaman secara penuh(Septianto & Ade Sekarwati, 2019). Apabila *packages* ini terbukti tidak membebankan sebuah halaman dengan tidak ada tambahan waktu proses yang signifikan, maka dilakukan penilaian dengan bobot skor dan pernyataan berdasarkan asumsi peneliti selama menggunakan *packages* ini selama pengerjaan *issue* proyek PPSDM sebagai berikut.

a. Sangat Baik (nilai 3), *packages* ini memiliki waktu proses yang hampir sama jika dibandingkan tanpa menggunakan *packages* ini. Hal ini terjadi karena sebanyak apapun *collectors* beserta setiap *resource* yang di-*load*, *packages* ini seperti tidak ada *gap* atau rentang waktu proses untuk menjalankan fungsinya bahkan jauh lebih cepat dibandingkan melakukan *load* penuh sebuah halaman.

b. Cukup (nilai 2), *packages* ini memiliki tambahan waktu proses yang bervariasi ketika melakukan *load* secara penuh sebuah halaman. Hal ini dapat terjadi karena setiap halaman memiliki jumlah *resource* tertentu untuk melakukan *load* halaman beserta semua *collector* dari *packages*. Selain itu faktor dari kualitas jaringan, ukuran elemen

yang di-*load* seperti (*html, javascript, css, images* dan lain sebagainya) juga harus dalam kondisi yang memadai karena dapat mempengaruhi waktu proses sebuah halaman.

c. Kurang (nilai 1), *packages* ini memiliki tambahan waktu proses yang banyak karena *collectors* beserta *resource* yang di-*load* sebuah halaman bervariasi sehingga memerlukan waktu tambahan untuk menampilkan halaman beserta *packages* secara penuh.

2.2.5 Parameter Resource Utilization

Resource Utilization yaitu tingkatan jumlah dan jenis sumber daya yang digunakan oleh sistem atau perangkat lunak ketika menjalankan fungsinya memenuhi persyaratan (ISO/IEC, 2011). Maksudnya adalah parameter ini digunakan untuk menilai apakah *packages* ini memakai banyak memori untuk melakukan *load* halaman secara penuh dalam menjalankan fungsinya untuk menampilkan ringkasan informasi setiap *collectors.*

Adapun tahapan-tahapan dalam melakukan pengujian parameter ini adalah dengan mencatat pemakaian memori yang dihabiskan untuk melakukan *load* penuh sebuah halaman beserta *packages* ini(Septianto & Ade Sekarwati, 2019). Apabila ternyata *packages* ini menghabiskan banyak memori untuk menjalankan fungsinya dibandingkan jika tidak menggunakan *packages* ini, maka dilakukan penilaian dengan bobot skor dan pernyataan berdasarkan asumsi peneliti selama menggunakan *packages* ini selama pengerjaan *issue* proyek PPSDM sebagai berikut.

a. Sangat Baik (nilai 3), *packages* ini ternyata menghabiskan sedikit memori atau pemakaian memori yang tidak berpengaruh signifikan, karena *packages* ini hanya menampilkan ringkasan informasi sebuah halaman yang tentu memiliki ukuran serta penggunaan *list html, css* dan *javascript* sederhana yang jauh lebih hemat memori.
b. Cukup (nilai 2), *packages* ini menghabiskan memori yang relatif besar karena menyesuaikan *resource* yang di-*load* masing-masing *collectors* pada setiap halaman yang diakses.
c. Kurang (nilai 1), *packages* ini menghabiskan banyak memori dan membebani kinerja dari perangkat lunak karena semakin banyak *resource* yang digunakan pada *packages Laravel Debugbar* dalam hal ini adalah *collectors* maka sangat memungkinkan penambahan pemakaian memori yang besar.

2.3 Kajian Pustaka

Penelitian ini menjelaskan tentang penggunaan karakteristik *functional suitability* dan *performance efficiency* dari model ISO/IEC 25010 sebagai parameter pengujian pada aplikasi bernama Tesadaptif.net. Pengujian *functional suitability* menggunakan metode *blackbox* yang menghasilkan 18 instrumen pengujian yang diukur dengan skala *Guttman,* hasil yang diperoleh dari 88 responden dihitung dengan formulasi matriks *feature completeness.*Sedangkan untuk pengujian *performance efficiency* akan dievaluasi menggunakan *tool* GTmetrix yang dihasilkan nilai yaitu waktu respon *Page Load, Page Speed* dan *YSlow*. Hasil dari penelitian ini yaitu pengukuran *functional suitability* memberikan nilai 0,999 atau 99% berhasil, disimpulkan bahwa hampir semua fitur aplikasi berjalan lancar. Sementara pengukuran *performance efficiency* yang dilakukan pada browser *Chrome* dan *Firefox* berturut-turut rata-rata *page speed* adalah 86% dan 86%, sementara itu nilai rata-rata dari YSlow adalah 89,42% dan 89,4%. Nilai tersebut menyatakan aplikasi Tesadaptif.net termasuk kategori *Grade* B. Sedangkan untuk waktu respon mengakses halaman aplikasi menggunakan *Chrome* (1,542) dan *Firefox* (1,057). Hasil pengukuran tersebut menjadi referensi perbaikan aplikasi Tesadaptif.net (DR Dako & Ridwan, 2021).

Penelitian kedua menjelaskan tentang pengujian kualitas pengembangan sebuah Sistem Informasi Akademik Mahasiswa (SIAKAD) menggunakan 8 aspek dari model ISO/IEC 25010, salah satu aspek yang diperhatikan disini adalah pengujian *performance efficiency.* Pengujian ini dilakukan dengan mencatat waktu proses halaman melalui *add-on developer tools network* pada *Chrome* untuk menguji parameter *time behavior* yang dari aplikasi SIAKAD, sedangkan dari pengujian parameter *resource utilization* dilakukan dengan mencatat pemakaian memori (*memory usage*) pada saat program aplikasi dijalankan. Hasil dari pengujian dari *performance efficiency* adalah sistem SIAKAD sudah berjalan sesuai dengan baik dengan waktu proses halaman dibawah dari 10 detik dan pemakaian rata-rata memori sebesar 2,02 MB(Septianto & Ade Sekarwati, 2019).

Dari dua penelitian tersebut menjadi dasar referensi yang dipilih sebagai acuan untuk melakukan evaluasi pemanfaatan *packages Laravel Debugbar,* berdasarkan parameter dari subkarakteristik *functional suitability* dan *performance efficiency* dari model ISO/IEC 25010. Evaluasi berdasarkan parameter subkarakteristik *functional suitability* dilakukan dengan metode *blackbox testing* atau pengujian skenario yang sudah dilakukan selama penggunaan *packages* ini dalam mengerjakan *issue* proyek PPSDM apakah bersesuaian dengan fungsi-fungsi yang seharusnya dilakukan oleh *packages* ini. Sedangkan pengujian *performance efficiency* menggunakan ekstensi aplikasi *Firefox* yaitu *page load time* dan pemakaian memori dapat dilihat menggunakan *task manager* dari aplikasi *Firefox.* Penilaian pengujian ini menggunakan skala

likert 3 kategori yaitu Sangat Baik (SB), Cukup (C), dan Kurang (K). Penilaian yang dilakukan pada laporan ini berdasarkan asumsi penulis pada saat menggunakan *packages* ini selama pengerjaan *issue* proyek PPSDM.

BAB III
PELAKSANAAN MAGANG

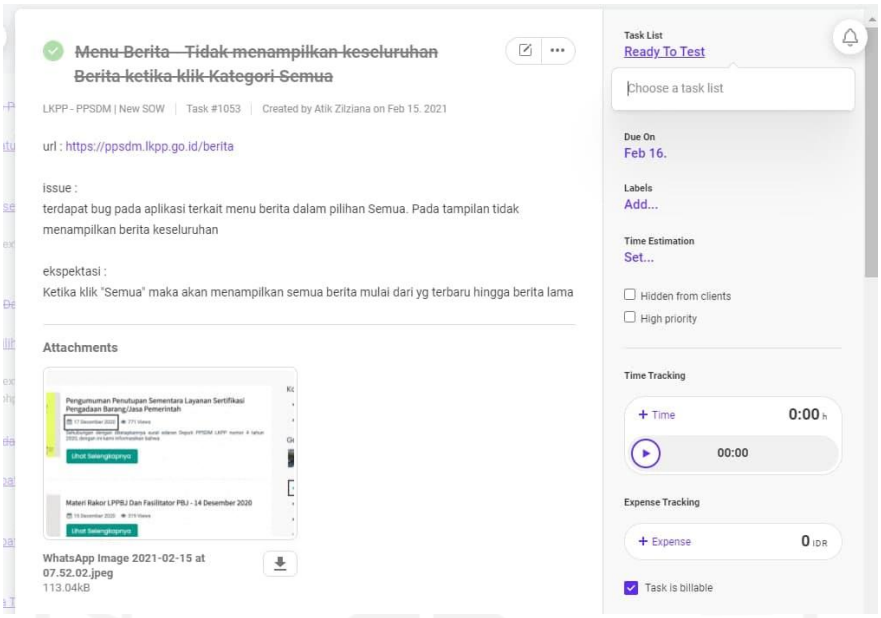
Program magang di PT.Javan Cipta Solusi dengan kontrak selama 6 bulan pada tanggal 1 September 2020 hingga 28 Februari 2021 dengan posisi *programmer* PHP. Secara umum aktivitas yang dikerjakan penulis selama magang adalah menyelesaikan *issue* pada *production server* di proyek Pusat Pengembangan Sumber Daya Manusia (PPSDM). Production server merupakan sebuah server yang digunakan untuk menjalankan aplikasi web yang diakses oleh *end- users*, biasa disebut dengan istilah *live server* (TechTarget, 2017). *Issue* dari pengguna atau klien ini kemudian dikumpulkan dan dibuat skala prioritas oleh *System Analyst* berdasarkan tingkat urgensi tertentu, selanjutnya akan dibuat penugasan kepada setiap *programmer* untuk menyelesaikan tugas-tugas tersebut.

3.1 Tahapan Penyelesaian Sebuah Tugas Dalam Proyek PPSDM

Proses dalam penyelesaian sebuah tugas termasuk tugas dengan jenis *issue* dalam proyek PPSDM oleh seorang *programmer* terbagi menjadi beberapa tahapan sebagai berikut.

A. Memahami ekspektasi dari tugas yang dikerjakan

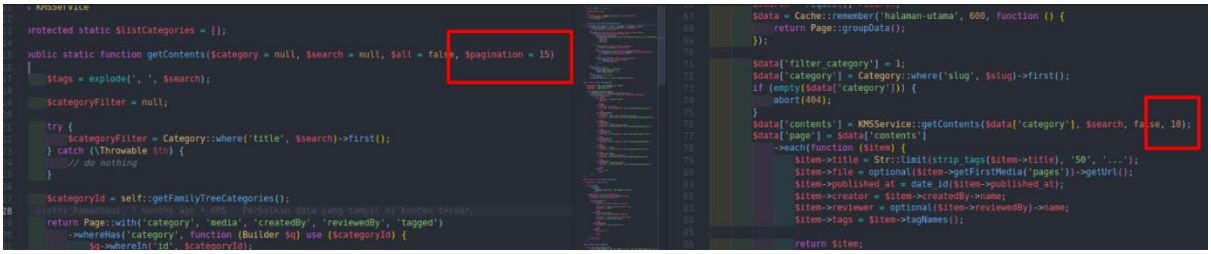
   Mengetahui maksud dan tujuan dengan tugas yang sedang dikerjakan dengan jelas dan informasi yang lengkap tanpa adanya ambiguitas yang dapat mempengaruhi proses pengerjaan. Jika ada hal yang kurang jelas seperti nama *user*, *password, url* yang belum terlampir di deskripsi tugas atau informasi lain langsung ditanyakan atau konfirmasi kembali kepada pembuat tugas agar dapat memulai proses pengerjaan dan hasil sesuai ekspektasi yang diharapkan. Label tugas perlu digantik menjadi "*in progress"* sebagai penanda sedang mengerjakan tugas tersebut. Pada deskripsi dari setiap tugas khususnya jenis *issue*(*bug* dan *error*) akan dilampirkan *url* atau halaman yang terdapat masalah, kemudian dengan menggunakan *packages Laravel Debugbar* dapat mempermudah mencari nama fail *views*, *route* dan *controller* yang diguanakan halaman tersebut. Berikut pada Gambar 3.1 merupakan contoh deskripsi tugas jenis *issue bug* pada proyek PPSDM.



Gambar 3.1 Deskripsi tugas jenis *issue bug*

B. Menulis kode baru atau melakukan *refactoring* kode lama

Programmer mulai menulis kode baru atau melakukan *refactoring* kode lama sesuai dengan kebutuhan yang ditentukan dari tugas. *Refactoring* merupakan tahapan mengubah desain kode tanpa menghilangkan fungsionalitas dari kode. Dalam penulisan kode baru atau *refactoring, programmer* harus melakukan pengecekan hasil kode yang ditulis agar tidak terdapat kesalahan ketika dijalankan dan kode tetap dinamis. Cara melakukan pengecekan adalah dengan melempar *variable* data ke *function dump and die* atau *dd(). Packages Laravel Debugbar* dapat melakukan pengecekan hasil kode yang ditulis dengan melempar *variable* data menggunakan *façade class* dari *Laravel Debugbar* atau metode *non-static* dengan notasi perintah seperti pada Gambar 3.13. Adapun keuntungan *packages Laravel Debugbar* dalam pengecekan ini yaitu program aplikasi tetap berjalan sekaligus dapat melempar lebih dari satu *variable* data, tidak seperti *function dd()* yang akan memberhentikan program aplikasi yang berjalan dan hanya dapat melakukan pengecekan satu *variable*. Pada Gambar 3.2 merupakan contoh sebagian dari *refactoring* kode dimana setelah melakukan filter pada *category*, konten yang ditampilkan harus 10 data *pagination*, sehingga mengubah kode dengan menambahkan parameter *default* untuk *pagination* dengan 15 data.



Gambar 3.2 Contoh *refactoring* kode dengan menambahkan *default pagination*

C. Melakukan *commit*, *push* kode dan *merge request*

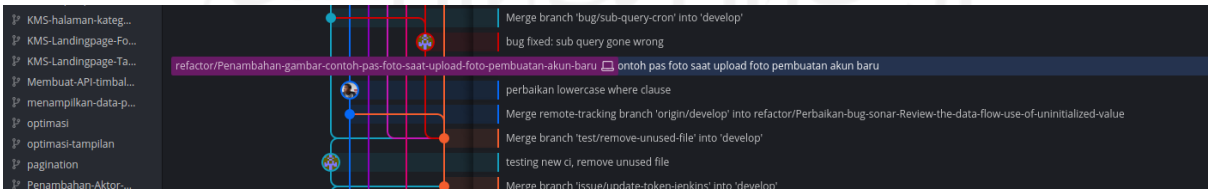
Setelah programmer menyelesaikan penulisan kode, selanjutnya sebelum melakukan *commit* terlebih dahulu melakukan standarisasi kode menggunakan *tools* PHP CS Fixer dengan menjalankan perintah di *console*. Hal ini dilakukan agar *pipeline styling_checking* tidak mengalami *error*. Adapun perintah yang dijalankan dapat dilihat pada Gambar 3.3.

```
1       vendor/bin/php-cs-fixer fix --diff --diff-format udiff
2
```

Gambar 3.3 Notasi PHP CS Fixer sebelum melakukan *commti* kode

Langkah selanjutnya adalah melakukan *push* kode tersebut ke *remote gitlab* PPSDM dengan penamaan *branch* memilik format "nama-prefix/judul-tugas" contoh dapat dilihat pada Gambar 3.4. Adapun nama *prefix* yang tersebut adalah sebagai berikut.

- feature/, untuk fitur baru atau enhancement fitur yang sudah ada sebelumnya.
- issue/, untuk bugfix.
- hotfix/, untuk bugfix yang sangat penting dan harus segera dideploy ke production.
- refactor/, untuk perbaikan kode tanpa adanya penambahan fitur baru.
- styling/, untuk perbaikan tampilan tanpa adanya penambahan fitur.



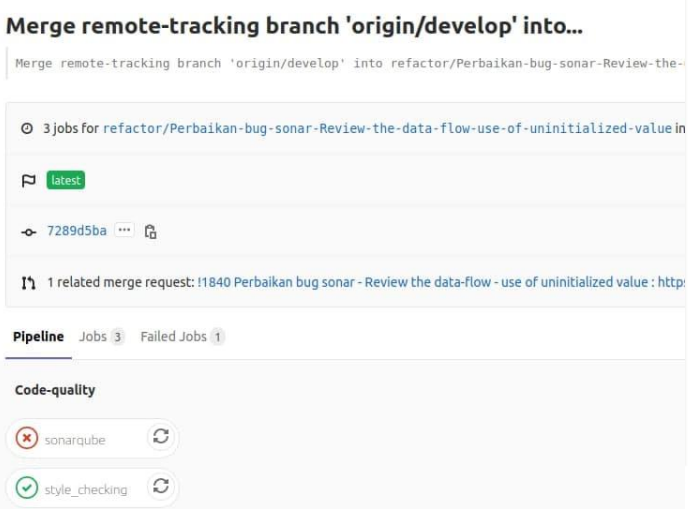
Gambar 3.4 Contoh prefix nama *branch*

Selanjutnya yaitu mengajukan *merge request* dengan ketentuan judul *merge request* menyesuaikan dengan judul tugas yang diberikan, *assign* nama maintainer/senior

programmer untuk melakukan *review* kode yang dikerjakan, lampirkan juga link tugas pada bagian deskripsi dan ubah label menyesuaikan dengan kondisi merge *request* saat ini.

D. Melakukan pengecekan *Gitlab pipeline* setelah *merge request*

Setelah melakukan *merge request* ke *branch development* proyek PPSDM, kode harus melewati tahapan *automated build and test* yaitu dengan pengecekan *styling_checking* dan *sonarqube. Styling_checking* merupakan *tools* standarisasi kode dari PHP CS Fixer, sedangkan *sonarqube* merupakan sebuah *tools* analisis kode yang bertujuan untuk melakukan pengecekan kualitas dan keamanan kode yang akan di-*deploy*, *tools* sonarqube sudah terintegrasi dengan Gitlab proyek PPSDM dan apabila kode yang ditulis terindikasi melanggar *rules* yang sudah ditetapkan maka akan terjadi *error* pada *pipeline.* Pada Gambar 3.5 merupakan contoh *error pipeline* pada *merge request.* Ulangi dari tahapan ke-2 apabila masih terdapat error *pipline* pada *merge request* yang diajukan.

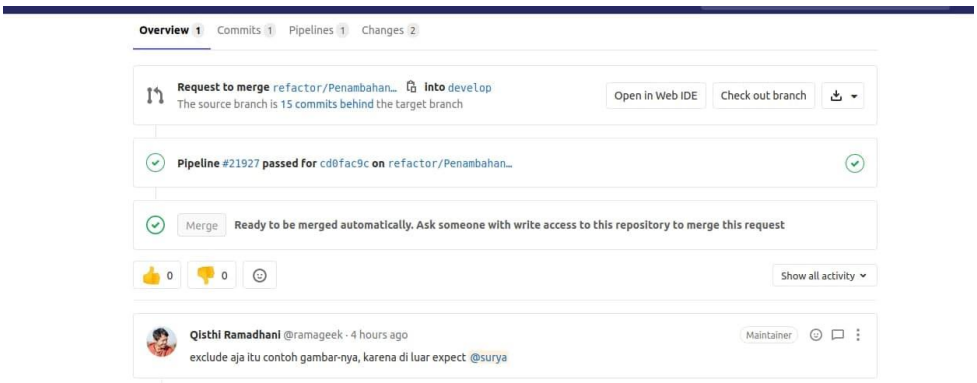


Gambar 3.5 Contoh *error pipeline* pada *merge request*

E. *Review* kode oleh *maintainer Gitlab*/*senior programmer*

Setelah lolos melewati *automated build* dan *test* menggunakan *styling_checking* dan *sonarqube*, kode yang ditulis perlu untuk dilakukan *review* dan *approve* secara langsung oleh seorang *maintainer*/*senior programmer* agar memastikan kualitas kode yang dihasilkan dapat terbaca oleh semua *programmer*, dipahami dengan mudah, tingkat *cognitive complexity* kode yang rendah, dapat mengikuti standar penulisan kode seperti format penamaan variabel, penggunaan function, penggunaan *loop*, dan sebagainya.

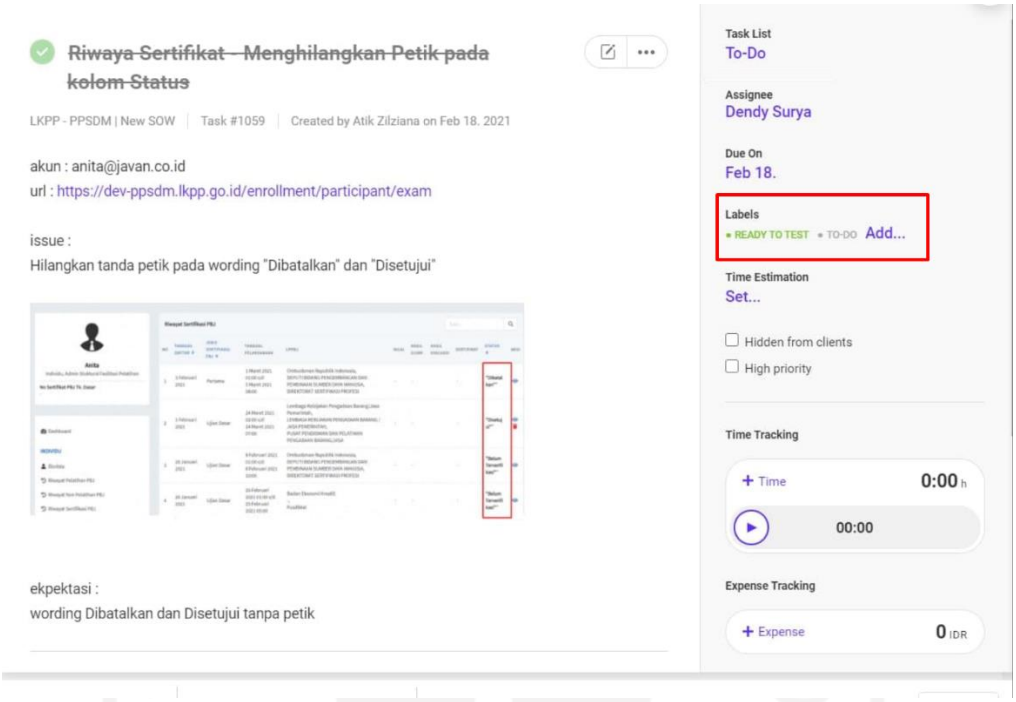
Inspeksi manual terhadap kode juga dapat mengurangi error atau defects perangkat lunak dan meningkatkan kualitas proyek perangkat lunak(Ackerman et al., 1989). Pada Gambar 3.6 merupakan contoh review dari *senior programmer* proyek PPSDM dari *merge request* yang diajukan.



Gambar 3.6 Contoh *review* kode pada salah satu *merge request*

F. Ubah label tugas menjadi *ready to test*.

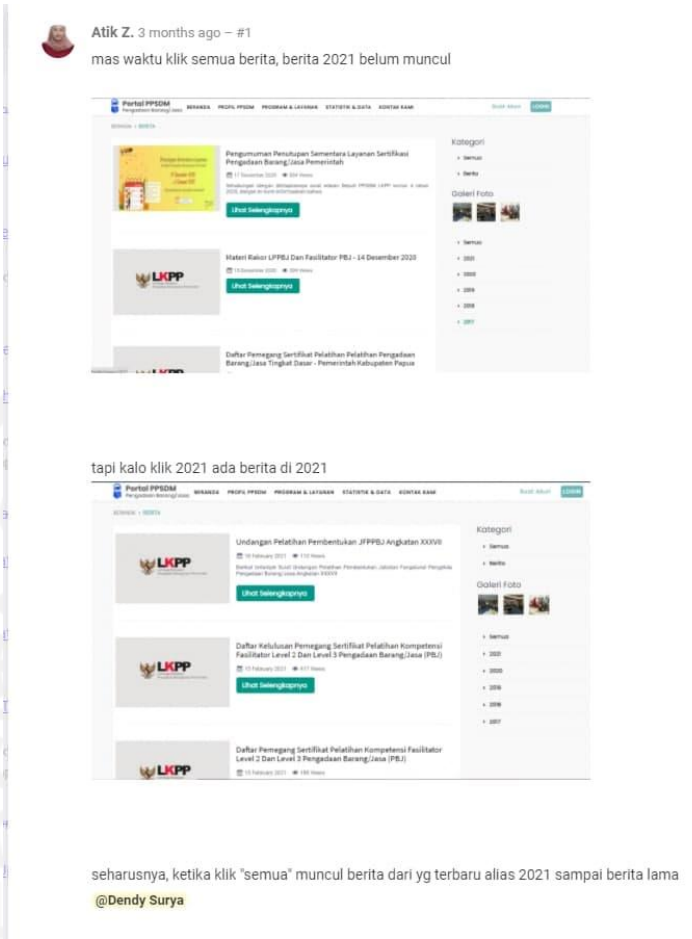
Setelah melewati proses *review* dan *merge request* diterima oleh *maintainer/senior programmer*, selanjutnya adalah mengubah label menjadi *ready to test* dan *assign* nama *tester* untuk segera melakukan pengujian. Pada Gambar 3.7 merupakan contoh label tugas untuk pengerjaan *issue bug* pada salah satu halaman.



Gambar 3.7 Contoh label pengerjaan tugas jenis *issue bug*

G. Hasil pengujian implementasi kode oleh *tester*

Apabila dari seorang *tester* menguji hasil impelemntasi kode dan ternyata masih belum sesuai dengan ekspektasi, maka harus mengulang kembali ke tahapan ke-2. Pada Gambar 3.8 merupakan contoh komentar dari *tester* terkait pengujian yang telah dilakukan dari hasil implementasi kode yang dikerjakan.

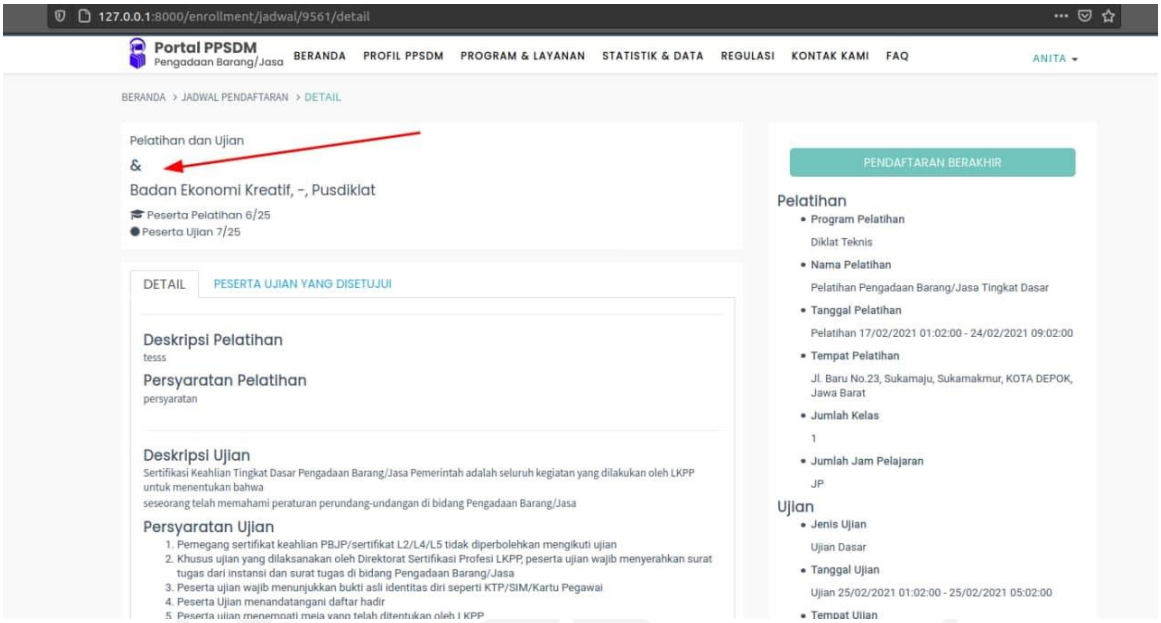


Gambar 3.8 Contoh komentar dari *tester*

3.2 Pemanfaatan Packages Laravel Debugbar Dalam Pengerjaan Issue Proyek PPSDM

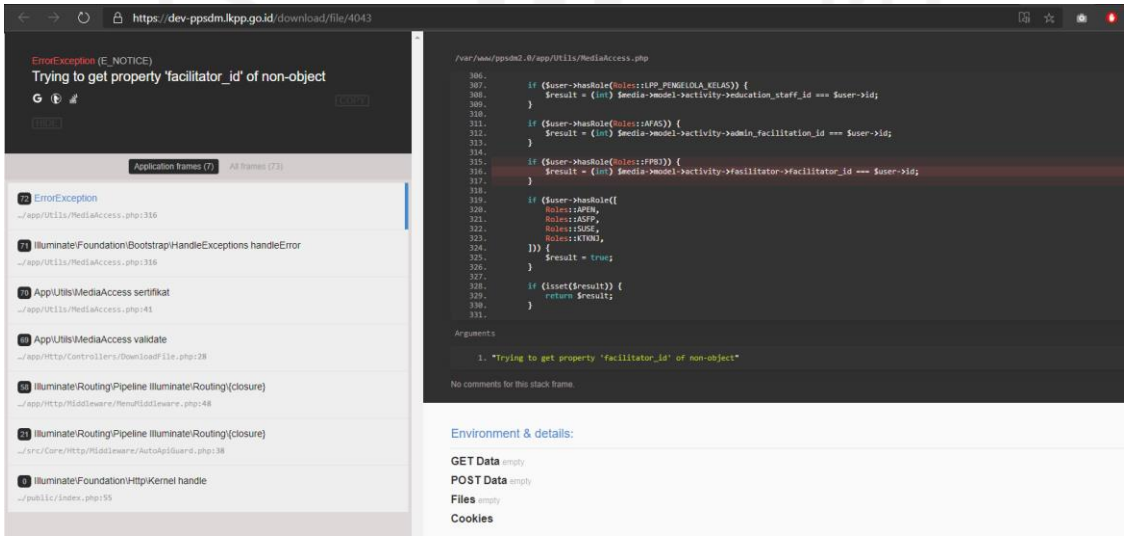
Issue merupakan salah satu jenis tugas yang dikerjakan oleh *programmer* yaitu masalah yang ditemukan oleh klien atau pengguna saat menjalankan aplikasi setelah rilis versi terbaru. Adapun jenis *issue* yang dikerjakan sebagai berikut.

a. *Bug*, sebuah kesalahan dimana perangkat lunak tidak menjalankan seperti yang seharusnya dilakukan begitupun sebaliknya. Contoh seperti *issue bug* nama pelatihan dan nama ujian yang hilang seperti pada Gambar 3.9.



Gambar 3.9 Contoh *issue bug* proyek PPSDM

b. *Error*, sebuah kesalahan pada aplikasi yang menyebabkan aplikasi tidak dapat menjalankan fungsinya. Contoh ketika *download* sebuah sertifikat dan terjadi *error* dengan pesan "*Trying to get property 'facilitator_id' of non-object* " seperti pada Gambar 3.10.

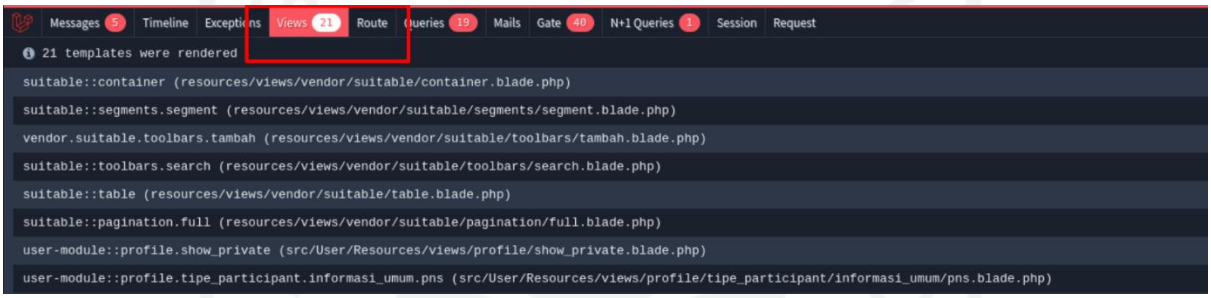


Gambar 3.10 Contoh *issue error* pada proyek PPSDM

Pengerjaan *issue-issue* diatas pada proyek PPSDM dibantu dengan *tools packages Laravel Debugbar*. Adapun secara spesifik pemanfaatan dari *packages Laravel Debugbar* sebagai berikut.

A. Pencarian nama fail *views*

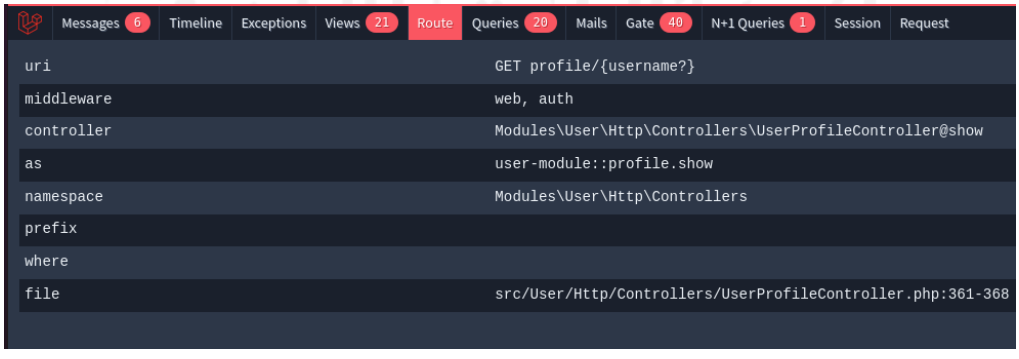
Packages Laravel Debugbar ini mempunyai *ViewCollector* yang mendapatkan informasi semua nama fail yang membangun sebuah halaman. Efisiensi kerja dari *programmer* juga meningkat karena tidak perlu melakukan pencarian secara manual berbagai *resource* fail *views*, apabila sebuah halaman mengimplementasikan *blade extend* tentu *packages* ini akan menampilkan semua nama fail tersebut. Semua informasi nama fail terdapat pada *tab views* seperti pada Gambar 3.11.



Gambar 3.11 Contoh nama fail yang membangun sebuah halaman pada *tab views*

B. Pencarian nama *route* dan *controller*

Packages Laravel Debugbar ini memberikan informasi nama *route* dan *controller* yang digunakan pada halaman yang diakses serta dilengkapi dengan informasi lengkap nama *prefix*, *uri*, baris kode *method controller* yang digunakan. Semua informasi tersebut ada pada *tab Route* seperti pada Gambar 3.12. Secara umum semua nama route dan *controller* terdapat pada fail *web.php* namun ada juga beberapa proyek yang mengkonfigurasi dan menyimpan bukan pada fail tersebut. Dengan begitu *packages* ini sangat efektif memberikan informasi terkait pencarian nama *route* dan *controller*.



Gambar 3.12 *Tab Route* untuk pencarian nama *route* dan *controller*

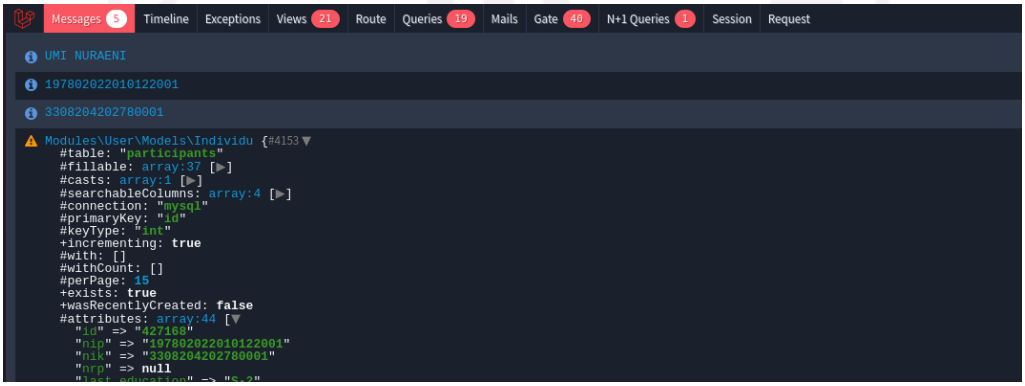
C. Melakukan *print out* sebuah *variable* dalam pengecekan *output* baris kode

Dalam menulis sebuah baris kode perlu dipastikan bahwa kode menghasilkan *output* yang benar sesuai ekspektasi yang diharapkan. Salah satu cara yang digunakan untuk memastikan baris kode yang ditulis sudah benar adalah dengan melakukan pengecekan dengan *print out variable* dari baris kode tersebut secara langsung. Pada *packages Laravel Debugbar* dapat melakukan *print* yaitu dengan memasukkan *variable* yang ingin dicek ke dalam *façade class* dari *Laravel Debugbar* atau menggunakan metode *non-static* dengan perintah seperti pada baris ke-4 dan baris ke-9 pada Gambar 3.13.

```
1     $user = auth()->user();
2
3     //façade Laravel Debugbar/static method
4     Debugbar::nama_properti_atau_method('nama_variabel')
5     //contoh
6     Debugbar::debug($user);
7
8     //non-static method
9     app('debugbar')->nama_properti_atau_method('nama_variabel');
10    //contoh
11    app('debugbar')->debug($user);
12
```

Gambar 3.13 Notasi print *variable* pada *packages Laravel Debugbar*

Adapun hasil *print variable* tersebut akan tampil pada *tab messages* seperti pada Gambar 3.14, pada gambar itu juga terdapat 5 jumlah *messages* yang menandakan banyaknya jumlah *variable* yang di-*print.*



Gambar 3.14 Contoh hasil *print out variable* pada *tab messages*

3.3 Evaluasi Pemanfaatan *Packages Laravel Debugbar* Dalam Pengerjaan *Issue*

Evaluasi yang dilakukan yaitu mengevaluasi pemanfaatan *packages Laravel Debugbar* dalam pencarian nama fail *views*, *route* dan *controller* serta penggunaan packages ini dalam melakukan *print out* sebuah *variable.* Adapun parameter peniliaian yaitu dari subkarakteristik

function suitability yaitu *functional completeness, functional correctness, functional appropriateness* dan subkarakteristik *performance efficiency* yaitu *time behavior, resource utilization* pada ISO/IEC 25010.

Penilaian yang dilakukan setelah pengujian menggunakan skala *likert*. Skala ini digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok tentang kejadian (Riduwan, 2017). Dalam laporan tugas akhir ini yaitu pengukuran kualitas dari *packages Laravel Debugbar*. Skala *likert* yang digunakan adalah 3 kategori seperti pada tabel Table 3.1. Pemilihan jumlah kategori tersebut dengan alasan karena dalam laporan ini evaluasi *packages* dinilai langsung oleh penulis sendiri, sehinga pemilihan 5 atau 7 kategori dianggap terlalu berlebihan dan cukup menyesuaikan kondisi sesuatu yang ingin diukur (Friedman et al., 1981).

Table 3.1 Penilaian dan bobot skor evaluasi parameter subkarateristik ISO/IEC 25010

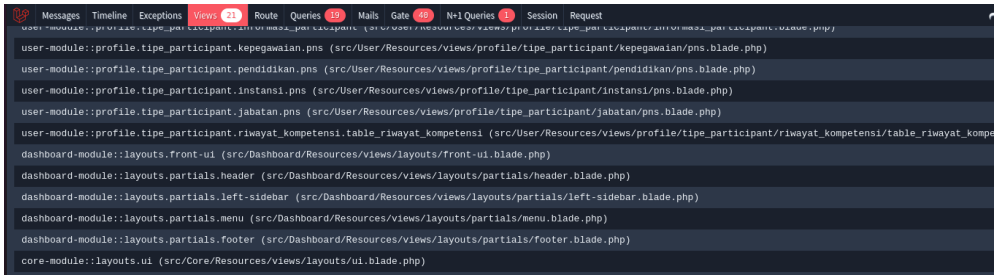
Pernyataan	***Bobot Skor***
Sangat Baik (SB)	3
Cukup (C)	2
Kurang (K)	1

3.3.1 Evaluasi Packages Laravel Debugbar

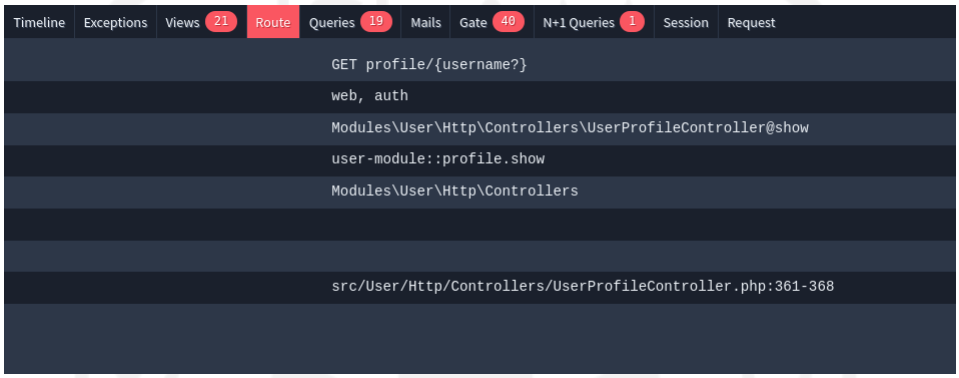
A. Functional Completeness, kemampuan sebuah sistem atau perangkat lunak untuk memberikan fungsi yang dapat mencakup semua yang ditentukan dan tujuan dari pengguna.

1) Pencarian nama fail views, route, dan controller

Evaluasi berdasarkan *functional completeness, packages Laravel Debugbar* dapat menyediakan informasi yang dibutuhkan *programmer* terkait nama fail *views, route* dan *controller* yang digunakan pada Gambar 3.15 dan Gambar 3.16. Hal ini mempermudah *programmer* karena tidak perlu mencari secara manual informasi tersebut, bahkan ada kemungkinan sebuah halaman menggunakan *blade extend* (penggunaan kode program lain pada kode program lainnya) pada Gambar 3.15.



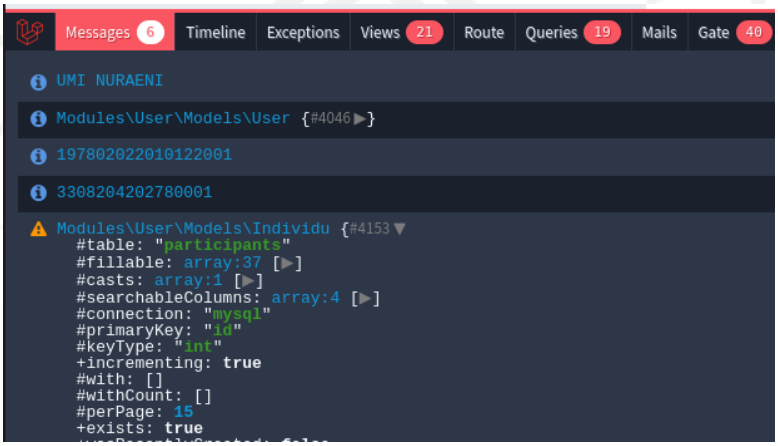
Gambar 3.15 Tampilan nama-nama fail *views* yang mengimpelementasikan *blade extend* pada halamannya



Gambar 3.16 Tampilan *tab Route*

2) *Melakukan print out sebuah variable*

Evaluasi berdasarkan *functional completeness, packages Laravel Debugbar* dapat melakukan *print out* sebuah *variable* dengan *output* pada *tab messages*, yang digunakan untuk melakukan pengecekan *output* sebuah *variable* dalam baris kode terdahulu atau menguji *variable* yang ingin diimplementasikan (Gambar 3.17).

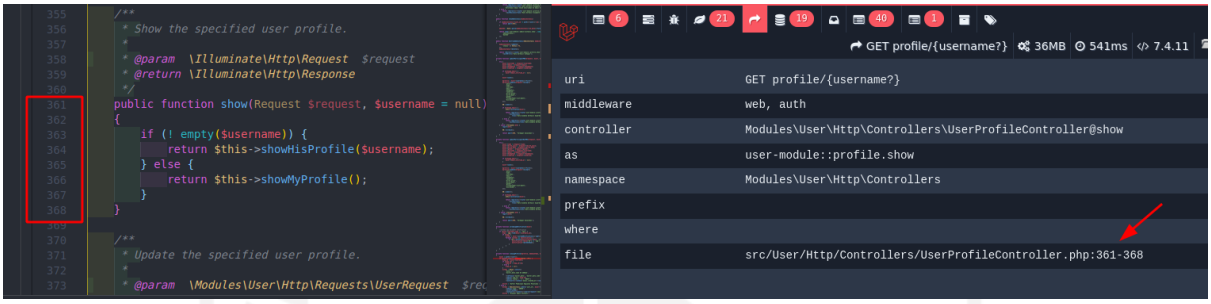


Gambar 3.17 Tampilan *print out variable*

B. *Functional Correctness*, yaitu kemampuan sebuah sistem atau perangkat lunak untuk menyediakan hasil yang benar dengan tingkat presisi yang diperlukan.

1) *Pencarian nama fail views, route, dan controller*

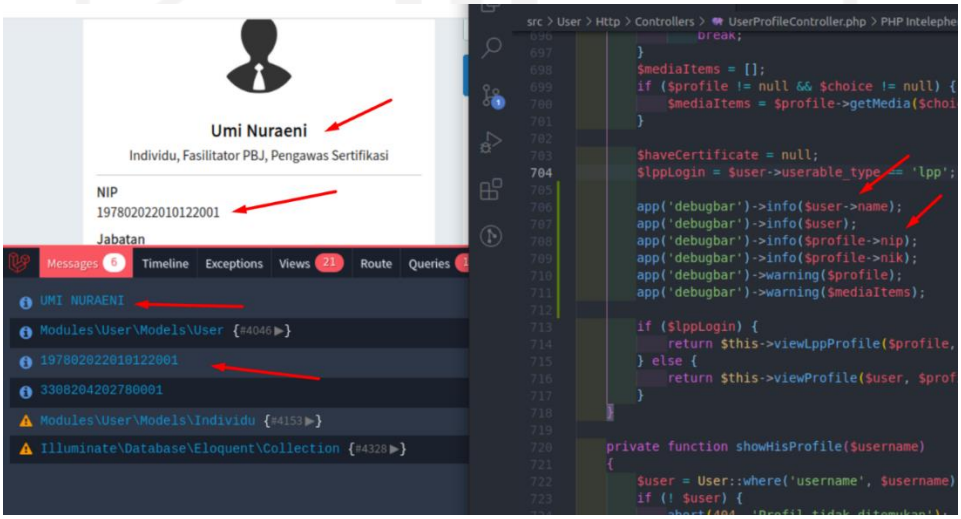
Evaluasi berdasarkan *functional correctness, packages Laravel Debugbar* menyediakan hasil data yang benar mengenai letak baris kode dari *method controller* yang digunakan pada halaman tersebut (Gambar 3.18).



Gambar 3.18 Letak baris *method controller* sesuai berdasarkan informasi dari *tab Route*

2) *Melakukan print out sebuah variable*

Evaluasi berdasarkan *functional correctness, packages Laravel Debugbar* menampilkan hasil data yang benar terkait print out *variable pada tab messages* (Gambar 3.19).

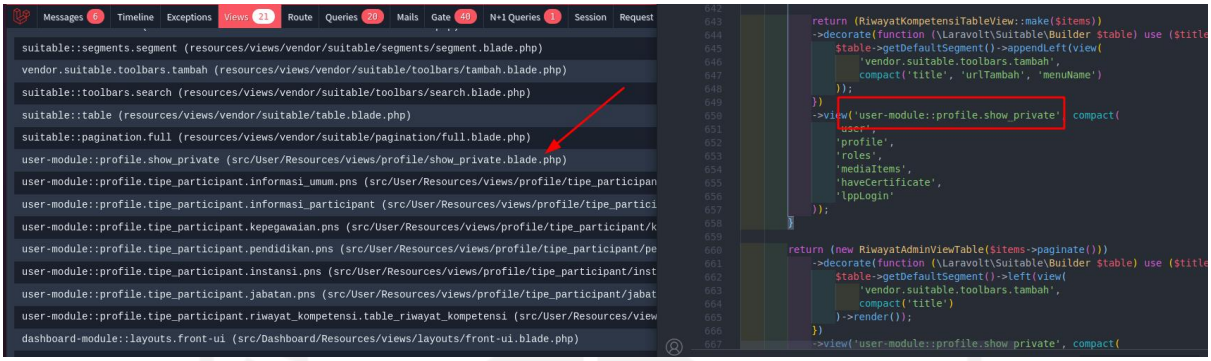


Gambar 3.19 Data yang benar pada saat *print out variable*

C. *Functional Appropriateness*, kemampuan sebuah sistem atau perangkat lunak dengan fungsi yang ada mampu memfasilitasi pencapaian tugas dan tujuan tertentu.

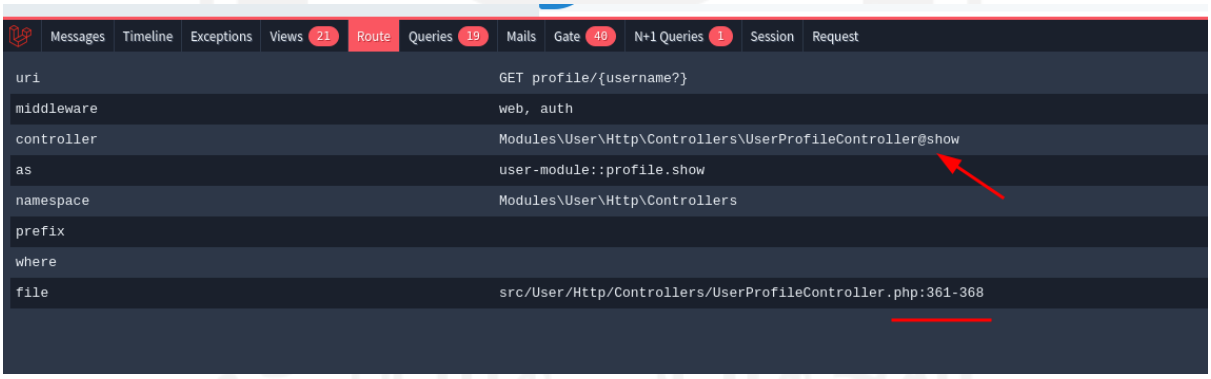
1) *Pencarian nama fail views, route, dan controller*

Evaluasi berdasarkan *functional appropriateness*, pencarian nama fail views berdasarkan *functional appropriateness*, *ViewsCollector* hanya menampilkan informasi nama fail *views* yang di-*load*, tidak dapat memfasilitasi satu nama fail *views* yang menjadi *base html* yang digunakan pada setiap halaman yang diakses (Gambar 3.20).



Gambar 3.20 Tidak memfasilitasi *base html* yang digunakan di halaman tersebut

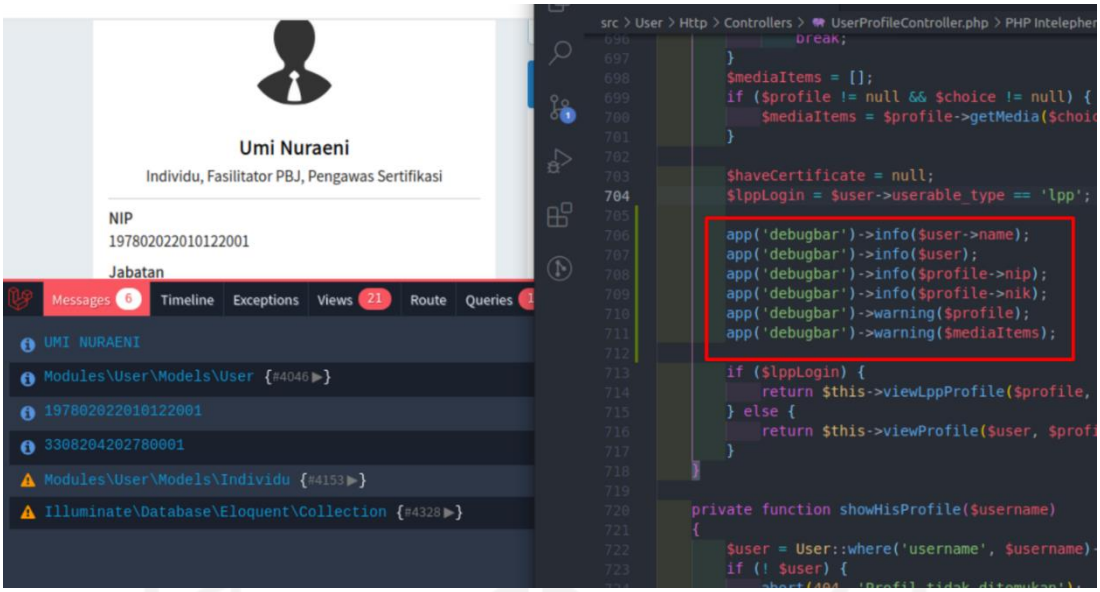
Sedangkan pencarian nama *route dan controller* juga memfasilitasi *programmer* berupa informasi letak baris dari *method controller* yang digunakan pada halaman tersebut (Gambar 3.21).



Gambar 3.21 Informasi tambahan pada *tab Route* berupa letak baris kode *method controller*

2) *Melakukan print out sebuah variable*

Evaluasi berdasarkan *functional appropriateness*, *packages Laravel Debugbar* memang tidak menyediakan sebuah *shortcut synatx* atau pemanfaatan lainnya untuk mempermudah dalam melakukan *print out variable*, namun ada pemanfaatan lain yang mungkin dapat membantu yaitu *packges* ini dapat melakukan *print out* beberapa *variable*. (Gambar 3.22).



Gambar 3.22 *Multiple print out* beberapa *variable*

D. *Time Behavior*, tingkatan waktu respon dan waktu proses serta tingkat keluaran dari sebuah sistem atau perangkat lunak ketika menjalankan fungsinya memenuhi persyaratan.

1) *Pencarian nama fail views, route, dan controller*

Evaluasi berdasarkan *time behavior*, waktu proses yang dibutuhkan oleh *packages Laravel Debugbar* untuk *load* halaman secara penuh termasuk semua *collectors* khususnya *tab views* dan *route.* mendapatkan penambahan waktu yang tidak terlalu banyak dan relatif sama jika tanpa menggunakan *packages* tersebut. Pada Table 3.2 merupakan perbandingan waktu proses *load* halaman utama aplikasi lokal penuh dengan menggunakan *packages Laravel Debugbar* dan tanpa *packages tersebut.*

Table 3.2 Pengujian *load time* halaman utama PPSDM

URL	*Load Time (second)* dengan *packages*	*Load Time (second)* tanpa *packages*
http://127.0.0.1:8000/	10,1	9,93
http://127.0.0.1:8000/	10,6	9,97
http://127.0.0.1:8000/	13,3	9,89
http://127.0.0.1:8000/	9,19	10,4
http://127.0.0.1:8000/	8,32	10,2
Rata-rata	**10,32**	**10,07**

Dari pengujian tersebut dapat diketahui bahwa tidak ada waktu tambahan signifikan antara halaman lokal yang dijalankan dengan *packages Laravel Debugbar* dan tanpa *packages* tersebut yaitu 10,32 detik dan 10,07 detik.

2) Melakukan print out sebuah variable

Evaluasi berdasarkan *time behavior*, waktu proses yang dibutuhkan untuk melakukan *print out variable* pada *tab messages* ternyata tidak mengalami penambahan waktu signifikan, hal ini terjadi karena semua *collectors packages* di-*load* secara bersamaan dengan kata lain tidak memiliki waktu yang signifikan dengan waktu yang dibutuhkan *packages* untuk *load* halaman secara penuh. Pada Table 3.3 merupakan pengujian *load time* yang dibutuhkan pada saat melakukan *print out variable.*

Table 3.3 Pengujian *load time* ketika melakukan *print out variable*

URL	*Load Time (second)* dengan *print out*	*Load Time (second)*
http://127.0.0.1:8000/	11,1	10,5
http://127.0.0.1:8000/profile	7,95	6,52
Rata-rata	**9,25**	**8,51**

Dari pengujian tersebut dapat diketahui bahwa tidak ada perbedaan tambahan waktu yang signifikan antara aplikasi lokal ketika melakukan *print out variable* dengan dengan waktu *load* halaman yaitu 9,25 detik dan 8,51 detik.

E. Resource Utilization, tingkatan jumlah dan jenis sumber daya yang digunakan oleh sistem atau perangkat lunak ketika menjalankan fungsinya memenuhi persyaratan.

1) Pencarian nama fail views, route, dan controller

Evaluasi berdasarkan *resource utilization*, untuk melakukan *load* halaman secara penuh serta *packages Laravel Debugbar* termasuk *tab views* untuk pencarian nama fail *views* dan *tab route* untuk pencarian nama *route* dan *controller*, ternyata pemakaian memori(*memory usage*) yang dibutuhkan tidak memiliki perbedaan yang signifikan. Pada Table 3.4 merupakan pengujian pemakaian memori yang sudah dilakukan.

Table 3.4 Pengujian *memory usage* dengan dan tanpa *packages Laravel debugbar*

URL	*Memory usage*(MB) dengan *packages*	*Memory usage*(MB) tanpa *packages*
http://127.0.0.1:8000/	23,3	17,8
http://127.0.0.1:8000/	23,3	17,9
http://127.0.0.1:8000/	23,3	17,8
http://127.0.0.1:8000/profile	21,6	15,4
http://127.0.0.1:8000/profile	21,7	15,5
Rata-rata	**22,64**	**16,86**

Dari pengujian tersebut dapat diketahui bahwa tidak ada perbedaan pemakaian memori yang signifikan anatar penggunaan *packages Laravel Debugbar* dan tanpa *packages* tersebut yaitu 22,64 MB dan 16,86 MB.

2) Melakukan print out sebuah variable

Evaluasi berdasarkan *resource utilization*, pada saat melakukan *print out variable* ternyata ada perbedaan pemakaian memoriantara penggunaan *packages Laravel*

Debugbar dan tanpa *packages* tersebut. Pada Table 3.5 merupakan pengujian pemakaian memori pada saat melakukan *print out variable* yang sudah dilakukan.

Table 3.5 Pengujian pemakaian memori pada saat melakuan *print out variable*

URL	*Memory usage (MB)* dengan *print out*	*Memory usage*(MB) tanpa *packages*
http://127.0.0.1:8000/	22,3	17,8
http://127.0.0.1:8000/profile	23,8	15,4
Rata-rata	**23,05**	**16,6**

Dari pengujian tersebut dapat diketahui bahwa ada perbedaan yang relatif besar antara pemakaian memori ketika melakukan *print out variable* menggunakan *packages Laravel Debugbar* dan tanpa *packages* tersebut yaitu 23,05 MB dan 16,6 MB.

3.3.2 Tabel dan Analisis Hasil Evaluasi Packages Laravel Debugbar

Pada Table 3.6 merupakan tabel dan penilaian yang sudah dirangkum dari evaluasi *packages Laravel Debugbar* dengan parameter subkarateristik *function suitability* dan *performance efficiency* ISO/IEC 25010 sebagai berikut.

Table 3.6 Hasil evaluasi pemanfaatan *packages Laravel Debugbar* dalam pengerjaan *issue* proyek PPSDM

Parameter Evaluasi	Pemanfaatan *packages Laravel Debugbar* dalam pengerjaan *issue* proyek PPSDM		
	Pencarian nama fail views	***Pencarian nama route dan controller***	***Melakukan print out sebuah variable***
Functional Completeness	Sangat Baik	Sangat Baik	Cukup
Functional Correctness	Sangat Baik	Sangat Baik	Sangat Baik
Functional Appropriateness	Kurang	Sangat Baik	Cukup
Time Behavior	Sangat Baik	Sangat Baik	Sangat Baik
Resource Utilization	Cukup	Cukup	Cukup
Total Skor	**12**	**14**	**12**
Rata-rata	2,4 (Cukup)	2,8 (Sangat Baik)	2,4 (Cukup)

perbaiki font nya dengan timenewroman

A. Hasil evaluasi berdasarkan *functional completeness*

1) Pencarian nama fail *views* menggunakan packages Laravel Debugbar dianggap **sangat baik** dengan asumsi mampu memenuhi kebutuhan *programmer* untuk mencari informasi terkait nama fail *views* yang digunakan pada halaman.

2) Pencarian nama *route* dan *controller* menggunakan packages Laravel Debugbar dianggap **sangat baik** dengan asumsi mampu memenuhi kebutuhan programmer untuk informasi nama route dan controller yang digunakan pada sebuah halaman.

3) Melakukan *print out* sebuah *variable* dianggap **cukup** dengan asumsi karena *Laravel* juga sudah menyediakan fungsi yang sama dapat melakukan *print out* sebuah *variable*.

B. Hasil evaluasi berdasarkan *functional correctness*

1) Pencarian nama fail *views* dianggap **sangat baik** dengan asumsi memberikan hasil data yang benar semua nama fail *views* yang ada pada tab *views*.

2) Pencarian nama *route* dan *controller* dianggap **sangat baik** dengan asumsi memberikan data yang benar terkait nama *route* dan *controller* yang digunakan pada halaman yang diakses.
3) Melakukan *print out* sebuah *variable* dianggap **sangat baik** dengan asumsi *output* yang diberikan pada *tab messages* terbukti benar.

C. Hasil evaluasi berdasarkan *functional appropriateness*
1) Pencarian nama fail *views* dianggap **kurang** dengan asumsi tidak secara spesifik memberikan informasi terkait nama fail *views* yang menjadi *base html*, sehingga perlu mencari nama fail secara manual pada halaman yang diakses.
2) Pencarian nama *route* dan *controller* dianggap **sangat baik** dengan asumsi memfasilitasi informasi tambahan dengan memberikan letak baris kode dari *method controller.*
3) Melakukan *print out* sebuah *variable* dianggap **baik** dengan asumsi dapat memfasilitasi *programmer* dengan *print out* beberapa *variable* sekaligus.

D. Hasil evaluasi berdasarkan *time behavior*
1) Pencarian nama fail *views* dianggap sangat **sangat baik** dengan asumsi waktu proses untuk *load* halaman penuh tidak memiliki perbedaan signifikan jika dibandingkan tanpa menggunakan *packages* itu yaitu rata-rata 10,32 detik dan 10,07 detik.
2) Pencarian nama *route* dan *controller* dianggap **sangat baik** dengan asumsi waktu proses untuk *load* halaman penuh tidak memiliki perbedaan signifikan jika dibandingkan tanpa menggunakan *packages* itu yaitu rata-rata 10,32 detik dan 10,07 detik. Penilaian ini sama karena *packages* tersebut menampilkan informasi secara bersamaan setiap *collectors* sehingga tidak ada perbedaan masing-masing *tab.*
3) Melakukan *print out* sebuah *variable* dianggap **sangat baik** dengan asumsi waktu proses untuk *load* halaman penuh tidak memiliki perbedaan signifikan jika dibandingkan tanpa menggunakan *packages* itu yaitu rata-rata 9,25 detik dan 8,51 detik pada halaman utama dan profil ketika *login* salah satu pengguna.

E. Hasil evaluasi berdasarkan *resource utilization*
1) Pencarian nama fail *views* dianggap **cukup** dengan asumsi ada perbedaan yang memang besar namun tidak terlalu jauh pemakaian memori antara *load packages Laravel Debugbar* dengan tanpa *packages* tersebut yaitu rata-rata 22,64 MB dan 16,86 MB.

2) Pencarian nama *route* dan *controller* dianggap **sangat baik** dengan asumsi ada perbedaan yang memang besar namun tidak terlalu jauh pemakaian memori antara *load packages Laravel Debugbar* dengan tanpa *packages* tersebut yaitu rata-rata 22,64 MB dan 16,86 MB. Penilaian ini sama karena *packages* tersebut menampilkan informasi secara bersamaan setiap *collectors* sehingga tidak ada perbedaan masing-masing *tab.*
3) Melakukan *print out* sebuah *variable* dianggap **cukup** dengan asumsi ada perbedaan yang memang besar namun tidak terlalu jauh pemakaian memori antara *print out variable* dengan *packages Laravel Debugbar* dengan tanpa *packages* tersebut yaitu rata-rata 23,05 MB dan 16,6 MB.

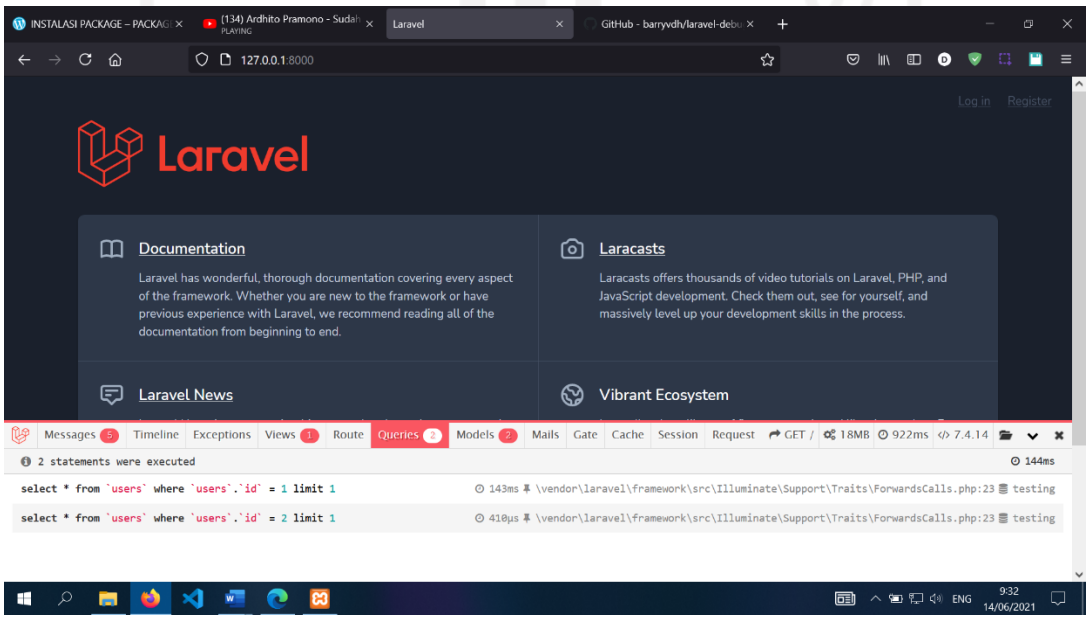
Kesimpulan yang dapat ditarik dari Table 3.6 dari pemanfaatan *packages Laravel Debugbar* dalam mempermudah pengerjaan *issue* proyek PPSDM berdasarkan evaluasi dengan parameter subkarakteristik *function suitability* dan *performance efficiency* menghasilkan kesimpulan sebagai berikut.

- Pencarian nama fail *views* menggunakan *packages Laravel Debugbar* berdasarkan hasil evaluasi 5 parameter dapat **cukup mempermudah** *programmer*. *Packages* ini menyediakan semua nama fail *views* dengan data yang benar. Namun kekurangan utama yaitu tidak ada *highlight* atau tanda nama fail *views* yang menjadi *base html,* hal ini sama saja dengan melakukan pencarian nama fail *views* secara manual.
- Pencarian nama *route* dan *controller* menggunakan *packages Laravel Debugbar* berdasarkan hasil evaluasi 5 parameter **sangat mempermudah** *programmer*. *Packages* ini menyediakan informasi spesifik dan lengkap seperti nama *prefix, uri,* jenis *middleware,* nama *route,* nama *controller, nama method* serta letak baris yang digunakan dengan data yang terbukti benar.
- Melakukan *print out* sebuah *variable* dengan menggunakan *packages Laravel Debugbar* berdasarkan hasil evaluasi 5 parameter dapat **cukup mempermudah** *programmer*. Secara umum *Packages* ini dapat melakukan perintah *print out variable* dengan data yang terbukti benar namun perlu diperhatikan untuk pemakaian memori sekaligus *load time* jika ingin melakukan *print out* menggunakan *packages* ini.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

5.1 Teknis

Packages Laravel Debugbar dapat disebut sebagai pengumpul informasi yang diperlukan oleh saat mengakses halaman tertentu pada pengembangan proyek berbasis website. Informasi tersebut bermanfaat karena membantu menampilkan ringkasan sebuah halaman dibentuk, mulai dari nama fail *views* yang digunakan, nama *route* dan *controller* yang dipakai, *query* data yang dikirimkan pada halaman tersebut dan *collector* lain yang mempunyai informasi spesifik lainnya. Cara instalasi dan penggunaannya juga sederhana yaitu dengan perintah " *composer require barryvdh/laravel-debugbar –dev* " akan menginstal seluruh *dependencies* terkait dengan packages dalam aplikasi, panel *packages Laravel Debugbar* akan muncul dibagian bawah aplikasi dengan posisi *overlay* setiap mengakses halaman di aplikasi (Gambar 5.1). Adapun penggunaanya yaitu dengan mengakses masing-masing *tab* dengan kursor *mouse*. *Packages* ini disarankan hanya digunakan pada lingkup *development* karena setiap melakukan *refresh* halaman akan melakukan *request* ke semua *collector* yang menyebabkan aplikasi melambat, dan pada lingkup *development* pun jika aplikasi berjalan lambat maka disarankan mematikan beberapa *collector*.



Gambar 5.1 Tampilan *packages Laravel Debugbar*

Pemanfaatan *packages Laravel Debugbar* dalam pengerjaan proyek PPSDM hanya menggunakan 3 jenis *collector* yaitu *views, route* dan *messages. Tab* tersebut digunakan untuk mempermudah dalam pencarian nama fail *views, route* dan *controller* serta melakukan *print out* sebuah *variable* untuk melakukan pengecekan *output variable* pada baris kode tertentu sebagai pengganti *function dump and die* atau *dd().* Sedangkan *collector* lain penggunaanya tidak terlalu berdampak signifikan terkait pengerjaan *issue*.

5.2 Non Teknis

5.2.1 Manfaat Magang

Selama proses magang berlangsung, berikut ini merupakan manfaat yang selama magang di PT. Javan Cipta Solusi.

- Manajamen Diri

  Program magang yang dijalankan selama 6 bulan membuat penulis harus dapat beradaptasi dengan aktivitas yang diikuti setiap hari. Mulai dari disiplin waktu serta mengikuti jadwal yang sudah ditetapkan perusahaan, pengerjaan tugas semaksimal mungkin tidak melebih batas waktu yang ditentukan, kualitas serta performa kerja dalam pengerjaan tugas juga harus konsisten. Setiap hari juga harus memiliki target tertentu agar dapat mengetahui *progress* apa yang dicapai dan belum tercapai, mengetahui kendala dan hambatan yang dialami serta bagaimana solusi yang harus digunakan untuk menyelesaikan masalah yang dihadapi. Apabila mengalami masalah terkait aktivitas pengerjaan tugas atau adanya keraguan tentang suatu hal, cara menyampaikan masalah terstruktur dan memiliki etika perlu diperhatikan agar orang lain dapat menanggapi positif sehingga mendapatkan jawaban atau solusi dalam penyelesaian masalah. Contoh-contoh diatas harus dilakukan secara terus menerus agar melatih proses manajemen dalam diri penulis tetap konsisten dan teratur.

- Tanggung Jawab

  Amanah dari setiap tugas yang diberikan ke penulis harus dapat dijalankan dengan semaksimal mungkin serta menghasilkan kualitas yang baik pula. Penugasan yang diberikan bukan semata untuk menyelesaikan permasalahan yang ada dalam proyek, namun membantu membentuk sebuah *mindset* terkait apapun jenis dan tingkatan tugas yang diberikan akan selalu berusaha untuk dapat menyelesaikannya dengan

cara apapun, apabila mengalami kesulitan tentu dapat meminta bantuan dengan *senior programmer* atau teman magang lain.

- Pengalaman dan Ilmu

  Selama proses magang banyak pengalaman serta ilmu baru yang didapatkan dari pengerjaan tugas maupun aktivitas diselenggarakan oleh perusahaan. Penugasan yang diberikan mengharuskan penulis belajar untuk menambah pengetahuan pemrograman baru untuk menyelesaikan permasalahan tersebut. Seiring pengerjaan berbagai jenis tugas membuat pengalaman semakin bertambah sehingga mempermudah dalam menyelesaikan kasus yang sama atau serupa. Selain itu ilmu-ilmu yang diberikan oleh *senior programmer* atau teman magang lain juga menambah wawasan baru terkait pemrograman secara teknis atau praktek maupun secara teori.

- Penggunaan *tools* baru

  Selama proses pengerjaan tugas-tugas proyek, ada beberapa *tools* atau alat yang dapat mempermudah pekerjaan yang menurut penulis dapat digunakan sebagai standar seorang *programmer* khususnya bahasa PHP dengan *framework Laravel.* Berikut ini beberapa *tools* yang digunakan penulis selama magang di PT Javan Cipta Solusi.

  a. *Packages Laravel Debugbar*

     Packages ini memiliki fungsi untuk menampilkan ringkasan infromasi suatu halaman terbentuk mulai dari *view, model, route, controller, query, exception* dan sebagainya. Dengan adanya *packages* ini tentu memudahkan karena memberikan informasi ringkasan informasi halaman dengan mudah dan cepat tanpa harus mengakses atau mencari fail dari setiap folder yang ada pada proyek. Dari semua *collectors* yang ada pada *packages* ini, *collectors query* dan *route* merupakan salah satu yang terbaik karena menampilkan seluruh *query* data yang digunakan *controller* dan informasi nama *route, prefix, middleware, controller* beserta letak kode *method controller* halaman yang diakses.

  b. *Ekstensi Sonarlint Vscode*

     Ekstensi dari *visual studio code* ini berfungsi untuk mengidentifikasi dan menganalisis setiap kode yang ada dalam proyek, apabila terindikasi melanggar aturan *sonarlint* maka akan ada *feedback* berupa peringatan bahwa

kode sedang ditulis atau kode fail yang sedang diakses terdapat kesalahan dari segi penulisan, penggunaan *function* yang kurang baik atau pelanggaran lain. Selain itu ekstensi ini juga dapat mengidentifikasi *bug, code smells,* dan *security vulnerabilities.* Ekstensi ini membantu agar kode yang ada sudah sesuai dengan standar bahasa pemrograman serta menciptakan *clean code.* Adapun *rule* dari ekstensi *sonarlint* sudah mendukung beberapa bahasa pemrograman.

c. *Git Gui Client* dengan *Gitkraken*

Penulis memang belum pernah terlibat proyek secara langsung yang menggunakan *git* sebagai *version control system*. Pengetahuan dasar mengenai *git* hanya sebatas praktik menggunakan *syntax* pada *console* dan belum mengetahui bahwa sebenarnya ada *git* berbasis *gui*(*graphic user interface*), salah satunya adalah *Gitkraken.* Penulis sangat dimudahkan dengan adanya *gitkraken* karena mudah dioperasikan serta mudah dipahami dari setiap fungsi yang ada dibandingkan dengan *git* berbasis *syntax*. Hal utama dari yang penggunaan dari *git* berbasis *gui* adalah dapat melihat *timeline branch* setiap kali *developer push* kode terbaru, dari *timeline* tersebut penulis dapat melihat perubahan-perubahan kode yang dilakukan oleh *programmer* lain pada *timeline*.

5.2.2 Hambatan Selama Magang

Selama proses magang berlangsung, berikut ini merupakan hambatan yang dialami selama magang di PT. Javan Cipta Solusi.

- Kinerja OS Lambat

  Penggunaan OS (*Operating System*) Ubuntu sejak 4 bulan terakhir setelah terlibat ke dalam proyek ternyata tidak sesuai dengan ekspektasi. Salah satu standarisasi yang diberlakukan untuk *programmer* di PT. Javan Cipta Solusi adalah menggunakan sistem operasi Ubuntu sebagai *daily driver* utama semua aktivitas seperti pengerjaan *issue,* mengikuti *training* perusahaan, dan lain sebagainya. Pada awal penggunaan sistem operasi ini tidak terlalu berdampak signifikan terhadap aktivitas, namun semakin lama kinerja dari sistem operasi ini melambat. Asumsi utama karena penggunaan aplikasi wajib dari perusahaan yang memakan banyak kapasitas *ram*, namun kemungkinan utama karena kemampuan dari *Input/Ouput Hardisk* yang

sudah menurun menyebabkan kinerja OS melambat. Hal ini sangat menyulitkan untuk melakukan *multitasking* pekerjaan karena harus membatasi penggunaan beberapa aplikasi sekaligus secara bersamaan karena dapat berakibat *hang*.

- Membaca dan Memahami Kode dalam proyek

  Pengalaman atau jam terbang yang kurang banyak yang membuat pemahaman tentang maksud dan alur kode menjadi pekerjaan tambahan. Dampak terburuknya adalah kode yang diimplementasikan bahkan belum mendekati apa yang seharusnya kode tersebut ditulis, hal ini tentu akan menambah *effort* lebih dari segi waktu yang dihabiskan, *resource* yang diperlukan hingga tenaga yang dikeluarkan hanya untuk membaca dan memahami kode. Masalah bahkan belum selesai apabila pemahaman kode sudah baik namun belum bisa mengimplementasikan kode yang diinginkan.

5.2.3 Tantangan Selama Magang

Selama proses magang berlangsung, berikut ini merupakan tantangan yang harus dihadapi selama magang di PT. Javan Cipta Solusi.

- Code yang tidak *readable*, efektif dan efisien

  Kode yang selesai dikerjakan harus menghasilkan kode yang mudah dibaca dan dipahami oleh semua *programmer*, efektif dalam menggunakan *resource* seperti *function* tertentu dalam mengimpelementasikan kode yang diinginkan dan efisien dengan banyaknya jumlah baris kode yang dihasilkan.

- Mengimplementasikan kode yang baru dipelajari

  Dengan beragam permasalahan *issue* yang sudah dikerjakan, semakin berbeda pula pendekatan yang diambil dalam menyelesaikannya. Salah satu pendekatannya adalah mengimplementasikan kode-kode yang belum pernah digunakan, hal ini menjadi tantangan karena setiap *issue* memiliki batas waktu yang sudah ditetapkan. *Programmer* diharapkan dapat beradaptasi dengan cepat untuk mempelajari dan mengerti kode baru sehingga menjadi solusi penyelesaian dari *issue* yang dikerjakan.

- Menyelesaikan *issue* sebanyak mungkin

  Salah satu bentuk umum sebuah kontribusi dalam sebuah tim adalah seberapa banyak tugas atau *issue* yang sudah dikerjakan. Dengan semakin banyaknya tugas atau *issue* yang sudah dikerjakan hal ini membuktikan *skills* yang dimiliki seorang *programmer* sudah berkembang dengan pesat karena dapat menyelesaikan banyak tugas dengan kurun waktu yang singkat. Hal ini tentu menjadi penilaian bagi *senior programmer*

untuk menemukan dengan cepat *junior programmer* yang memiliki bakat dan potensi tersembunyi agar *programmer* tersebut dapat menempati tingkatan tugas yang sebanding dengan kemampuan yang dimiliki.

BAB V
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil evaluasi yang dilakukan untuk menilai pemanfaatan *packages Laravel Debugbar* dalam mempermudah pengerjaan *issue* proyek PPSDM, berikut ini kesimpulan yang dapat diambil.

a. Parameter evaluasi dari subkarakterisitik *function suitability* dan *performance efficiency* ISO/IEC 25010 yaitu *functional completeness, functional correctness, functional appropriateness, time behavior* dan *resource utilization* dapat menjadi parameter evaluasi dalam menilai pemanfaatan *packages Laravel Debugbar*. Pengukuran parameter pada beberapa jenis pemanfaatan *packages Laravel Debugbar* yang diteliti atau dikerjakan menunjukkan skor evaluasi yang beragam. Hal ini terjadi karena dari segi fungsional, *packages* ini harus dapat menampilkan informasi mengenai pemanfaatan *packages* tersebut dengan data yang valid serta menyediakan fasilitas berupa informasi tambahan agar lebih mempermudah *programmer* untuk mendapatkan tujuan yang diinginkan. Dari segi efisiensi performa, *packages* ini seharusnya dapat mengolah sumber daya yang dimiliki sehingga menampilkan informasi baru yang spesifik atau lengkap, dengan waktu proses dan respon secara fungsional lebih cepat dibandingkan dengan cara manual.
b. Dari hasil evaluasi menggunakan parameter dari subkarakteristik *function suitability* dan *performance efficiency* ISO/IEC 25010 yaitu *functional completeness, functional correctness, functional appropriateness, time behavior* dan *resource utilization* menujukkan skor evaluasi pemanfaatan *packages Laravel Debugbar* yaitu 1) pencarian nama fail *views* dengan rata-rata skor 2,4 atau cukup, 2) pencarian nama *route* dan *controller* dengan rata-rata skor 2,8 atau sangat baik, dan 3) melakukan *print out* sebuah *variable* dengan rata-rata skor 2,4 atau cukup. Dari ketiga skor tersebut kemudian ditarik rata-rata skor dengan hasil yaitu 2,53 atau cukup. Dengan begitu dapat disimpulkan pemanfaatan *packages Laravel* yang diteliti atau dikerjakan dengan hasil skor cukup atau *packages* ini cukup membantu mempermudah pengerjaan *issue* proyek PPSDM.

6.2 Saran

Dalam penulisan laporan tugas akhir ini, ada beberapa catatan yang perlu disampaikan agar kedepannya menjadi bahan evaluasi secara bersama sebagai berikut.

a. Adapun skala penilaian evaluasi yang telah dilakukan masih menggunakan asumsi penulis berdasarkan fakta yang terjadi dilapangan, hal ini seharusnya dapat dibuktikan dengan skala penilaian yang jelas serta dapat diukur.
b. Dengan skala penilaian yang masih bersifat asumsi, penarikan kesimpulan juga tidak mempunyai acuan mutlak bagaimana data evaluasi diolah dan disajikan sehingga menghasilkan kesimpulan yang masih mengambang dan kurang terbukti kebenarannya, dan tidak bisa dijadikan standar hasil yang seharusnyaa.
c. Untuk pengujian fungsionalitas dari *packages Laravel Debugbar* tidak tepat jika dibandingkan dengan fungsi umum *print out variable* dari *Laravel yaitu function dd()*, karena tidak sepadan terkait dengan fungsi yang dapat dilakukan serta keluaran yang dihasilkan oleh keduanya.

DAFTAR PUSTAKA

Ackerman, A. F., Buchwald, L. S., & Lewski, F. H. (1989). Software inspections: an effective verification process. *IEEE Software*, *6*(3), 31–36.

DR Dako, R., & Ridwan, W. (2021). Pengujian karakteristik Functional Suitability dan Performance Efficiency tesadaptif.net. *Jambura Journal of Electrical and Electronics Engineering*, *3*(2), 66–71.

Friedman, H., Wilamowsky, Y., & Friedman, L. (1981). A Comparison of Balanced and Unbalanced Rating Scales. *Mid-Atlantic Journal of Business*, *19*, 1–7.

Heuvel, B. vd. (2021). *barryvdh/laravel-debugbar:Laravel Debugbar - Github*. https://github.com/barryvdh/laravel-debugbar

Hung, C.-Y., Sun, J. C.-Y., & Yu, P.-T. (2015). The benefits of a challenge: student motivation and flow experience in tablet-PC-game-based learning. *Interactive Learning Environments*, *23*, 172–190.

ISO/IEC. (2011). *ISO / IEC 25010 : 2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*.

PT Javan Cipta Solusi: Home. (2019). https://javan.co.id/

Riduwan. (2017). *Metode Penelitian untuk Tesis*. Alfabeta.

Septianto, A., & Ade Sekarwati, K. (2019). The Analysis of Academic Information System in The Aerospace Air Marshal Suryadarma University Using ISO/IEC 25010. *Jurnal Ilmiah Bidang Teknologi, ANGKASA*, *6*(2), 140–151.

TechTarget. (2017). *What is production server? - Definition from WhatIs.com*. https://whatis.techtarget.com/definition/production-server

Wilkie, K. (2016). Rise or Resist: Exploring Senior Secondary Students' Reactions to Challenging Mathematics Tasks Incorporating Multiple Strategies. *Eurasia Journal of Mathematics, Science and Technology Education*, *12*, 2061–2083.

1. LAMPIRAN