JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

IMPLEMENTASI UNIT TESTING DAN END-TO-END TESTING PADA SISTEM INFORMASI AKADEMIK TEKNIK INFORMATIKA

Muhammad Ghozy Alkhairi[1)], Syarifah Putri Agustini Alkadri[2)], Putri Yuli Utami[3)]
1. Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Muhammadiyah Pontianak, Indonesia
2. Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Muhammadiyah Pontianak, Indonesia
3. Sistem Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Muhammadiyah Pontianak, Indonesia

Article Info
Kata Kunci: Pengujian perangkat lunak , Unit testing, Basis path, End-to-end testing

Keywords: Software testing, Unit testing, Basis path, End-to-end testing

Article history:
Received 29 September 2024
Revised 13 Oktober 2024
Accepted 4 November 2024
Available online 4 December 2024

DOI :
https://doi.org/10.29100/jipi.v9i4.5626

* Corresponding author.
Muhammad Ghozy Alkhairi
E-mail address:
inighozy@gmail.com

ABSTRAK
Pengujian perangkat lunak bertujuan untuk mengidentifikasi bug, kesalahan logika, atau potensi masalah lainnya yang dapat memengaruhi kinerja dan keamanan perangkat lunak. Sistem Informasi Manajemen Akademik Teknik Informatika (SIMATIK) adalah salah satu sistem informasi yang digunakan di Universitas Muhammadiyah Pontianak yang digunakan untuk memfasilitasi penilaian kerja praktik dan tugas akhir. Penting untuk memastikan bahwa SIMATIK beroperasi dengan baik dan bebas dari *bug* yang dapat mengganggu proses penggunaan sistem oleh mahasiswa maupun dosen. Oleh karena itu, diimplementasikan *end-to-end testing* dan *unit testing* pada pengujian SIMATIK. Teknik yang digunakan pada unit testing yaitu basis path yang dimulai dari membuat flowgraph, menghitung *cyclomatic complexity* untuk menentukan jalur pengujian dan melakukan unit test. Tahapan yang dilakukan dalam *end-to-end testing* dimulai dari menyusun skenario pengujian yang mencakup daftar fitur yang dipilih dan melakukan *end-to-end test.* Setelah dilakukan pengujian aplikasi menggunakan metode unit testing dan *end-to-end testing* tidak ditemukan kesalahan, maka pengujian dipastikan berhasil.

ABSTRACT
The process of software testing aims to identify potential issues such as bugs and logic errors that could impact the performance and security of the software. At Muhammadiyah University Pontianak, they use an information system called the Academic Management Information System for Informatics Engineering (SIMATIK) to evaluate practical work and final assignments. To ensure SIMATIK runs smoothly and is free of bugs that could disrupt the experience for both students and faculty, the testing process involves both end-to-end testing and unit testing. Unit testing involves utilizing the basis path technique, which requires creating a flowgraph, calculating the cyclomatic complexity to determine testing paths, and executing unit tests. In contrast, end-to-end testing begins with formulating testing scenarios that cover selected feature lists, followed by the actual end-to-end testing. Once the application testing has been conducted through both unit testing and end-to-end testing methods and no errors have been found, the testing is considered successful.

.

I. PENDAHULUAN

PERANGKAT lunak adalah istilah umum yang digunakan untuk menunjukkan informasi data di dalam komputer serta kumpulan instruksi yang terorganisir. Dalam konteks ini, perangkat lunak bertanggung jawab untuk mengendalikan, mengelola, dan mengintegrasikan komponen perangkat keras dari sebuah sistem komputer dengan tujuan menyelesaikan tugas atau perintah tertentu [1].

Pengujian perangkat lunak adalah salah satu tahapan penting dalam siklus pengembangan perangkat lunak yang bertujuan untuk memastikan bahwa perangkat lunak yang dibangun berfungsi dengan baik, memiliki kualitas yang baik, dan dapat memenuhi kebutuhan pengguna. Pengujian perangkat lunak bertujuan untuk mengidentifikasi bug, kesalahan logika, atau potensi masalah lainnya yang dapat memengaruhi kinerja dan keamanan perangkat lunak [2]. Suatu pengujian juga penting dilakukan karena dapat memastikan apakah program yang digunakan telah beroperasi sesuai dengan kebutuhan-kebutuhan yang diperlukan [3].

Sistem Informasi perpustakaan adalah sebuah sistem dalam organisasi publik yang dirancang untuk memenuhi

kebutuhan pengelolaan transaksi [4]. Pada era ini, penggunaan sistem informasi dalam berbagai sektor, termasuk pendidikan, telah menjadi hal yang tidak terhindarkan. Sistem Informasi Manajemen Akademik Teknik Informatika (SIMATIK) adalah salah satu contoh sistem informasi yang digunakan di Universitas Muhammadiyah Pontianak. Saat ini, SIMATIK digunakan untuk memfasilitasi penilaian kerja praktik dan tugas akhir.

Penting untuk memastikan bahwa SIMATIK beroperasi dengan baik dan bebas dari *bug* yang dapat mengganggu proses penggunaan sistem oleh mahasiswa maupun dosen. Oleh karena itu, pengujian perangkat lunak, termasuk pengujian tingkat unit, menjadi krusial dalam memastikan kualitas dan keandalan SIMATIK.

Namun pada proses pengembangan SIMATIK, tahap pengujiannya belum dilakukan secara optimal. Berdasarkan survei, menurut beberapa mahasiswa teknik informatika, proses pengujian pada aplikasi SIMATIK diperlukan karena sebelumnya pernah menemukan beberapa galat pada fungsionalitas aplikasi.

Padahal pengujian merupakan langkah yang sangat penting untuk mengurangi kemungkinan terjadinya kesalahan yang dapat merugikan suatu program [5]. Menurut sumber lain, pengujian sangat penting karena bertujuan untuk menjamin kualitas perangkat lunak, serta dapat berfungsi sebagai pemeriksaan akhir terhadap pengkodean, desain, dan spesifikasi [6].

Unit testing, juga dikenal sebagai *component testing* dilakukan dengan mencari cacat dan memverifikasi fungsi item perangkat lunak (misalnya modul, program, objek dan kelas) yang dapat diuji secara terpisah. *Unit testing* umumnya didasarkan pada persyaratan dan spesifikasi desain rinci yang berlaku untuk komponen yang sedang diuji, serta pada kode itu sendiri[2].

Komponen yang sedang diuji dapat mencakup komponen-komponen individual, tabel basis data, modul, prosedur, integritas referensial, dan batasan lapangan, bahkan seluruh basis data. Unit testing dilakukan menggunakan pengujian *white box*. Salah satu teknik pengujian *white box* adalah *basis path*. *Basis path* adalah teknik pengujian yang dilakukan dengan beberapa tahapan yaitu dengan membuat *flowgraph* dari fungsi yang akan diuji, menghitung *cyclomatic complexity* dan melakukan *unit test* [7]. Flowgraph adalah grafik program yang dihasilkan dari pemetaan *flowchart* program yang ada untuk merepresentasikan aliran kontrol logika dalam program tersebut [8]. Flowgraph digunakan dalam pengujian yang berfokus pada visualisasi aliran dari suatu program [9].

Unit test pada aplikasi SIMATIK dilakukan menggunakan *Pest*. *Pest* adalah suatu kerangka pengujian (*testing framework*), yang dirancang untuk menyederhanakan proses pengujian dalam pengembangan perangkat lunak. *Pest* menyediakan sintaksis yang lebih singkat dan mudah dibaca. Ini membuatnya lebih mudah untuk menulis dan memahami kode pengujian. *Pest* juga dirancang untuk bekerja dengan baik dengan *Laravel*, salah satu *framework* PHP yang populer [10].

End-to-end testing, juga dikenal sebagai *system testing* atau *functional testing*, pada dasarnya adalah pengujian perangkat lunak dari awal hingga selesai, termasuk integrasi, ketergantungan, dan ketersediaan. Dalam pengujian *end-to-end*, persyaratan pengujian dapat langsung diterjemahkan menjadi kasus penggunaan, contohnya ketika seorang pelanggan membeli produk dari situs *e-commerce* [11].

Pengujian *end-to-end bisa* diotomatisasi untuk menyediakan rangkaian regresi guna memastikan bahwa setiap perubahan pada sistem tidak memengaruhi fungsionalitas yang sudah ada secara negatif. Pemangku kepentingan dapat menggunakan informasi dari pengujian sistem untuk memutuskan apakah sistem sudah siap untuk pengujian penerimaan pengguna (UAT) [11].

End-to-end testing pada aplikasi SIMATIK dilakukan dengan menggunakan *Cypress*. *Cypress* adalah *framework end-to-end testing* yang populer. *Cypress* dirancang untuk memungkinkan pengujian *end-to-end* dengan mudah. Pengujian *end-to-end* ini melibatkan simulasi perilaku pengguna nyata, termasuk interaksi dengan elemen UI pada sebuah *website* [12]. *Cypress* juga memiliki performa yang lebih baik ketika digunakan untuk pengujian website saat dibandingkan dengan *end-to-end testing tools* lain yang juga populer yaitu *Selenium* [13].

Dengan mengikuti metode pengujian *end-to-end*, memungkinkan untuk menguji program pada semua sistem target yang telah direncanakan. Metode ini memverifikasi keluaran yang diinginkan dari setiap *input* dalam aplikasi. Metode ini umumnya digunakan untuk menguji pengalaman pengguna dengan aplikasi tersebut[14].

Kedua metode tersebut dipilih untuk pengujian aplikasi SIMATIK karena menawarkan cakupan pengujian yang komprehensif dan mendalam dari berbagai aspek sistem. *Unit testing* memastikan bahwa setiap komponen atau modul kecil dari aplikasi berfungsi dengan benar secara individu. Sementara itu, *end-to-end testing* memeriksa alur kerja dan integrasi antar komponen dalam lingkungan yang menyerupai kondisi nyata penggunaan oleh pengguna akhir. Dengan demikian, kombinasi kedua metode ini dapat menjamin bahwa aplikasi tidak hanya bebas dari kesalahan pada level unit, tetapi juga mampu menjalankan skenario penggunaan secara keseluruhan dengan baik, sehingga meningkatkan kualitas dan keandalan sistem.

Penelitian yang berkaitan dengan unit testing maupun *end-to-end testing* telah dilakukan pada beberapa penelitian. Dalam penelitian yang berjudul "Pemanfaatan *Cypress* Untuk Pengujian *End-to-End* (Studi Kasus: Pengembangan Aplikasi Indicar)". Penelitian tersebut menerapkan metode *end-to-end testing* dengan *Cypress* yang

merupakan salah satu dari dua metode pengujian yang akan penulis lakukan pada pengujian aplikasi SIMATIK. Tahapan-tahapan yang dilakukan pada pengujian tersebut antara lain pembuatan skenario pengujian, pemilihan *testing tools*, validasi skenario pengujian, pembuatan kode pengujian, eksekusi kode pengujian dan memberikan hasil pengujian kepada pengembang aplikasi [15]. Pada penelitian kedua yang berjudul "Unit Testing Pada Aplikasi Web Mobile (Studi Kasus Bisnis Jasa *Laundry*)". Pada penelitian tersebut dilakukan pengujian aplikasi menggunakan metode unit testing dengan *framework* PHPUnit, metode pengujian yang juga akan digunakan pada pengujian aplikasi SIMATIK. Beberapa tahapan yang dilakukan pada pengujian antara lain, memilih beberapa fungsi yang memiliki kondisi percabangan, mengonversikan *source code* ke dalam bentuk *flowgraph* kemudian menghitung nilai *cyclomatic complexity* untuk mendapatkan jalur *independent* [16].

Oleh karena itu, penulis akan membahas tentang implementasi *end-to-end testing* dan unit testing pada pengujian SIMATIK. Dengan memfokuskan pada kedua metode pengujian tersebut, penelitian ini bertujuan untuk memastikan kualitas dan keandalan SIMATIK.

II. METODE PENELITIAN

Pengujian sistem dilakukan dengan metode *unit testing* dan *end-to-end testing* terhadap fungsi-fungsi serta fitur dan skenario yang dipilih. Penulis akan melakukan unit testing menggunakan teknik *basis path*. Teknik ini memungkinkan penguji untuk menilai kompleksitas logika dalam desain prosedural. Skenario uji coba yang dikembangkan untuk menguji Teknik *basis path* ini menjamin bahwa setiap pernyataan dalam aplikasi yang diuji akan dieksekusi setidaknya sekali selama tahap pengujian [17]. Teknik ini juga digunakan untuk mengetahui kompleksitas logika aplikasi [18]. Tahapan yang dilakukan dalam pengujian unit ini adalah:

1. Menetapkan kode program atau fungsi yang akan menjadi subjek uji.
2. Mengubah *source code* ke dalam bentuk *flowgraph*.
3. Melakukan perhitungan nilai *cyclomatic complexity* untuk menentukan jumlah jalur dasar.
4. Melakukan unit test.

Untuk menghitung nilai *cyclomatic complexity* digunakan rumus sebagai berikut [19]:

$$V(G) = E - N + 2 \quad (1)$$

V(G) : *Cyclomatic Complexity*
E : Jumlah *edge*
N : Jumlah *node*
Rentang nilai V(G) yang didapat memiliki arti yang disajikan pada tabel 1 [20].

TABEL I
NILAI CYCLOMATIC COMPLEXITY

Nilai V(G)	Makna
0-5	*Code* terstruktur dengan baik dan mudah diuji.
6-10	*Code* cukup kompleks dan relatif sulit diuji
>10	*Code* memiliki kompleksitas yang sangat tinggi dan sangat sulit diuji. Disarankan refaktorisasi.
0-5	*Code* terstruktur dengan baik dan mudah diuji.
6-10	*Code* cukup kompleks dan relatif sulit diuji
>10	*Code* memiliki kompleksitas yang sangat tinggi dan sangat sulit diuji. Disarankan refaktorisasi.

Setelah memastikan tidak ada kesalahan yang ditemukan pada fungsionalitas yang diuji dalam unit testing, penulis melanjutkan pengujian dengan pengujian *end-to-end* untuk memastikan alur kerja aplikasi sudah sesuai spesifikasi dan tidak terdapat kesalahan. Tahapan yang dilakukan dalam pengujian *end-to-end* antara lain perancangan skenario pengujian yang mencakup daftar fitur yang akan diuji dan target hasil yang diharapkan. Salah satu skenario yang diuji yaitu ketika mahasiswa yang menggunakan aplikasi sedang mengisi formulir untuk

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

mengajukan proposal skripsinya. Setelah merancang skenario pengujian maka dilakukan pengujian *end-to-end*.

End-to-end testing dapat dikerjakan secara manual apabila aplikasi yang diuji memiliki ruang lingkup yang tidak terlalu besar dan tidak ada rencana untuk penambahan fitur secara berkala. Sebaliknya, jika aplikasi yang diuji memiliki lingkup yang besar dan ada rencana penambahan fitur, maka dapat digunakan *automated software testing tools* [21]. Dalam konteks pengujian *end-to-end* pada aplikasi SIMATIK yang memiliki lingkup cukup besar dan direncanakan penambahan fitur maka digunakan salah satu *automated testing tools* yang umum digunakan yaitu *Cypress*.

III. HASIL DAN PEMBAHASAN

Proses pengujian unit dilakukan dengan memilih fungsi yang memuat percabangan, selanjutnya menghitung jumlah kondisi pada setiap fungsi. Sehubungan dengan keterbatasan dalam penyajian laporan ini, hanya beberapa fungsi di yang akan dibahas. Pada tabel 1 terdapat daftar fungsi terdapat pada setiap kelas pada aplikasi SIMATIK yang akan diuji.

TABEL II
FUNGSI PENGUJIAN

No	Nama Kelas	Nama Fungsi
1-B	DocumentTAControl-ler	submitProposalTA
2-B	PengujiSemproCon-troller	submitPenilai-anSempro

Aplikasi SIMATIK diuji menggunakan dua metode, salah satunya unit testing menggunakan *framework Pest* dengan penerapan teknik *basis path*. Sejumlah fungsi yang memiliki percabangan dipilih untuk diujikan. Pengujian dianggap berhasil setelah dieksekusi dan semua jalur pengujian menghasilkan *output* PASSED.

A. *Rancangan Unit Test 1-B*

Kelas *DocumentTAController* memiliki fungsi *submitProposalTA()* yang berfungsi untuk mahasiswa mengajukan proposal skripsinya dengan mengisi formulir yang berisikan *input* judul, dokumen proposal, dokumen hasil cek plagiat, dokumen kartu hadir sempro/sidang dan dokumen sertifikat PKKMB yang opsional. Berikut adalah *source code* dari fungsi *submitProposalTA()*:

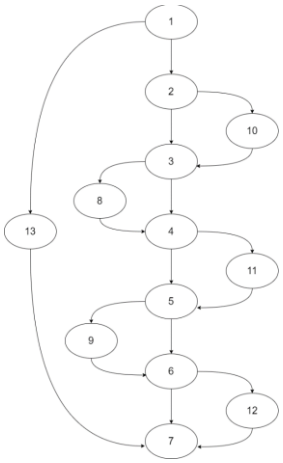
submitProposalTA()

```
public function submitProposalTA(DocumentTA $document, SubmitProposalTARequest $submitTARequest)
    {
        // * 1
        if ($document->status_proposal == 'submitted' || $document->status_proposal == 'diterima') {
            // * 13 - 7
        $data['status_proposal'] = 'submitted';

         .....

        // * 2
        if ($submitTARequest->sertifikat_pkkmb) {
            // * 10
            $namaFile = auth()->user()->dataTambahanMahasiswa->nim . '_' . 'sertifikat_pkkmb';
         .....
        }
        // * 3
        if ($document->proposal_skripsi == $pdfPath1) {
            // * 8
   Storage::delete('document/proposalTA/' . $namaFile);
        }
        // * 4
        if ($document->plagiat == $pdfPath2) {
            // * 11
            Storage::delete('document/plagiat/' . $namaFile);
        }
        // * 5
        if ($document->kartu_hadir == $pdfPath3) {
            // * 9
            Storage::delete('document/kartu_hadir/' . $namaFile);
        }
        // * 6
        if ($document->sertifikat_pkkmb) {
            // * 12
            Storage::delete('document/sertifikat/' . $namaFile);
        }
        $document->update($data);
```

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

```
    // * 7
    return back()->with('success', 'Proposal TA berhasil disubmit.');
}
```

Pengubahan kode sumber (*source code*) menjadi *flowgraph* pada fungsi *submitProposalTA()* untuk menentukan jalur yang dapat dilalui dapat diamati pada gambar 1.



Gambar 1 *Flowgraph* fungsi submitProposalTA

Berdasarkan *flowgraph* dari fungsi *submitProposalTA()* di atas terdapat 12 *node* (N) dan 17 *edge* (E). Kemudian dihitung nilai *cyclomatic complexity* menggunakan rumus (1) sehingga didapatkan nilai V(G) = E – N + 2 = 18 – 13 + 2 = 7. Maka dapat ditentukan fungsi *submitProposalTA()* memiliki 7 jalur. Adapun kemungkinan jalur yang akan diuji antara lain:

1) Jalur 1-BA = 1, 2, 10, 3, 4, 5, 6, 7
   Mahasiswa mengisi semua formulir proposal skripsi dan sebelumnya belum pernah meng-*upload* semua dokumen
2) Jalur 1-BB = 1, 2, 3, 4, 5, 6, 7
   Mahasiswa mengisi semua formulir proposal skripsi kecuali dokumen pkkmb dan sebelumnya belum pernah meng-upload semua dokumen
3) Jalur 1-BC = 1, 2, 3, 8, 4, 5, 6, 7
   Mahasiswa mengisi semua formulir proposal skripsi dan sebelumnya pernah meng-*upload* dokumen proposal
4) Jalur 1-BD = 1, 2, 3, 4, 11, 5, 6, 7
   Mahasiswa mengisi semua formulir proposal skripsi dan sebelumnya pernah meng-*upload* dokumen cek hasil plagiat
5) Jalur 1-BE = 1, 2, 3, 4, 5, 9, 6, 7
   Mahasiswa mengisi semua formulir proposal skripsi dan sebelumnya pernah meng-*upload* dokumen kartu hadir sempro/sidang
6) Jalur 1-BF = 1, 2, 3, 4, 5, 6, 12, 7
   Mahasiswa mengisi semua formulir proposal skripsi dan sebelumnya pernah meng-*upload* dokumen sertifikat pkkmb
7) Jalur 1-BG = 1, 7
   Mahasiswa mengisi formulir proposal skripsi lebih dari satu kali atau dokumennya sedang diperiksa oleh admin

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

Berikut adalah rancangan pengujian fungsi *submitProposalTA()*:

TABEL III
NILAI RANCANGAN PENGUJIAN FUNGSI SUBMITPROPOSALTA

No	Input	Perkiraan
1-BA	Mengisi semua *form* proposal skripsi dan sebelumnya belum pernah meng-upload semua dokumen	Berhasil menyimpan data proposal skripsi
1-BB	Mengisi semua *form* proposal skripsi kecuali dokumen pkkmb dan sebe-lumnya belum pernah meng-upload semua dokumen	Berhasil menyimpan data proposal skripsi
1-BC	Mengisi semua *form* proposal skripsi dan sebelumnya pernah meng-upload dokumen proposal	Berhasil menyimpan data proposal skripsi
1-BD	Mengisi semua *form* proposal skripsi dan sebelumnya pernah meng-upload dokumen cek hasil plagiat	Berhasil menyimpan data proposal skripsi
1-BE	Mengisi semua *form* proposal skripsi dan sebelumnya pernah meng-upload dokumen kartu hadir sempro/sidang	Berhasil menyimpan data proposal skripsi
1-BF	Mengisi semua *form* proposal skripsi dan sebelumnya pernah meng-upload dokumen sertifikat pkkmb	Berhasil menyimpan data proposal skripsi
1-BG	Mahasiswa mengirim *form* proposal skripsi lebih dari satu kali atau dokumennya sedang diperiksa oleh admin	Gagal menyimpan data proposal skripsi

Setelah merancang pengujian fungsi, penulis membuat *code* pengujian untuk setiap jalur. Pengujian fungsi *submitProposalTA()* dilakukan menggunakan akun yang berperan sebagai mahasiswa. Berikut potongan *code* pengujian fungsi *submitProposalTA()*:

Pengujian submitProposalTA()

```
        it('berhasil submit saat mengisi form proposal, belum pernah submit semua dokumen', function(){
            $documentTA = DocumentTA::factory()->create();
            $userModel = new User();
            $user = $userModel->where('role', 'mahasiswa')->where('id', 10)->first();
            $submitProposalTARequest = new SubmitProposalTARequest();
            $submitProposalTARequest->judul_skripsi = 'Rancang Bangun Sistem Informasi';
            $submitProposalTARequest->proposal_skripsi = UploadedFile::fake()-
        >create('proposal_skripsi.pdf', 900);
            $submitProposalTARequest->kartu_hadir = UploadedFile::fake()->create('kartu_hadir.pdf',
        900);
            $submitProposalTARequest->plagiat = UploadedFile::fake()->create('plagiat.pdf', 900);
            $submitProposalTARequest->sertifikat_pkkmb = UploadedFile::fake()-
        >create('sertifikat_pkkmb.pdf', 900);
            $documentTAController = new DocumentTAController();
            $this->actingAs($user);
            $documentTAController->submitProposalTA($documentTA, $submitProposalTARequest);
            $this->assertEquals('Proposal TA berhasil disubmit.', session('success'));
});
```

B. *Rancangan Unit Test 2-B*

Kelas *PengujiSemproController* memiliki fungsi *submitPenilaianSempro()* yang berfungsi untuk dosen memberi penilaian seminar proposal mahasiswa dengan mengisi formulir penilaian yang tersedia pada halaman penilaian

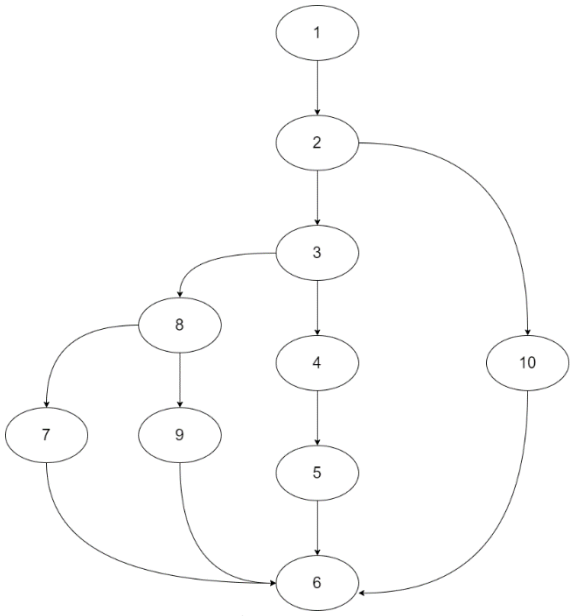
JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

seminar proposal. Berikut adalah *source code* fungsi *submitPenilaianSempro()*:

submitPenilaianSempro()

```
  public    function    submitPenilaianSempro(DocumentTA    $document,    PenilaianSempro    $penilaian,
PenilaianRequest $penilaianRequest, User $user)
      {
          // * 1
          $dataPenilaian = $penilaianRequest->penilaian;
          $jumlah = count($dataPenilaian);
          // * 2
          if  ($penilaian->where('document_id',  $document->id)->where('dosen_id',  $user->id)->latest()-
>first()) {
              // * 10
              return back()->with('error', 'Anda sudah memberikan nilai.');
          }
          // * 3
          for ($i = 1; $i <= $jumlah; $i++) {
              // * 8
              if ($dataPenilaian[$i]['nilai'] > 100) {
                  // * 7
                  return back()->with('error', 'Bapak/Ibu jangan nak ngada-ngada ngasi nilai.');
              }
              if ($dataPenilaian[$i]['nilai'] == 0) {
                  // * 9
                  return back()->with('error', 'Pastikan bapak ibu telah memberikan nilai yang benar.');
              }
          }
          // * 4
          for ($i = 1; $i <= $jumlah; $i++) {
              $dataNilai[] = [
                  'rubrik_id' => intval($dataPenilaian[$i]['rubrik_id']),
                  'document_id' => $document->id,
                  'nilai' => $dataPenilaian[$i]['nilai'],
                  'dosen_id' => auth()->user()->id,
              ];
          }
          // * 5
          foreach ($dataNilai as $dataPenilaian) {
              $penilaian->create($dataPenilaian);
          }
          // * 6
          return back()->with('success', 'Nilai Seminar Proposal ' . $document->user->name . ' berhasil
disimpan permanen.');
      }
```

Pengubahan kode sumber (*source code*) menjadi *flowgraph* pada fungsi *submitPenilaianSempro()* untuk menentukan jalur yang dapat dilalui dapat diamati pada gambar 2.



Gambar 2 Flowgraph fungsi submitPenilaianSempro

Berdasarkan *flowgraph* dari fungsi *submitPenilaianSempro()* di atas terdapat 10 *node* (N) dan 12 *edge* (E). Lalu menghitung nilai *cyclomatic complexity* menggunakan rumus (1) didapatkan nilai V(G) = E – N + 2 = 12 –

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

10 + 2 = 4. Maka dapat ditentukan fungsi *submitLaporan* memiliki 4 jalur. Adapun kemungkinan jalur yang akan diuji antara lain:

1) Jalur 2-BA = 1, 2, 3, 4, 5
   Dosen mengisi dengan lengkap formulir penilaian seminar proposal skripsi dengan masing-masing rentang nilai 1 sampai dengan 100
2) Jalur 2-BB = 1, 2, 3, 6, 8, 5
   Dosen mengisi salah satu nilai dari formulir penilaian seminar proposal skripsi lebih dari angka 100
3) Jalur 2-BC = 1, 2, 4, 6, 7, 5
   Dosen mengisi salah satu nilai dari formulir penilaian seminar proposal skripsi sama dengan angka 0
4) Jalur 2-BD = 1, 2, 9, 5
   Dosen sebelumnya sudah pernah mengisi formulir penilaian seminar proposal skripsi yang sama

Berikut adalah rancangan pengujian fungsi *submitPenilaianSempro()*:

TABEL IV
NILAI RANCANGAN PENGUJIAN FUNGSI SUBMITPENILAIANSEMPRO

No	Input	Perkiraan
2-BA	Mengisi form penilaian seminar proposal skripsi dengan rentang nilai 1 sampai dengan 100	Berhasil menyimpan penilaian seminar proposal skripsi
2-BB	Mengisi form penilaian seminar proposal skripsi dengan rentang nilai lebih dari 100	Gagal menyimpan penilaian seminar proposal skripsi
2-BC	Mengisi form penilaian seminar proposal skripsi dengan nilai sama dengan 0	Gagal menyimpan penilaian seminar proposal skripsi
2-BD	Mengisi form penilaian seminar proposal skripsi yang sebelumnya sudah dinilai	Gagal menyimpan penilaian seminar proposal skripsi

Setelah merancang pengujian fungsi, penulis membuat *code* pengujian untuk setiap jalur. Pengujian fungsi *submitPenilaianSempro()* dilakukan menggunakan akun yang berperan sebagai dosen. Berikut potongan *code* pengujian fungsi *submitPenilaianSempro()*:

Pengujian submitPenilaianSempro()

```
  it('Berhasil submit penilaian seminar proposal dengan rentang 1 sampai 100', function(){
      $document = DocumentTA::factory()->create();
      $userModel = new User();
      $user = $userModel->where('role', 'dosen')->first();
      $penilaian = PenilaianSempro::factory()->create([
          'document_id' => $document->id,
      ]);
      $listRubrik = RubrikTA::get();
      $jumlahRubrik = count($listRubrik);
      for ($i = 1; $i <= $jumlahRubrik; $i++) {
       $arrayPenilaian[$i] = array(
       'rubrik_id' => $i,
       'nilai' => 100,
       'document_id' => $document->id
          );
      }
      $penilaianRequest = new PenilaianRequest();
      $penilaianRequest->merge(['penilaian' => $arrayPenilaian]);
      $pengujiSemproController = new PengujiSemproController();
      $this->actingAs($user);
      $pengujiSemproController->submitPenilaianSempro($document, $penilaian, $penilaianRequest, $user);
      $this->assertEquals('Nilai Seminar Proposal ' . $document->user->name . ' berhasil disimpan
permanen.', session('success'));
});    }
```

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

C. *Rancangan End-To-End Test*

Sebelum membuat kode pengujian *end-to-end*, penting untuk menyusun skenario terlebih dahulu guna memastikan bahwa tidak ada fitur yang terlewatkan. Skenario ini mencakup daftar fitur yang akan diuji dan target hasil yang diharapkan.

TABEL V
SKENARIO PENGUJIAN END-TO-END

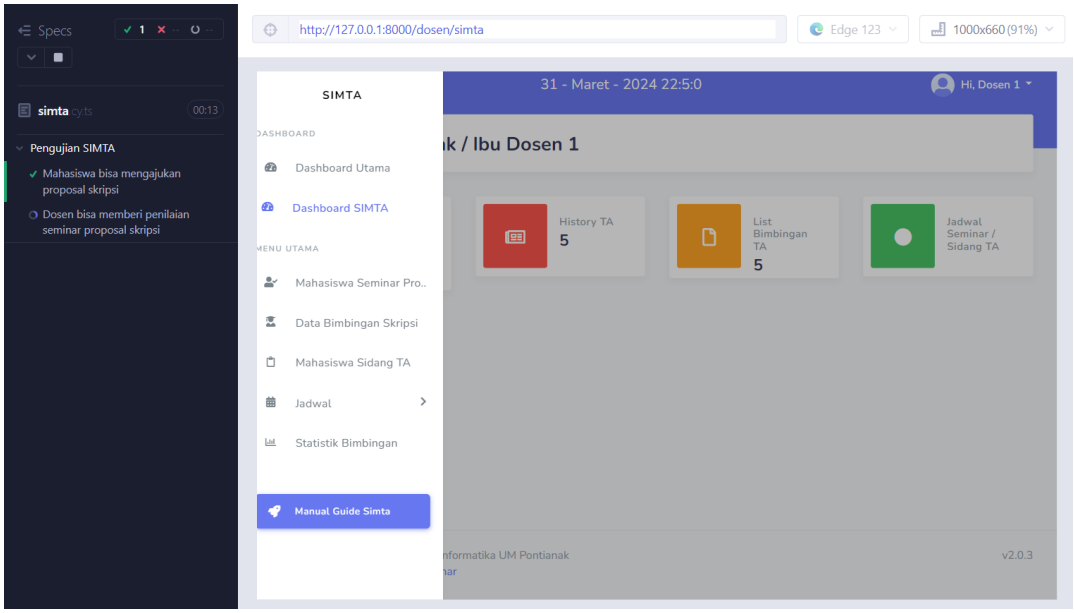
No	Input	Perkiraan
1	Mahasiswa mengajukan proposal skripsi	Berhasil mengajukan proposal skripsi
2	Dosen memberi penilaian seminar pro-posal skripsi	Berhasil memberi penilaian kerja praktik

Setelah merancang skenario pengujian, dilakukan pembuatan kode *Cypress* untuk pengujian *end-to-end*. Berikut potongan *code* pengujian *end-to-end*:

Pengujian End-To-End

```
describe("Pengujian SIKARTIK", () => {
    it("Mahasiswa bisa mengajukan proposal skripsi", () => {
        cy.visit("/login");
        cy.get("#email").type("zayn@ifump.net");
        cy.get("#password").type("password");
        cy.get(".btn").click();
        cy.get(".form-inline > .navbar-nav > li > .nav-link").click();
        cy.get(":nth-child(4) > .nav-link").click();
        cy.get(".form-inline > .navbar-nav > li > .nav-link").click();
});
```

Kemudian kode pengujian dieksekusi untuk melakukan pengujian *end-to-end* menggunakan *Cypress*.



Gambar 3 Eksekusi *code* pengujian di *Cypress*

D. *Hasil Pengujian*

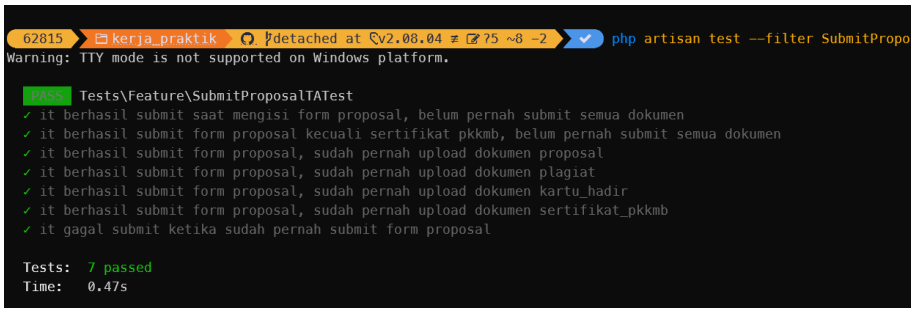
Dalam penelitian ini, *unit testing* dilakukan dengan menggunakan *framework Pest* dan metode pengujian *white box* dengan teknik *basis path*. Langkahnya melibatkan pembentukan *flowgraph* yang terdiri dari *node* dan *edge*, diisi dengan nomor pernyataan untuk menentukan jalur yang akan diuji.

Setelah dilakukan *unit testing*, penulis melanjutkan pengujian dengan *end-to-end testing* menggunakan *framework Cypress*. Pengujian ini bertujuan untuk memastikan bahwa semua pernyataan pada kode sumber yang diuji bebas dari kesalahan logika program.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

1) Hasil *Unit Test* 1-B

Gambar 4 menunjukkan tampilan pada terminal setelah kode pengujian fungsi *submitProposalTA()* dieksekusi. Didapatkan hasil pengujian bahwa ketujuh jalur fungsi telah berjalan sesuai dengan perkiraan dengan keterangan 7 *passed* yang berarti lolos pengujian dengan durasi waktu pengujian 0.47 detik.



Gambar 4 Hasil Pengujian Fungsi submitProposalTA()

Tabel 6 menyajikan detail pengujian fungsi *submitProposalTA()* dari *output* pengujian serta keterangan hasil pengujiannya.

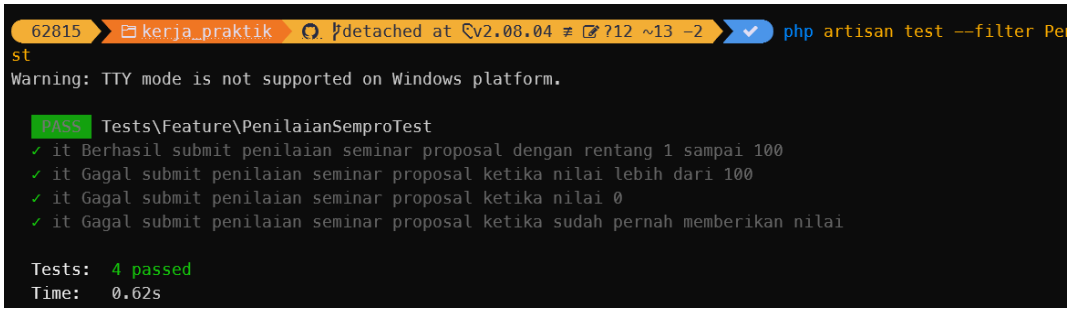
TABEL VI
HASIL PENGUJIAN FUNGSI SUBMITPROPOSALTA

Jalur	Output Pengujian	Keterangan
1-BA	Berhasil menyimpan data proposal skripsi	Lolos
1-BB	Berhasil menyimpan data proposal skripsi	Lolos
1-BC	Berhasil menyimpan data proposal skripsi	Lolos
1-BD	Berhasil menyimpan data proposal skripsi	Lolos
1-BE	Berhasil menyimpan data proposal skripsi	Lolos
1-BF	Berhasil menyimpan data proposal skripsi	Lolos
1-BG	Gagal menyimpan data proposal skripsi	Lolos

Tabel 6 menunjukkan bahwa enam jalur pengujian (1-BA hingga 1-BF) berhasil menyimpan data proposal skripsi dengan keterangan "Lolos", sementara satu jalur pengujian (1-BG) dirancang untuk gagal menyimpan data proposal skripsi dan juga dinyatakan "Lolos". Hal ini menunjukkan bahwa fungsi tersebut beroperasi sesuai dengan yang diharapkan dalam berbagai skenario baik yang berhasil maupun yang gagal, memastikan keandalan sistem dalam menangani *input* yang berbeda.

2) Hasil *Unit Test* 2-B

Gambar 5 menunjukkan tampilan pada terminal setelah kode pengujian fungsi *submitPenilaianSempro()* dieksekusi. Didapatkan hasil pengujian bahwa keempat jalur fungsi telah berjalan sesuai dengan perkiraan dengan keterangan 4 *passed* yang berarti lolos pengujian dengan durasi waktu pengujian 0.62 detik.



Gambar 5 Hasil Pengujian Fungsi submitPenilaianSempro()

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 4, Desember 2024, Pp. 2208-2219

Tabel 7 menyajikan detail pengujian fungsi *submitPenilaianSempro()* dari *output* pengujian serta keterangan hasil pengujiannya.

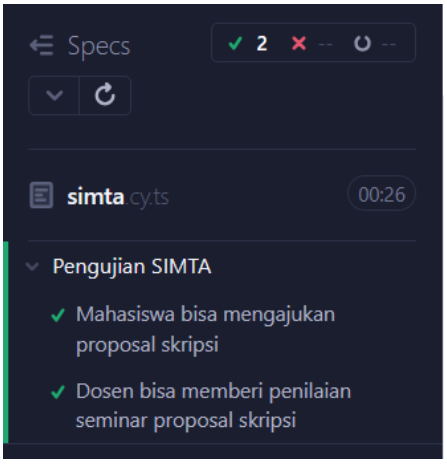
TABEL VII
HASIL PENGUJIAN FUNGSI SUBMITPENILAIANSEMPRO

Jalur	Output Pengujian	Keterangan
2-BA	Berhasil menyimpan penilaian seminar pro-posal skripsi	Lolos
2-BB	Gagal menyimpan penilaian seminar pro-posal skripsi	Lolos
2-BC	Gagal menyimpan penilaian seminar pro-posal skripsi	Lolos
2-BD	Gagal menyimpan penilaian seminar pro-posal skripsi	Lolos

Hasil unit testing untuk fungsi *submitProposalTA()* pada aplikasi SIMATIK menunjukkan bahwa satu jalur pengujian (2-BA) berhasil menyimpan data penilaian dengan keterangan "Lolos", sedangkan tiga jalur pengujian lainnya (2-BB hingga 2-BD) dirancang untuk gagal menyimpan data penilaian dan semuanya juga dinyatakan "Lolos". Ini menunjukkan bahwa fungsi tersebut bekerja dengan baik dalam menangani berbagai skenario penyimpanan, baik yang sukses maupun yang gagal.

3) Hasil *End-To-End Test*

Ada 2 skenario pengujian yang dipilih sebagai subjek pengujian *end-to-end*. Setelah merancang skenario pengujian, penulis membuat kode pengujian *end-to-end* untuk menyimulasikan interaksi pengguna akhir aplikasi yang akan dieksekusi menggunakan *Cypress*. Di gambar 6 dapat dilihat bahwa pengujian 2 skenario lolos pengujian dengan durasi pengujian 26 detik.



Gambar 6 Hasil Pengujian End-To-End

Tabel 8 berikut menyajikan hasil pengujian *end-to-end* pada aplikasi SIMATIK.

TABEL VIII
HASIL PENGUJIAN END-TO-END

No.	Skenario	Perkiraan Output	Hasil Pengujian
1	Mahasiswa mengajukan proposal skripsi	Berhasil mengajukan proposal skripsi	Lolos
2	Dosen memberi penilaian seminar pro-posal skripsi	Berhasil memberi penilaian kerja praktik	Lolos

Tabel 8 menunjukkan bahwa pada skenario pertama, mahasiswa berhasil mengajukan proposal skripsi dengan keterangan "Lolos", dan pada skenario kedua, dosen berhasil memberikan penilaian seminar proposal

skripsi juga dengan keterangan "Lolos". Ini mengindikasikan bahwa aplikasi berfungsi sesuai harapan dalam kedua skenario kunci tersebut, memastikan kelancaran proses pengajuan dan penilaian proposal skripsi.

Keterbatasan dari pengujian aplikasi yang dilakukan adalah terdapat beberapa skenario atau kondisi tertentu yang belum teruji, yaitu skenario mahasiswa mengirim berkas syarat sidang skripsi, dosen memberi penilaian sidang skripsi, dan dosen memberi keputusan sidang skripsi. Untuk mengatasi keterbatasan ini di masa depan, rencana yang perlu dilakukan adalah menambah cakupan pengujian dengan memasukkan skenario-skenario tersebut dalam daftar pengujian, mengembangkan *code* pengujian yang komprehensif untuk setiap kondisi yang belum teruji, serta melakukan pengujian secara berkala untuk memastikan semua fungsi aplikasi berjalan dengan baik dan sesuai dengan kebutuhan pengguna akhir.

IV. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa unit testing pada aplikasi SIMATIK berhasil dilakukan menggunakan *framework Pest*, dengan metode pengujian unit yang diterapkan melalui teknik *basis path* untuk mengevaluasi kompleksitas alur dan menentukan jumlah jalur dari setiap fungsi yang diuji. Selain itu, *end-to-end testing* pada aplikasi SIMATIK juga berhasil dilakukan menggunakan *framework Cypress*, yang melibatkan beberapa langkah seperti penyusunan skenario pengujian, pembuatan kode *Cypress*, dan eksekusi kode pengujian. Meskipun penulis telah berhasil melakukan *unit testing* dan *end-to-end testing* pada aplikasi, terdapat beberapa saran untuk perbaikan dan pengembangan lebih lanjut, yaitu memperluas cakupan unit testing pada fungsi-fungsi lainnya dalam aplikasi SIMATIK untuk memastikan seluruh kode telah diuji dengan baik dan meminimalkan potensi *bug* atau kesalahan, serta menambahkan lebih banyak kasus uji atau skenario pengujian pada aplikasi SIMATIK untuk meningkatkan ketahanan aplikasi terhadap berbagai skenario.

DAFTAR PUSTAKA

[1] D. L. Kaligis dan R. R. Fatri, "PENGEMBANGAN TAMPILAN ANTARMUKA APLIKASI SURVEI BERBASIS WEB DENGAN METODE USER CENTERED DESIGN," *JUST IT J. Sist. Inf. Teknol. Inf. Dan Komput.*, vol. 10, no. 2, hlm. 106, Jun 2020, doi: 10.24853/justit.10.2.106-114.
[2] D. Graham, R. Black, dan E. van Veenendaal, *Foundations of software testing: ISTQB certification*, Fourth edition., Updated for ISTQB Foundation Syllabus 2018. London: Cengage Learning, 2020.
[3] W. Wibisono dan F. Baskoro, "PENGUJIAN PERANGKAT LUNAK DENGAN MENGGUNAKAN MODEL BEHAVIOUR UML," *JUTI J. Ilm. Teknol. Inf.*, vol. 1, no. 1, hlm. 43, Jul 2002, doi: 10.12962/j24068535.v1i1.a95.
[4] D. B. Muslimin, D. Kusmanto, K. F. Amilia, M. S. Ariffin, S. Mardiana, dan Y. Yulianti, "Pengujian Black Box pada Aplikasi Sistem Informasi Akademik Menggunakan Teknik Equivalence Partitioning," *J. Inform. Univ. Pamulang*, vol. 5, no. 1, hlm. 19, Mar 2020, doi: 10.32493/informatika.v5i1.3778.
[5] H. Hendri, J. W. Hasiholan Manurung, R. A. Ferian, W. F. Hanaatmoko, dan Y. Yulianti, "Pengujian Black Box pada Aplikasi Sistem Informasi Pengelolaan Masjid Menggunakan Teknik Equivalence Partitions," *J. Teknol. Sist. Inf. Dan Apl.*, vol. 3, no. 2, hlm. 107, Apr 2020, doi: 10.32493/jtsi.v3i2.4694.
[6] D. K. Pallas, "BLACK BOX TESTING APLIKASI POINT OF SALES POST," *Kurawal - J. Teknol. Inf. Dan Ind.*, vol. 4, no. 1, hlm. 1–16, Mar 2021, doi: 10.33479/kurawal.v4i1.399.
[7] D. Sakethi, D. Kurniawan, dan H. Tantriawan, "Pengujian dan Perawatan Sistem Informasi Menggunakan White Box Testing," 2016.
[8] S. Pare, "Desain Dan Implementasi E-Commerce Pada Toko As 88 Celluler Merauke," *J. Ilm. Mustek Anim*, 2017.
[9] Program Studi Informatika, Universitas Nasional *dkk.*, "Pengembangan Aplikasi Sistem Informasi Smart Register Online Berbasis Android Menggunakan Algoritma BruteForce," *Edumatic J. Pendidik. Inform.*, vol. 4, no. 1, hlm. 47–56, Jun 2020, doi: 10.29408/edumatic.v4i1.2106.
[10] M. Hafid, "PHP Unit VS Pest PHP," Medium. Diakses: 23 November 2023. [Daring]. Tersedia pada: https://muhammadhafid385.medium.com/php-unit-vs-pest-php-c8e33e03489e
[11] L. Badal dan D. Grunler, "Automated End-to-End Testing: Useful Practice or Frustrating Timesink?" 2021. Diakses: 2 Oktober 2023. [Daring]. Tersedia pada: https://raw.githubusercontent.com/KTH/devops-course/2021/contributions/essay/badal-grunler/E2E_Testing_Essay.pdf
[12] "Cypress Adalah: Alat Pengujian Aplikasi Web Masa Kini | APPKEY," Cypress Adalah: Alat Pengujian Aplikasi Web Masa Kini. Diakses: 23 November 2023. [Daring]. Tersedia pada: https://appkey.id/pembuatan-website/maintenance/cypress-adalah/
[13] G. W. R. B. Putro, "Perbandingan Performa Selenium Dan Cypress Dalam Pengujian Website Bimbingan Mahasiswa Universitas Atma Jaya Yogyakarta," 2021.
[14] T. Taky dan M. Tasnim, "Automated Testing With Cypress".
[15] B. Prakoso, "Pemanfaatan Cypress Untuk Pengujian End-to-End (Studi Kasus: Pengembangan Aplikasi Indicar)," 2023.
[16] D. K. P. Rahayu, "Unit Testing Pada Aplikasi Web Mobile (Studi Kasus Bisnis Jasa Laundry)," 2020.
[17] Handy dan Susilo J., "Aplikasi Pengujian White-Box Ibii Online Judge," *J. Inform. Dan Bisnis*, 2014.
[18] C. T. Pratala, E. M. Asyer, I. Prayudi, dan A. Saifudin, "Pengujian White Box pada Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path," *J. Inform. Univ. Pamulang*, vol. 5, no. 2, hlm. 111, Jun 2020, doi: 10.32493/informatika.v5i2.4713.
[19] "Software Testing: Perhitungan Cyclomatic Complexity," School of Computer Science. Diakses: 7 Desember 2023. [Daring]. Tersedia pada: https://socs.binus.ac.id/2016/12/29/software-testing-perhitungan-cyclomatic-complexity/
[20] S. McConnell, *Code complete*, Second edition. Redmond, Washington: Microsoft Press, 2004.
[21] J. L. Min, A. Istiqomah, dan A. Rahmani, "EVALUASI PENGGUNAAN MANUAL DAN AUTOMATED SOFTWARE TESTING PADA PELAKSANAAN END-TO-END TESTING," *JTT J. Teknol. Terap.*, vol. 6, no. 1, hlm. 18, 2020, doi: 10.31884/jtt.v6i1.256.