SHORTYURL: DASBOR MANAJEMEN *URL SHORTENER* TERINTEGRASI DENGAN SSO UII

Disusun Oleh:

Nama : Ade Bagas Permata Putra
NIM : 15523007

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

SHORTYURL: DASBOR MANAJEMEN *URL SHORTENER* TERINTEGRASI DENGAN SSO UII

TUGAS AKHIR

Disusun Oleh:

Nama : Ade Bagas Permata Putra
NIM : 15523007

Yogyakarta, 20 Februari 2022

Pembimbing,

(Dr. Mukhammad A Setiawan, S.T., M.Sc.)

HALAMAN PENGESAHAN DOSEN PENGUJI

SHORTYURL: DASBOR MANAJEMEN *URL SHORTENER* TERINTEGRASI DENGAN SSO UII

TUGAS AKHIR

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,17 Februari 2022

Tim penguji

Dr. Mukhammad A Setiawan, S.T., M.Sc.

Anggota 1

Kholid Haryono, S.T., M.Kom.

Anggota 2

Hari Setiaji, S.Kom., M.Eng.

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 : Ade Bagas Permata Putra

NIM : 15523007

Tugas akhir dengan judul:

SHORTYURL: DASBOR MANAJEMEN *URL SHORTENER* TERINTEGRASI DENGAN SSO UII

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 apa pun.

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

Yogyakarta, 01 Januari 2022

(Ade Bagas Permata Putra)

HALAMAN PERSEMBAHAN

Penulis menyatakan bahwa skripsi selaku tugas akhir dalam perkuliahan Universitas Islam Indonesia dipersembahkan untuk penulis di masa depan serta generasi muda yang memiliki rasa ingin tahu terhadap teknologi yang terkait di dalam laporan ini.

HALAMAN MOTO

Di mana ada kesulitan, di situ ada kemudahan.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Puji syukur ke hadirat Allah SWT. Atas segala rahmat dan hidayahnya sehingga penulis dapat menyelesaikan laporan dengan judul SHORTYURL: DASBOR MANAJEMEN URL SHORTENER TERINTEGRASI DENGAN SSO UII. Tak lupa shalawat dan salam kepada junjungan kita Nabi Muhammad SAW.

Skripsi merupakan tugas atau salah satu kewajiban yang harus diselesaikan untuk mendapatkan gelar sarjana, penulis sangat bersyukur akan pelajaran yang didapatkan ketika menjalani kegiatan tersebut, yang salah satunya menghasilkan sebuah laporan ini. Penulis juga mengucapkan terima kasih kepada:

1. Seluruh keluarga penulis terhadap doa yang dipanjatkan.
2. Imam Muslim Tri Pamuji S.Kom. yang bersedia menyediakan fasilitas untuk membantu pengerjaan laporan ini hingga selesai, serta tempat tinggal.
3. Dhika Bayu Kusuma terhadap fasilitas serta tempat tinggal maupun urusan makan.
4. Bang Ryan & Gilang, Kiki(Riski Wahyu), Farhan, Wahyu selaku sahabat yang telah meluangkan waktunya
5. Septia Rani, S.T., M.CS., selaku dosen pembimbing akademik yang selalu sabar serta bersedia ditanya dan juga membantu untuk menghubungi pihak dosen yang bersangkutan jika ada penulis ada kendala
6. Bapak Dr. Mukhammad Andri Setiawan, S.T., M.Sc., selaku dosen pembimbing Tugas Akhir yang tidak memberatkan mahasiswa.
7. Seluruh pihak yang tidak bisa disebutkan semua satu persatu.

Yogyakarta, 01 Januari 2022

(Ade Bagas Permata Putra)

SARI

Belum adanya layanan mempersingkat yang dimiliki UII membuat penulis mengembangkan layanan ini, terintegrasi dengan SSO UII yang menggunakan Shibboleth dengan SAML. Dengan IdP yang telah dibuat oleh UII, ShortyURL selaku SP yang harapannya bisa menjadi pertimbangan untuk kedepannya dan pastinya dapat menekan biaya pengeluaran.

Kata kunci: shibboleth, saml, sso, Laravel

GLOSARIUM

ACS	Assertion Customer Service.
Assertion	Tanggapan dari IdP oleh SP dalam bentuk dokumen XML yang strukturnya didefinisikan oleh standar SAML.
IAM	Framework proses bisnis.
IdM	Sistem pengelolaan identitas.
IdP	Tempat autentifikasi yang menerima maupun memberi tanggapan kepada SP.
Metadata	Konfigurasi pada SP dan IdP.
SAML	Security Assertion Markup Language.
SAML Request	Dokumen XML yang dikirim oleh SP untuk IdP.
SAML Response	Tanggapan IdP dalam bentuk dokumen XML untuk SP.
SP	Layanan yang nantinya berhubungan dengan IdP.
SSO	Metode login yang terpusat.
XML	Extensible Markup Language.

DAFTAR ISI

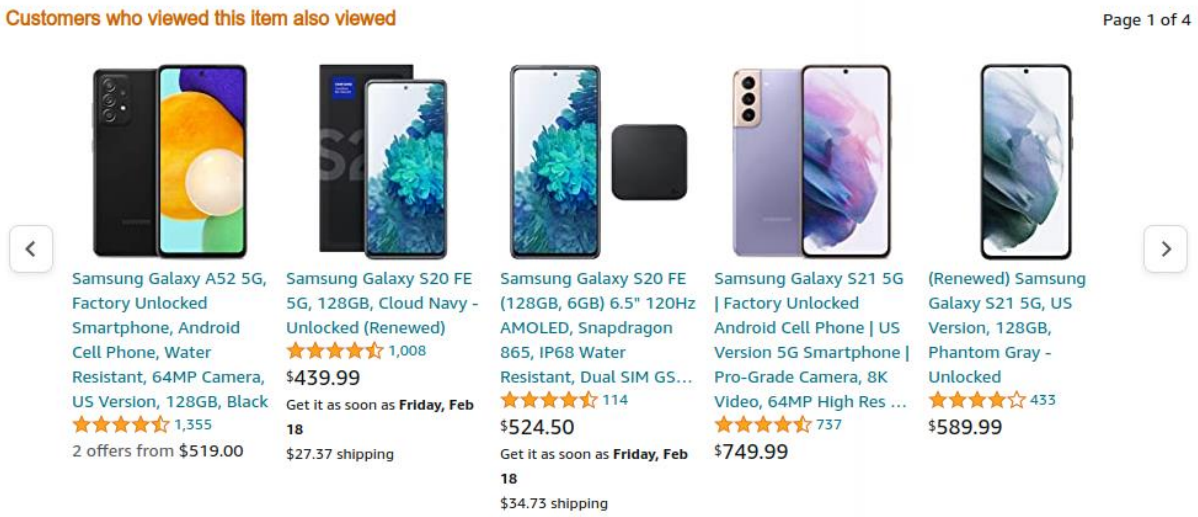
DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, *brand* melambangkan pengalaman pelanggan yang menyeluruh diberikan oleh perusahaan, tidak seperti pemikiran tradisional yang di mana *brand* itu adalah seperangkat gambar, sering kali sebuah nama, logo, dan *tagline* yang membedakan produk atau layanan yang diberikan perusahaan dari pesaingnya menurut (Kotler et al., 2017). Sejak tahun 1980, *brand positioning* telah diakui sebagai cara untuk merebut hati pelanggan (Kotler et al., 2017). Menurut (Kotler et al., 2017) *brand* harus jelas dan memiliki konsisten seperti diferensiasi yang otentik untuk membangun equitas yang kuat, serta untuk memahami pelanggan sebuah *brand* juga dimungkinkan untuk menggunakan analitis *big-data*. Saat ini sangat penting untuk mengingat bahwa *big-data* juga mungkin menghasilkan dampak positif (Pence, 2014) terhadap *brand* seperti pada bagian di dalam laman Amazon yang terdapat saran produk untuk pengunjung digambarkan pada Gambar 1.1 berikut:



Gambar 1.1 Bagian saran pada laman Amazon

Proses analitis telah tumbuh ke tingkatan selanjutnya di mana kita bisa memahami tingkah laku pelanggan secara *real time*, tidak hanya termasuk statistik penggunaan saja melainkan demografi dan bahkan keinginan mereka (Kingsnorth, 2016), agar sebuat brand

dapat memberikan saran dengan tepat dan mendapatkan keuntungan dari kenyamanan pengguna yaitu memperkuat *brand*. Semua itu dapat dilakukan karena adanya data yang tersedia, oleh karena itu data di sini memiliki peran yang sangat penting untuk analisis maupun analitis.

Salah satu brand yang terkenal di lingkup universitas adalah UII (Universitas Islam Indonesia) yang merupakan salah satu universitas tertua di Indonesia yang memiliki berbagai macam layanan yang tersedia untuk mahasiswa maupun dosen, salah satunya adalah sistem berbasis *website* bernama UII Gateway. Sistem ini dapat menjadi jembatan antar mahasiswa untuk melakukan pengisian mata kuliah di awal semester, bisa juga untuk memeriksa status kemahasiswaan, nilai mata kuliah tiap semester maupun keseluruhan, dan bisa juga untuk memeriksa nomor tagihan SPP. Gerbang utama yang dulunya menggunakan OpenSAML ini sekarang telah mengadopsi sistem SSO yang didukung oleh Shibboleth IdP serta SP, dengan implementasi teknologi yang dikembangkan oleh organisasi nirlaba bernama OASIS yaitu SAML (Lockhart et al., 2012) berbasis XML.

Sistem SSO yang telah ada sangat bergantung pada alamat email untuk menghubungkan akun dengan identitas asli, dan banyak penyedia layanan mengadopsi alamat email sebagai identifikasi akun utama (Liu et al., 2021), salah satu layanan dari *brand* raksasa yaitu Google, memiliki layanan Google Classroom yang menghasilkan kepuasan pada pelajar karena terbukti efektif dalam menjadi alat pembelajaran aktif (Shaharanee et al., 2016), layanan dengan sistem SSO ini digunakan oleh dosen UII untuk mengorganisir kelas yang diampu, seperti membagikan informasi pertemuan kuliah daring beserta tautan Zoom atau Google Meet sebagai mediasi. Saat ini, Zoom adalah aplikasi konferensi video paling popular (Singh & Awasthi, 2020), tautan Zoom yang panjang memiliki kemungkinan kecil untuk diingat sehingga layanan mempersingkat tautan seperti Bit.ly yang merupakan layanan pemendekan URL menjadi sangat digemari. Gagasan dibalik layanan pemendekan URL adalah membantu untuk berbagi URL dengan singkatan yang sepadan (Antoniades et al., 2011). Instagram juga menerapkan pola yang sama terhadap tautan pada *posting* setiap *user*. Belum adanya layanan mempersingkat tautan yang dimiliki UII merupakan alasan untuk menjadi pelanggan Bit.ly dengan cara membayar sejumlah uang sebelum menikmati fitur *enterprise*. Bit.ly adalah salah satu layanan pemendekan URL paling popular (Choi et al., 2018) dengan biaya langganan *enterprise* sekitar $10.000 (Muktar, 2020).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dibahas sebelumnya, dapat disimpulkan bahwa UII belum memiliki layanan pendukung yang bertugas untuk mempersingkat tautan sebagai sarana *branding* dalam bentuk website, oleh sebab itu beberapa rumusan masalah yang tercipta adalah sebagai berikut:

a. Bagaimana cara menyediakan data untuk kepentingan analisis atau analitis sebagai pendukung kebutuhan *branding* dalam bentuk *dashboard* berbasis *website* ?
b. Bagaimana cara mempersingkat tautan dengan pembuatan aplikasi ShortyURL berbasis *website* ?
c. Bagaimana cara integrasi ShortyURL dengan SSO UII ?

1.3 Batasan Masalah

Terdapat batasan yang dimiliki oleh aplikasi ShortyURL ini, adapun beberapa diantaranya adalah sebagai berikut:

a. Aplikasi ini hanya memberi izin kepada identitas yang terdaftar pada IdP UII untuk melakukan pemendekan tautan.
b. Aplikasi ini hanya melayani satu IdP.

1.4 Tujuan Penelitian

Mengetahui bahwa ShortyURL akan menjadi layanan pada UII merupakan salah satu tujuan, serta beberapa tujuan mengenai pembuatan layanan ShortyURL di antara lain adalah sebagai berikut:

a. Memperkuat *brand* sekaligus instansi dengan cara melengkapi layanan yang dibutuhkan.
b. Membangun ShortyURL untuk melengkapi layanan pendukung pada UII.
c. Membangun layanan dengan standar *enterprise*.
d. Mengurangi pemakaian layanan pihak ketiga.
e. Memanfaatkan layanan SSO UII yang tersedia.

1.5 Manfaat Penelitian

Dengan adanya ShortyURL yang menjadi layanan pada UII terdapat lebih dari satu manfaat di antaranya adalah sebagai berikut:

a. Aplikasi ShortyURL dapat menekan biaya penggunaan terhadap layanan tambahan seperti Bit.ly.

b. Seluruh anggota yang memiliki akun UII dapat menikmati layanan ShortyURL.
c. Meningkatkan layanan untuk anggota yang terkait pada UII.

1.6 Metodologi Penelitian

Tugas akhir ini diselesaikan dengan tahapan yang mirip dengan metodologi *waterfall* di mana pada metode tersebut yang dikerjakan secara runtun, adapun tahapan yang akan dikerjakan nantinya adalah sebagai berikut:

a. Perancangan aplikasi.
b. Implementasi SSO.
c. Implementasi Shortener.
d. Pengujian.

1.7 Sistematika Penulisan

Dengan laporan yang terbagi menjadi beberapa bab akan mempermudah pengelompokan materi yang ingin dijelaskan, seperti halnya laporan ini yang akan terbagi menjadi 5 BAB, adapun penjelasan untuk masing-masing bab akan dijelaskan selanjutnya.

BAB I Pendahuluan

Pada bagian ini penulis akan menjelaskan latar belakang laporan ini hingga sistematika penulisan

BAB II Landasan Teori

Di sini penulis akan sedikit menjelaskan tentang beberapa bagian yang nantinya akan digunakan untuk mendukung penyelesaian tugas akhir ini seperti apa itu IdP, SP, SAML dan yang lainnya sebagai landasan berpikir.

BAB III Metodologi Penelitian

Penjelasan mendetail juga rinci tentang tahapan pengerjaan akan di uraikan pada bagian ini, juga sesuai dengan metodologi yang digunakan.

Penjelasan metode akan diuraikan lebih rinci dan mendetail pada bagian ini.

BAB IV Hasil dan Pembahasan

Implementasi metodologi yang dibahas sebelum BAB ini akan membuahkan hasil yang nantinya akan ditulis pada bagian ini, serta pembahasan mengenai implementasi tersebut.

BAB V Simpulan dan Saran

Setelah hasil dari implementasi metodologi telah didapatkan, kesimpulan yang ditarik serta saran terhadap kesimpulan itu juga nantinya akan ditulis pada bab ini.

BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Skripsi ini mengembangkan ide dan mengambil inspirasi dari beberapa teknologi gabungan yang telah ditulis pada artikel, buku maupun jurnal dengan media pustaka *online* sebagai sarana penyimpanan, adapun beberapa sumber yang penulis temukan serta rangkum adalah sebagai berikut:

Tabel 2.1 Tinjauan pustaka

No	Pengarang	Judul	Kesimpulan
1	(Graves & Vandenbrink, 2014)	Implementing a Shibboleth SSO Infrastructure	Shibboleth layak memiliki reputasi kompleks, dan menambahkan *two-factor* autentifikasi dan mengamankan sekeliling sistem operasi serta lingkungan web *server* membuatnya lebih kompleks. *Framework* STRIDE membantu dalam menemukan *vulnerable* secara teori pada proyek ini yang mungkin luput dari perhatian.
2	(Manik & Sidharta, 2017)	The impact of academic service quality on student satisfaction	Beberapa faktor yang dapat mempengaruhi kepuasan mahasiswa adalah *influence tangible*, *reliability*, *responsiveness*, dan *empathy* yang apabila ditingkatkan, maka kepuasan mahasiswa juga dapat meningkat.
3	(Morgan et al., 2004)	Federated Security: The Shibboleth Approach	Shibboleth menyediakan solusi efektif untuk

			mengamankan *multi-organizational* website. Sangat disarankan untuk mengadopsi Shibboleth dan terdapat banyak kampus yang telah melakukannya dan *service providers* memiliki implikasi yang menarik bahkan untuk yang tidak membutuhkan diantaranya ada *meeting campus identity management standards*, *privacy control*, dan *attribute-based authorization.*
4	(Simco, 2001)	The Internet 2 Middleware Initiative	Lapisan *middleware* merupakan hal penting dalam distribusi sistem. Fokus dari penelitian serta pengembangan dalam I2MI adalah untuk menyajikan inti dari layanan *middleware*. Inti yang dimaksud mencakup *identification*, *authentication*, *authorization*, *directory*, dan *certification* yang merupakan standar.
5	(Liu et al., 2021)	An Investigation of Identity-Account Inconsistency in Single Sign-On	Menghadirkan ancaman *identity-account* yang tidak konsisten karena email yang dijadikan ID digunakan kembali atau terdapat ID yang sama di dalam sistem SSO melalui beberapa skenario. Tahapan yang dilakukan adalah menguji *feasibility* lalu melihat potensi *thread* dari *vulnerability* terhadap 100 SP populer. Demonstrasi terhadap penggunaan kembali alamat email

			sangat mungkin terjadi dalam kehidupan nyata.
6	(Kukic, 2011)	The Definitive Guide to Single Sign On (SSO)	Tulisan ini menerangkan banyak cara untuk implementasi SSO dan sangat penting untuk mencari solusi yang tepat untuk organisasi yang menggunakannya. Tulisan ini juga membantu pembaca untuk memahami SSO, mengerti keuntungan menggunakan SSO pada organisasi, mempelajari beberapa *identity protocol*, dan yang terakhir adalah panduan implementasi.
7	(Radha & Reddy, 2012)	A Survey on Single Sign-On Techniques	Tidak bisa di bantah bahwa SSO memberikan kemudahan dan keamanan, dengan mengurangi dan menjadikan satu akun untuk setiap layanan, banyaknya *password* dan pusat manajemen terhadap peran untuk menentukan akses kontrol. *End-user* di sini sangat mendapatkan keuntungan. Perlu di ingat bahwa SSO harus diimplementasikan dengan cara yang aman.
8	(Doupé et al., 2011)	Fear the EAR: Discovering and Mitigating Execution After Re-direct Vulnerabilities	Menjelaskan tipe baru dari *vulnerability* yaitu *Execution After Redi-rect* (EAR), dan mengembangkan alat analisa statistik untuk mencari EAR secara efektif. EAR sulit dicari karena menghindari spesifik *logic* dari aplikasi web dan memahami *logic*

			aplikasi harusnya membantu dalam membedakan mana yang rentan dan jinak, hal itu juga yang akan dijadikan fokus pada masa mendatang.

Tinjauan pustaka yang telah disimpulkan menghasilkan pernyataan bahwa Shibboleth hidup lingkungan *enterprise* juga merupakan teknologi yang mengharuskan campur tangan ahli untuk mengoperasikannya meskipun termasuk dalam kategori proyek *open source*,. Teknologi ini akan bagus berjalan dengan semestinya dan bahkan bisa lebih aman lagi apabila diterapkan dengan benar. Tidak ada yang bisa membantah bahwa SSO sangat memberikan kemudahan terhadap proses *autentification.*

2.2 Branding

Brand merupakan kata yang cocok untuk melambangkan suatu induk layaknya *brand* yang sudah terkenal seperti pada Gambar 2.1 berikut:

Gambar 2.1 Contoh brand

Sumber: https://goldlevelprint.com/wp-content/uploads/2019/12/company-logos-1-1.jpg

Menurut (Kotler et al., 2017) selaku penulis buku "Marketing 4.0: Moving From Traditional to Digital" ini menyatakan bahwa terdapat lima kategori atau tahapan pada pelanggan yaitu *aware*, a*ppeal*, *ask*, *act*, dan *advocate*. Seperti yang dilakukan oleh UII di mana sebelum menggunakan layanan seperti UII Gateway, Unisys, Google Classroom dan lain sebagainya, UII menerapkan sistem SSO sebagai gerbang utama yang secara tidak langsung itu meningkatkan *awareness* terhadap layanan yang diberikan kepada mahasiswa. Hal tersebut mempengaruhi *reliability* yang dapat meningkatkan kepuasan mahasiswa (Manik & Sidharta, 2017) serta memperkuat brand. Adapun penjelasan tentang SSO akan dijelaskan pada bagian selanjutnya.

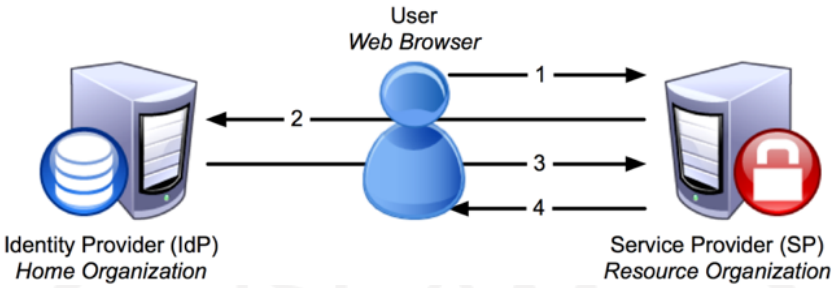
2.3 SSO

Teknologi dengan kepanjangan *Single Sign-On* ini biasa ditemukan di dalam lingkungan perusahaan yang di mana pekerjanya biasa menggunakan berbagai macam aplikasi dan layanan setiap harinya (Kukic, 2011). Teknologi ini membuat pengguna cukup melakukan *log-in* pada satu sistem dan otomatis bisa menggunakan layanan lainnya pada sistem yang berbeda (Kukic, 2011). Untuk menciptakan SSO terdapat teknologi lain yang harus digunakan sebelumnya, diantaranya adalah OpenAthens atau Shibboleth yang merupakan IdM (*Identity Management*) dalam bagian IAM (*Identity and Access Management*). Adapun IdM yang digunakan saat pelaksanaan tugas akhir ini adalah Shibboleth, teknologi tersebut akan dijelaskan pada bagian selanjutnya.

2.4 Shibboleth

Sebelum *Security and Identity Management* dinyatakan menjadi salah satu tantangan kritis yang dihadapi IT pada kampus (Spicer et al., 2004), komunitas Internet2 telah mendirikan I2MI (*Internet2 Middleware Initiative*) dengan tujuan untuk menyajikan inti dari layanan *middleware* yang mencakup *identification, authentication, authorization, directory,* dan *certification*. I2MI memiliki bagian yang bernama MACE (Middleware Architecture Committee for Education) dan di dalamnya memiliki tiga kelompok yang bernama MACE-Dir, MACE-PKI, dan MACE-Shibboleth yang merupakan kelompok terakhir dengan fokus *authentication* dan *authorization* proses untuk halaman web (Simco, 2001). Dengan pembagian kelompok tersebut, terciptalah *Shibboleth System* dan dikenal dengan Shibboleth (Morgan et al., 2004) yang merupakan proyek sistem *login Single Sign-On* dimulai pada tahun 2000 serta merilis proyek pertamanya pada tahun 2003 dengan nama Shibboleth 1.0 oleh Internet2, hingga akhirnya pada tahun 2013 proyek ini dikelola oleh Shibboleth Consortium.

Menurut (Shibboleth Consortium, 2009) interaksi dasar yang terjadi pada Shibboleth dapat digambarkan seperti pada Gambar 2.2 berikut:



Gambar 2.2 Interaksi dasar

Sumber: (Shibboleth Consortium, 2009)

Adapun penjelasan dari tahapan pada Gambar 2.2 yang merupakan interaksi dasar sebuah sistem SSO adalah berikut:

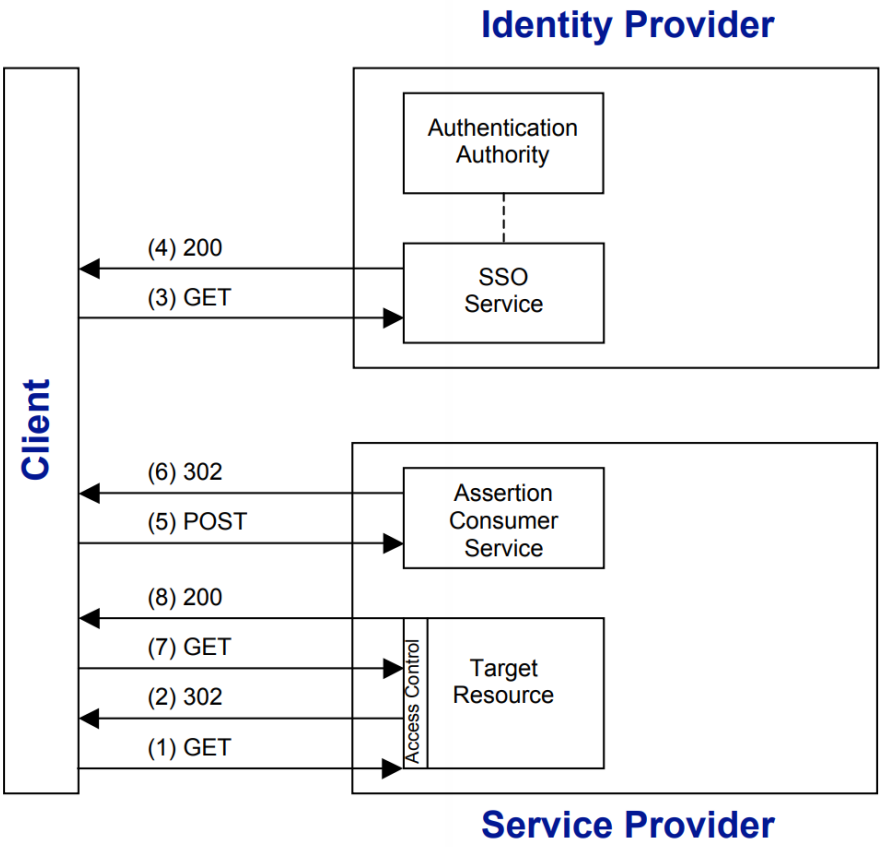
a. SP mendeteksi bahwa pengguna mencoba untuk mengakses konten yang dibatasi pada *resource.*
b. SP membuat permintaan autentikasi dan mengirimkannya bersama pengguna kepada IdP.
c. IdP melakukan autentikasi terhadap pengguna dan mengirim balik autentikasi balasan bersama *user* kepada SP.
d. SP memeriksa balasan dari IdP dan melanjutkan permintaan awal pengguna.

Terdapat dua produk yang digunakan untuk mendukung jalannya SSO yaitu *Shibboleth Identity Provider* (Shibboleth IdP) dan *Shibboleth Service Provider* (Shibboleth SP), adapun tahapan pada yang dilakukan pada Shibboleth menurut (Morgan et al., 2004) adalah sebagai berikut:

a. Pengguna menuju website menggunakan *browser.* Untuk web yang dilindungi membutuhkan informasi tambahan guna memutuskan apakah pengguna memiliki izin.
b. Shibboleth SP mengarahkan *browser* pada halaman navigasi yang disebut *where are you from* (WAYF), halaman ini menyediakan daftar *organizations* yang dapat digunakan.
c. Pengguna memilih *home organization*, dan *browser* diarahkan ke web yang berjalan dengan Shibboleth IdP sesuai dengan pilihan pengguna. Sekarang pengguna dapat melihat halaman *login* yang dikenali dan melakukan proses *login.*

d. Shibboleth IdP mengirim *browser* kembali ke *website* yang dituju oleh pengguna termasuk dengan *security information* yang disebut dengan *assertion*, lalu Shibboleth SP pada *website* yang dituju akan melakukan validasi terhadap *assertion* tersebut.
e. Shibboleth SP menerima *attributes* pengguna dari *home organization* dan menyediakan informasi untuk website tersebut selama website berjalan.

Terdapat beberapa skenario diantaranya menurut (Scavo & Cantor, 2005) alur yang terjadi pada Shibboleth *Browser/POST Profile* dapat digambarkan pada Gambar 2.3 seperti berikut:



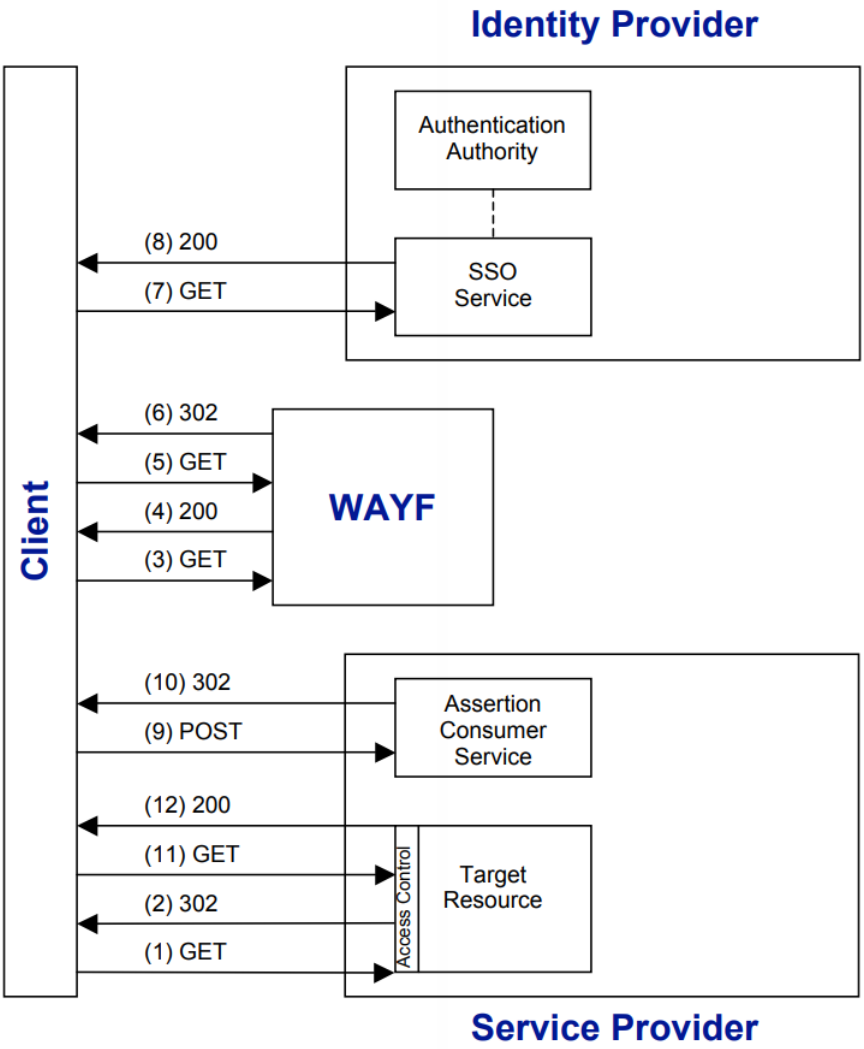
Gambar 2.3 Shibboleth *Browser/POST Profile*

Sumber: (Scavo & Cantor, 2005)

Adapun penjelasan dari seluruh tahapan pada penomoran Gambar 2.3 untuk alur Shibboleth *Browser/POST Profile* akan dijabarkan sebagai berikut:

a. Pengguna meminta *resource* pada SP *https://sp.example.org/myresource* lalu SP melakukan pemeriksaan keamanan. Apabila SP sebelumnya telah membuat *security context*, maka tahapan dua hingga tujuh tidak perlu dilalui kembali.
b. SP mengarahkan pengguna ke layanan SSO pada IdP.
c. Pengguna meminta layanan SSO pada IdP.
d. Layanan SSO membalas dengan dokumen yang mengandung formulir dalam bentuk HTML.
e. Pelanggan melanjutkan *request* POST pada *assertion consumer service* di SP.
f. *Assertion consumer service* membuat *security context* pada SP dan mengalihkan pelanggan ke *resource* tujuan.
g. Pelanggan melakukan *request* pada *resource* tujuan lagi.
h. Sejak *security context* ada maka SP mengembalikan *resource* yang diminta kepada pengguna.

Skenario lainnya menurut (Scavo & Cantor, 2005) yaitu Shibboleth *Browser/POST Profile with WAYF Service* yang alurnya dapat digambarkan seperti pada Gambar 2.4 berikut:



Gambar 2.4 Shibboleth *Browser/POST Profile with WAYF Service*

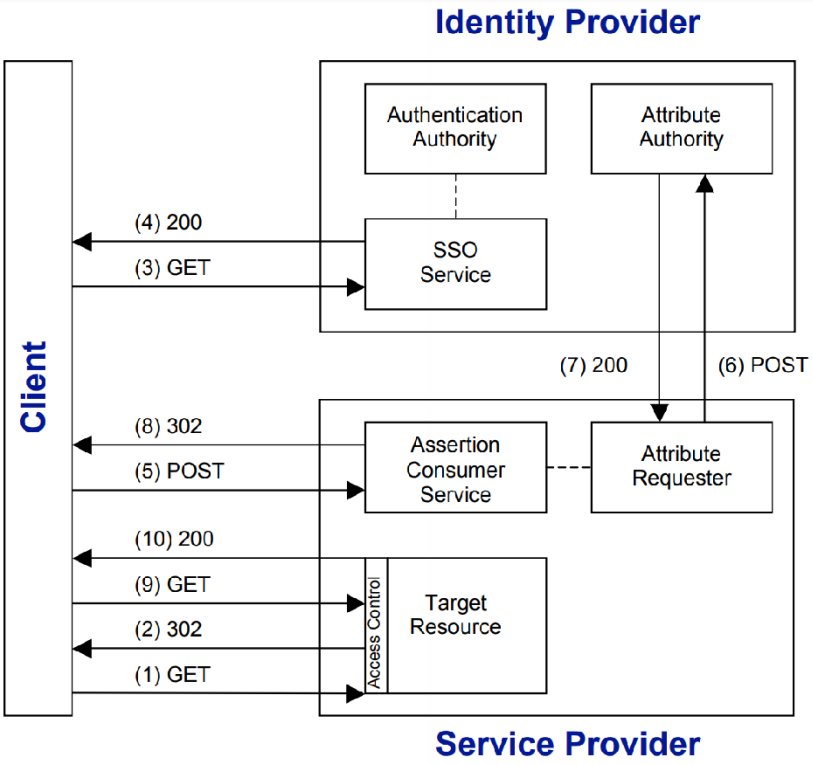
Sumber: (Scavo & Cantor, 2005)

Untuk penjelasan setiap bagian yang diberi nomor pada Gambar 2.4 di atas akan dijelaskan sebagai berikut:

a. Pengguna meminta *resource* pada SP *https://sp.example.org/myresource* lalu SP melakukan pemeriksaan keamanan. Apabila SP sebelumnya telah membuat *security context*, maka tahapan dua hingga sebelas tidak perlu dilalui kembali.
b. SP mengarahkan pelanggan ke WAYF.
c. Pelanggan meminta layanan WAYF.
d. WAYF membalas untuk pelanggan dalam bentuk formulir HTML.
e. Pelanggan memilih IdP dari daftar yang disediakan lalu melakukan *submit* terhadap formulir tersebut.

f. WAYF memperbaharui *cookie* dengan pilihan IdP dari pelanggan dan mengarahkan pelanggan ke layanan SSO.
g. Pelanggan melakukan *request* pada layanan SSO di IdP.
h. Layanan SSO membalas dengan formulir HTML seperti pada tahap 4 di *Browser/ POST Profile.*
i. Pelanggan melanjutkan *request* POST pada *assertion consumer service* di SP seperti tahap 5 pada *Browser/ POST Profile.*
j. *Assertion consumer service* membuat *security context* pada SP dan mengalihkan pelanggan ke *resource* tujuan.
k. Pelanggan melakukan *request* pada *resource* tujuan lagi.
l. Sejak *security context* ada maka SP mengembalikan *resource* yang diminta kepada pengguna.

Terdapat skenario lain menurut (Scavo & Cantor, 2005) yaitu Shibboleth *Browser/POST Profile with Attribute Exchange* yang alurnya dapat digambarkan seperti pada Gambar 2.5 berikut:



Gambar 2.5 Shibboleth *Browser/ POST Profile with Attribute Exchange*

Sumber: (Scavo & Cantor, 2005)

Untuk penomoran pada Gambar 2.5 dijelaskan sesuai urutan pada penjelasan sebagai berikut:

a. Pengguna meminta *resource* pada SP *https://sp.example.org/myresource* lalu SP melakukan pemeriksaan keamanan. Apabila SP sebelumnya telah membuat *security context*, maka tahapan dua hingga tujuh tidak perlu dilalui kembali.
b. SP mengarahkan pengguna ke layanan SSO pada IdP.
c. Pengguna meminta layanan SSO pada IdP.
d. Layanan SSO membalas dengan dokumen yang mengandung formulir dalam bentuk HTML.
e. Pelanggan melanjutkan *request* POST pada *assertion consumer service* di SP.
f. *Assertion consumer service* membaca POST *request* lalu melakukan validasi terhadap *signature* pada elemen *Response*. Membuat *security context* di SP dan melewati kontrol ke *attribute requester* untuk inisiasi pertukaran *attribute*. *Attribute requester* mengirimkan pesan SAML ke *attribute authority* (AA) di IdP.
g. AA di IdP menjalankan proses terhadap *request* dan menghasilkan *attributes* yang dibutuhkan sebagai balasan kepada *attribute requester*.
h. *Assertion consumer service* melakukan pemilahan terhada *attributes* lalu memperbaharui *security context* dan mengarahkan pelanggan ke *resource* tujuan.
i. Pelanggan melakukan *request* terhadap *resource* yang dituju lagi.
j. Sejak *security context* ada maka SP mengembalikan *resource* yang diminta kepada pengguna.

Hal ini yang membuat layanan memiliki kapabilitas untuk melakukan Cross-Domain Single Sign-On dan layanan tidak perlu lagi melakukan pemeliharaan terhadap *username* and *password*. Kedua produk tersebut menggunakan SAML sebagai standar yang telah ditetapkan untuk SSO berfungsi untuk melakukan pertukaran data yang akan dijelaskan selanjutnya.

2.5 SAML

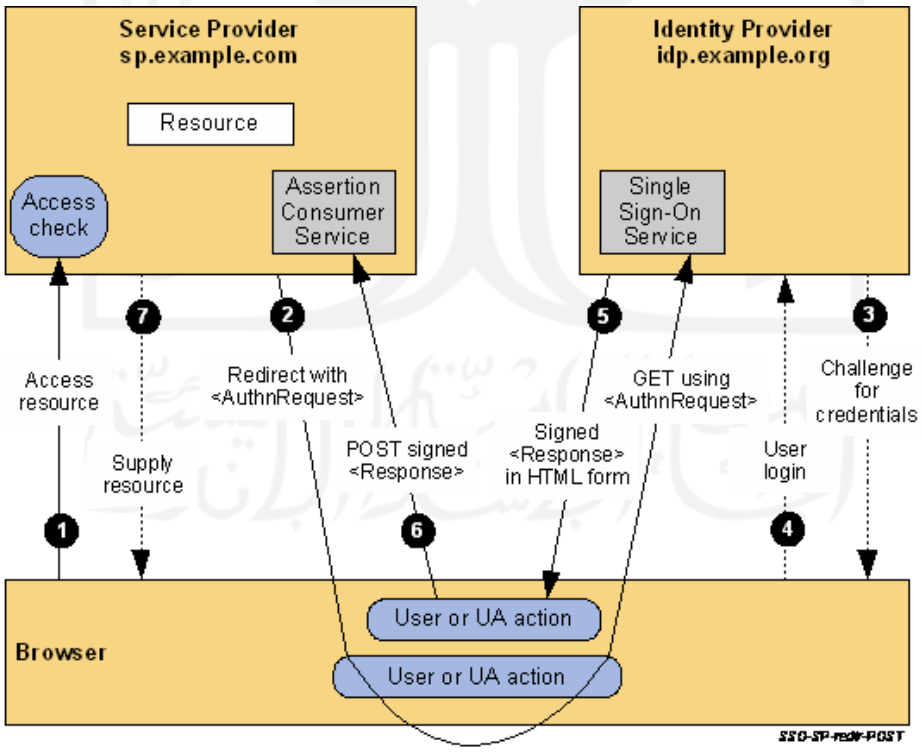
SAML (*Security Assertion Markup Language*) menjadi standar industri dan dokumen yang dibentuk oleh OASIS (Organization for the Advancement of Structured Information Standards) berbasis (*Extensible Markup Language*). Penggunaan SAML banyak ditemukan pada SSO lintas domain (Wilson & Hingnikar, 2019), SAML ini juga digunakan untuk

melakukan pertukaran *authentication* dan *authorization* data antara *domain* yang aman (Radha & Reddy, 2012) dan digunakan pada Shibboleth sebagai standar protokol.

Menurut (OASIS, 2008) *bindings* merupakan pesan bagaimana SAML menangani beragam macam protokol yang digunakan, melalui *transport protocols*, adapun daftar dari bindings yang didefinisikan adalah sebagai berikut:

a. *HTTP Redirect Binding.*
b. *HTTP POST Binding.*
c. *HTTP Artifact Binding.*
d. *SAML SOAP Binding.*
e. *Reverse SOAP* (*PAOS*) *Binding.*
f. *SAML URI Binding.*

Pada *HTTP Redirect Binding* biasanya mengirimkan pesan SAML dengan elemen *AuthRequest* ke IdP dan *HTTP POST Binding* digunakan untuk mengirim kembali pesan SAML dengan elemen *Response* yang mengandung *assertion* kepada SP, adapun alur pada penggunaan *Binding Redirect* dan *POST* dapat digambarkan pada Gambar 2.6 seperti berikut:



Gambar 2.6 *Redirect* dan *POST Binding*

Sumber: (OASIS, 2008)

Proses yang mengikuti penomoran pada Gambar 2.6 akan dijelaskan sesuai dengan daftar berikut:

a. Pengguna yang belum mempunyai valid *session* mencoba untuk mengakses *resource* pada *sp.example.com*. SP menyimpan *resource* URL yang diminta oleh pengguna yang dapat disimpan saat SSO berlangsung.

b. SP mengirim HTTP *redirect response* pada *browser*, serta *Location* pada HTTP *header* yang mengandung tujuan URI dari *Sign-On Service* milik IdP bersama dengan pesan yang berisikan AuthRequest yang telah melalui tahap *encode* dengan DEFLATE sebagai variabel URL *query* dengan nama SAMLRequest, adapun pesan yang belum melakukan proses *encode* dapat digambarkan pada Gambar 2.6 sebagai berikut:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                    ID="identifier_1"
                    Version="2.0"
                    IssueInstant="2004-12-05T09:21:59Z"
                    AssertionConsumerServiceIndex="1">

    <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
    <samlp:NameIDPolicy AllowCreate="true"
                        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>

</samlp:AuthnRequest>
```

Gambar 2.7 Pesan AuthRequest

Browser melakukan proses terhadap *redirect* tersebut lalu melakukan HTTP *Get request* pada IdP Single Sign-On Service dengan parameter *query* SAMLRequest.

c. *Single Sign-On Service* menentukan apakah pengguna telah memiliki *logon security context* pada IdP sebelumnya, jika tidak, IdP akan melakukan interaksi kepada *browser* untuk membuktikan bahwa pengguna memiliki hak akses.

d. Pengguna membuktikan dengan *credentials* yang valid lalu IdP akan membuat *logon ecurity context* untuk pengguna tersebut.

e. IdP *Single Sign-On Service* membuat SAML *assertion* yang mencerminkan *logon security context* dari pengguna. Karena yang digunakan adalah *POST Binding*, *assertion* ditandatangai secara digital dan ditempatkan bersama SAML *response*. Untuk tujuan kemudahan, HTML FORM akan disisipkan otomatis menggunakan *script*, adapun form yang disisipkan di dalamnya dapat digambarkan pada Gambar 2.8 seperti berikut:

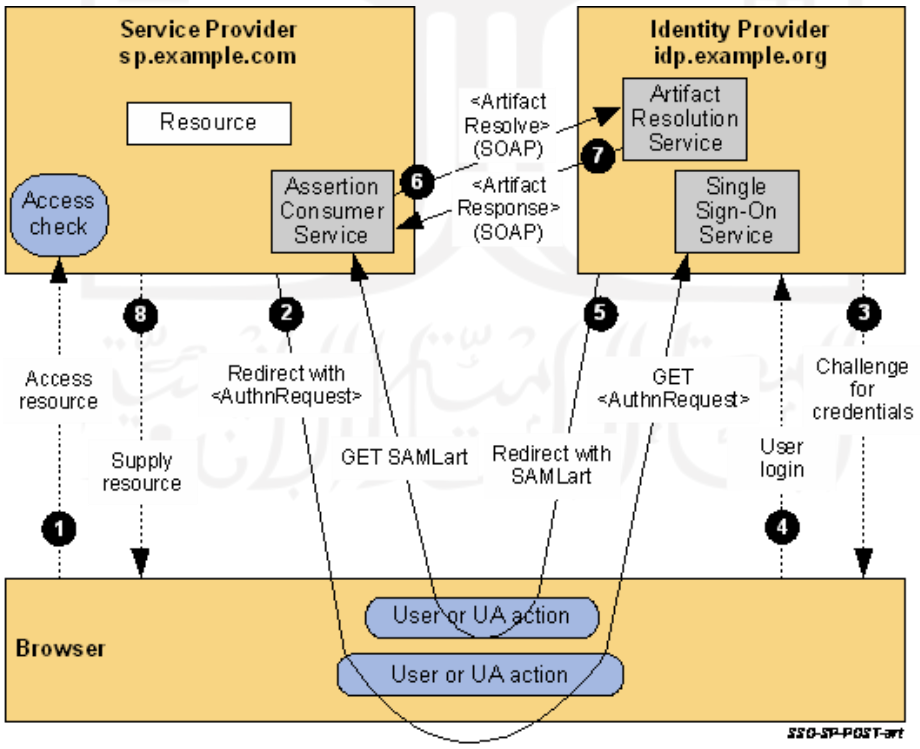
```
<form method="post" action="https://sp.example.com/SAML2/SSO/POST" ... >
    <input type="hidden" name="SAMLResponse" value="response" />
    <input type="hidden" name="RelayState" value="token" />
    ...
    <input type="submit" value="Submit" />
</form>
```

Gambar 2.8 SAML Form

f. *Browser* Mengirimkan *form* pada *Assertion Consumer* Service di SP setelah pengguna melakukan *submit form*.

g. Pemeriksaan dilakukan untuk menentukan apakah pengguna memiliki izin yang benar pada *resource* atau tidak. Jika pemeriksaan lolos, maka *resource* akan diberikan kepada *browser*

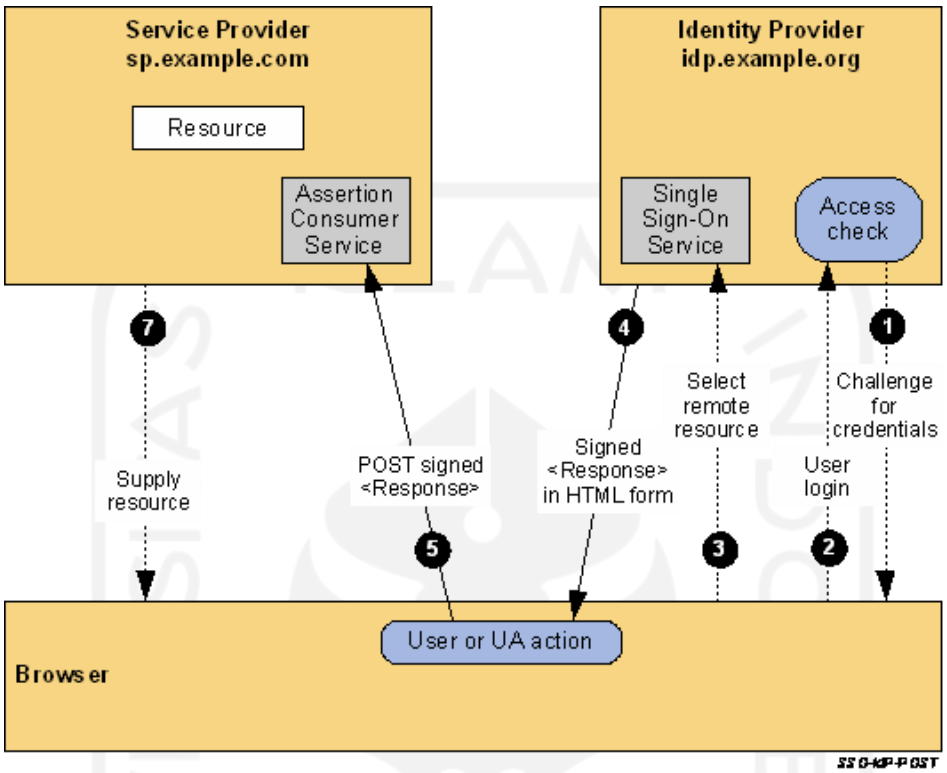
Contoh Alur *POST* dan *Artifact Binding* yang dapat terjadi bisa digambarkan pada Gambar 2.9 seperti berikut:



Gambar 2.9 *POST* dan *Artifact Binding*

Sumber: (OASIS, 2008)

Skenario lain dari alur antara ECP (*Enchanced Client and Proxy*), *Service Provider* dan *Identity Provider* dapat digambarkan pada Gambar 2.10 seperti berikut:



Gambar 2.10 SAML Post Binding

Sumber: (OASIS, 2008)

Seperti pada Shibboleth IdP yang menggunakan SAML dan akan dijelaskan pada bagian selanjutnya.

2.6 Identity Provider

Tujuan kehadiran IdP (*Identity Provider*) adalah untuk mengamankan *identity* dan *attributes* yang telah melakukan *login* agar SP dapat membuat keputusan terhadap kontrol akses pengguna (Graves & Vandenbrink, 2014), selain itu IdP juga menyediakan layanan otentikasi untuk melakukan transaksi data identitas. Tugas utama IdP adalah memberi keputusan apakah pengguna berhasil *login* atau tidak.

Shibboleth IdP bisa menggunakan protokol LDAP (*Lightweight Directory Access Protocol*) untuk melakukan penyimpanan data pengguna, dan juga memegang peran penting di dalam persetujuan proses *login* yang akan menerima SAML *request* serta memberikan SAML *respond* terhadap Shibboleth SP yang mengandung *assertion* dan *attribute*. Adapun beberapa

attribute seperti *displayName*, *surename*, *uid*, dan *email* dan masih banyak lagi sesuai dengan konfigurasi yang diterapkan, *attribute* tersebut yang nantinya akan dimanfaatkan oleh Shibboleth SP. SSO dapat terjalin apabila Shibboleth SP telah dikenali oleh Shibboleth IdP melalui metadata yang didaftarkan. Adapun penjelasan dari Shibboleth SP akan dijelaskan pada bagian selanjutnya.

2.7 Service Provider

Jika IdP adalah layanan penyedia data identitas, maka *Service Provider* adalah aplikasi yang menggunakan layanan tersebut, *Service Provider* atau biasa disingkat menjadi SP ini tidak serta merta bisa menggunakan IdP semaunya, harus ada persetujuan dari kedua belah pihak di mana SP akan mendaftarkan metadata IdP yang akan digunakan lalu mengajukan permintaan untuk mendaftarkan metadata SP pada IdP.

2.8 *Framework* Laravel

Banyaknya layanan yang dihadirkan oleh Laravel memberi kemudahan terhadap pengembang serta membuat efisiensi waktu pada penyelesaian sebuah proyek dengan kerangka kerja menjadi lebih baik jika dibandingkan dengan membuat dari awal atau dikenal dengan sebutan *From Scratch*. *Framework* yang bagus tidak hanya memberikan dasar yang kuat, melainkan kebebasan untuk menyesuaikan (Stauffer, 2019) dengan kebutuhan. Laravel mengatas namakan *clean code* yang terdiri dari kumpulan banyak *library* (Dockins, 2017), dan Composer digunakan untuk menangani banyaknya *depedency* yang tersedia.

Menurut (Laravel LLC, 2022) struktur direktori bawaan pada aplikasi Laravel beserta penjelasan untuk setiap bagian akan dijelaskan sebagai berikut:

a. *app*, merupakan direktori yang berisikan kode utama untuk aplikasi yang nantinya akan dikembangkan.
b. *bootstrap*, merupakan direktori berisikan berkas *app.php* yang bertugas sebagai *bootstrapping framework.*
c. *config*, sesuai dengan nama yang diberikan pada direktori tersebut yaitu direktori yang berisikan konfigurasi aplikasi.
d. *database*, merupakan direktori yang mengandung *migrations*, *model factories*, dan *seeds* di dalamnya.
e. *public*, merupakan direktori yang menjadi tempat *index.php* berada sebagai pintu masuk seluruh pengguna aplikasi.

f. *resources*, merupakan tempat di mana pendukung antarmuka yang belum melalui proses *compile* berada, seperti berkas JavaScript dan CSS.
g. *routes*, merupakan direktori yang mengandung pengaturan rute terhadap *request* pada aplikasi.
h. *storage*, merupakan tempat di mana rekam jejak aplikasi dicatat, seperti *session*, *cache* dan berkas yang dibuat oleh *framework.*
i. *tests*, merupakan tempat untuk menyimpan berkas pengujian seperti PHPUnit yang dijalankan menggunakan artisan.
j. *vendor*, merupakan tempat di mana seluruh pustaka pendukung untuk menjalankan *framework* Laravel.

Laravel menggunakan alat bernama Composer yang kegunaannya hampir mirip dengan *Node Package Manager* (NPM), alat tersebut dikembangkan oleh Nils Adermann, Jordi Boggio serta komunitas yang berkontribusi pada *repository* di GitHub, alat yang diterbitkan pada tanggal 01-Maret-2012 tersebut merupakan hasil *porting* untuk PHP dari pustaka *libzypp* yang mengandung *Satisfiability Solver* di dalamnya. Composer merupakan alat terbaik mengelola *package* yang tersedia untuk pengembang pada ruang lingkup proyek PHP, alat tersebut membuat pengembang bisa menyatakan apa saja yang dibutuhkan oleh proyek serta data seperti versi berapa yang digunakan, kebutuhan tersebut dinyatakan pada berkas yang bernama *composer.json* dan ditulis dengan format JSON (Machek, 2014). Composer tidak hanya untuk pustaka pihak ketiga melainkan juga bisa untuk mengelola pustaka buatan sendiri (Machek, 2014). Salah satu contoh pustaka yang bisa digunakan dengan Composer adalah Short URL yang akan dijelaskan pada bagian selanjutnya.

2.9 Short URL

Short URL adalah salah satu dari berbagai macam pustaka yang tersedia untuk *framework* Laravel, pustaka yang dikembangkan oleh Ash Allen selaku pemilik beserta *contributor* yang lain dan *release* pada GitHub ini memiliki tujuan untuk memperpendek URL pada aplikasi berbasis website. Saat ini pengembangan telah sampai pada versi v6.2.0 yang terbit pada 26-Nov-2021. Terdapat persyaratan yang harus dipenuhi sebelum menggunakan pustaka ini yaitu PHP dan Laravel, pustaka ini juga menggunakan pustaka lain bernama *hashids* yang memiliki persyaratan untuk bekerja diantaranya adalah dengan mengaktifkan modul *bcmath* dan *gmp* pada PHP.

Hashid menggunakan algoritma *Fisher-Yates shuffle* yang telah dimodifikasi sesuai kebutuhan yaitu dengan mengacak karakter berdasarkan *salt* yang diberikan, algoritma tersebut berfungsi untuk membuat permutasi acak dari urutan terbatas berdasarkan *base62* secara bawaan, karakter yang ada pada *base62* ini mengandung seluruh huruf kapital maupun kecil serta angka dari 0 hingga 9 dan apabila ditotal hasilnya akan menjadi 62 karakter, dan tidak menutup kemungkinan untuk menghapus karakter yang tidak ingin digunakan. Urutan terbatas tersebut juga di acak berdasarkan *salt* terlebih dahulu sebelum digunakan.

BAB III
METODOLOGI PENELITIAN

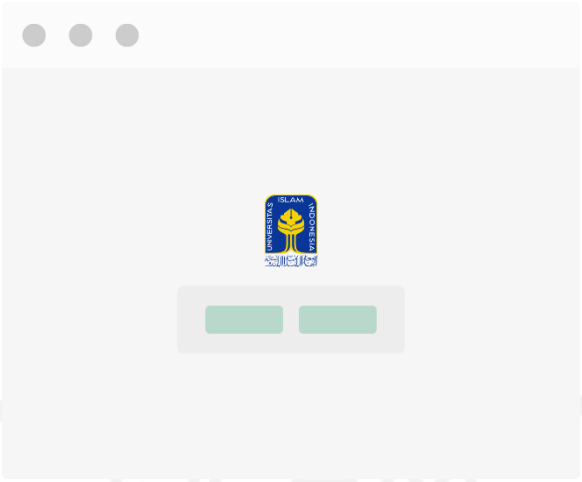
3.1 Perancangan *Dashboard*

Rancangan aplikasi ShortyURL yang akan dikembangkan nantinya menggunakan Figma sebagai aplikasi bantuan dalam pembuatan *prototype*, adapun tinjauan dari keseluruhan *prototype* digambarkan pada Gambar 3.1 seperti berikut:

Gambar 3.1 Tinjauan seluruh *prototype*

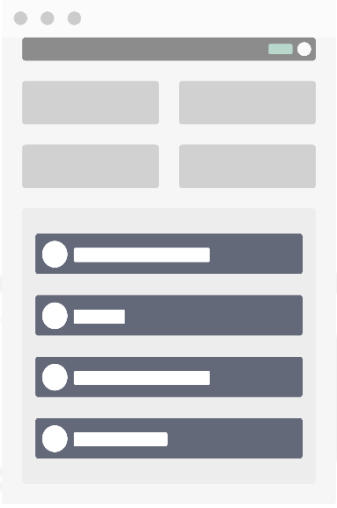
Berikut merupakan daftar serta penjelasan seluruh *prototype* untuk laman yang terdapat pada aplikasi ShortyURL:

a. *Prototype log-in* merupakan halaman yang dikunjungi ketika pertama kali membuka aplikasi ShortyURL untuk menikmati layanan maupun fitur yang ada di dalamnya, adapun gambar yang merujuk pada *prototype* tersebut digambarkan pada Gambar 3.2 seperti berikut:



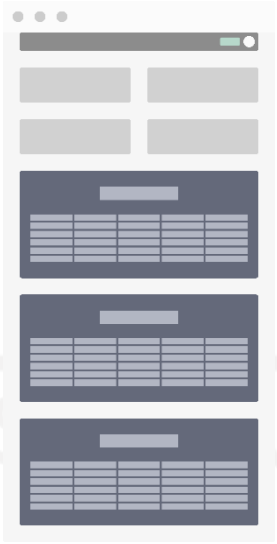
Gambar 3.2 *Prototype Log-In*

b. *Prototype Dashboard* merupakan halaman utama setelah *log-in*, di mana pada halaman tersebut *user* dapat membuat URL singkat, melihat daftar URL yang telah disingkat, rujukan tertinggi, total pengunjung, seluruh data bergantung pada akun *user* yang *log-in* saat itu, adapun tangkapan layar *prototype dashboard* tersebut digambarkan pada Gambar 3.3 seperti berikut:



Gambar 3.3 *Prototype Dashboard*

c. *Prototype Dashboard Analytic* merupakan halaman yang menampilkan seluruh data yang berada pada sistem tentang tautan yang dipersingkat seperti rujukan tertinggi dari seluruh tautan dan tidak terbatas pada *user* yang saat itu log-in, adapun tangkapan layar pada *prototype dashboard analytic* tersebut digambarkan pada Gambar 3.4 seperti berikut:

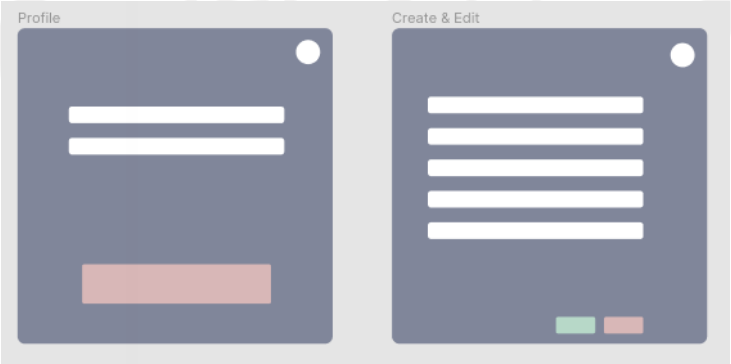


Gambar 3.4 *Prototype Analytic Dashboard*

d. *Prototype Detail* merupakan bagian dari halaman *dashboard* yang merupakan sekumpulan data tentang tautan yang telah dipendekkan secara rinci, adapun tangkapan layar dari *prototype detail* tersebut digambarkan pada Gambar 3.5 seperti berikut:

Gambar 3.5 *Prototype Detail*

e. *Pop-up* merupakan bagian dari halaman *dashboard* untuk membuat tautan pendek yang di dalamnya terdapat formulir dengan bentuk hampir sama, pemicu *pop-up* untuk membuat tautan pendek tersedia pada bagian atas halaman, sedangkan pemicu untuk mengubah terdapat pada bagian *detail* tautan. Tersedia juga tombol *profile* terletak di pojok kanan atas untuk memicu *pop-up* dan di dalamnya terdapat pilihan untuk pindah halaman *dashboard* serta *log-out*, adapun tangkapan layar pada *prototype pop-up* tersebut digambarkan pada Gambar 3.6 seperti berikut:



Gambar 3.6 *Prototype Popup*

f. *Prototype Success & Failure* merupakan keterangan ketika *user* berhasil melakukan tindakan seperti membuat atau mengubah untuk *prototype success*, dan sebaliknya pada *prototype failure* apabila *user* gagal melakukan tindakan atau melakukan kesalahan, adapun tangkapan layar dari *prototype success & failure* tersebut digambarkan pada Gambar 3.7 seperti berikut:

Gambar 3.7 *Prototype Success & Failure*

3.2 Implementasi SSO

Untuk pemasangan Shibboleth SP penulis menambahkan *shibboleth repository* dengan memasang *switchaai-apt-source_1.0.0_all.deb* yang didapat dari alamat *https://pkg.switch.ch/switchaai/debian/dists/buster/main/binary-all/misc/switchaai-apt-source_1.0.0_all.deb* lalu melakukan pembaruan *repository* dengan *apt*. Selanjutnya penulis melakukan pemasangan *shibboleth* sesuai rekomendasi menggunakan *apt* dengan parameter *--install-recommends*.

Selanjutnya penulis meletakkan berkas *metadata* IdP pada SP yang diunduh pada alamat *https://idp.uii.ac.id/idp/shibboleth* dan diberi nama *idp.uii.ac.id.xml*, lalu penulis membuat sertifikat menggunakan *shib-keygen* dengan parameter *-h shortyurl.uii.ac.id -f* serta melakukan *restart* pada *shibd* menggunakan *systemctl* setelah penulis mendaftarkan IdP dan *metadata* tersebut pada konfigurasi Shibboleth SP dalam berkas *shibboleth2.xml*, adapun tangkapan layar yang digambarkan pada Gambar 3.8 adalah seperti berikut:

```
...
    ...
        <SSO entityID="https://idp.uii.ac.id/idp/shibboleth"
             discoveryProtocol="SAMLDS" discoveryURL="https://ds.example.org/DS/WAYF">
          SAML2
        </SSO>
    ...
    <MetadataProvider type="XML" validate="true" path="idp.uii.ac.id.xml"/>
    <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
...
```

Gambar 3.8 Konfigurasi *shibboleth2.xml*

Selanjutnya penulis mengaktifkan *SSL* dengan cara membuat sertifikat menggunakan *openssl* seperti yang digambarkan pada Gambar 3.9 berikut:

```
sudo openssl genrsa -des3 -passout pass:1234 -out shortyurl.uii.ac.id.key 2048
sudo openssl rsa -passin pass:1234 -in shortyurl.uii.ac.id.key -out shortyurl.uii.ac.id.key
sudo openssl req -new -key shortyurl.uii.ac.id.key -out shortyurl.uii.ac.id.csr
sudo openssl x509 -req -days 365 -in shortyurl.uii.ac.id.csr -signkey shortyurl.uii.ac.id.key -out shortyurl.uii.ac.id.crt
```

Gambar 3.9 Pembuatan sertifikat dengan *openssl*

Lalu penulis menjalankan *systemctl* lagi untuk melakukan *restart* Apache setelah melakukan konfigurasi *virtual host* pada Apache yang digambarkan pada Gambar 3.10 seperti berikut:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin 15523007@students.uii.ac.id
        ServerName shortyurl.uii.ac.id
        DocumentRoot /var/www/shortyurl.uii.ac.id/public/
        SSLEngine on
        SSLCertificateFile      /etc/apache2/ssl/shortyurl.uii.ac.id.crt
        SSLCertificateKeyFile   /etc/apache2/ssl/shortyurl.uii.ac.id.key
        <Location /Shibboleth.sso>
            ShibUseHeaders On
            SetHandler shib
            AuthType shibboleth
            ShibRequestSetting requireSession true
            Require shibboleth
        </Location>
        <Directory /var/www/shortyurl.uii.ac.id/public/>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride All
            Require all granted
        </Directory>
    </VirtualHost>
</IfModule>
```

Gambar 3.10 Konfigurasi Apache *virtual host*

Di sini penulis mengubah berkas *.htaccess* agar Laravel bisa mendapatkan informasi Shibboleth dari *server* yang dicantumkan pada variabel *$_SERVER*, adapun perubahan yang digambarkan pada Gambar 3.11 adalah sebagai berikut:

```
<IfModule>
    AuthType shibboleth
    Require shibboleth
    ...
</IfModule>
```

Gambar 3.11 Konfigurasi *.htaccess*

Selanjutnya penulis memanfaatkan data pada variabel *$_SERVER* tersebut untuk menentukan izin terhadap pengguna setelah *login* yang dituliskan dalam fungsi pada *controller* digambarkan pada Gambar 3.12 sebagai berikut:

```
...
public function uii(){
    $Normalized = [
        'displayName'    => $_SERVER['displayName'],
        'givenName'      => $_SERVER['givenName'],
        'mail'           => $_SERVER['mail'],
        'memberOf'       => $_SERVER['employeeType'],
        'uid'            => $_SERVER['uid'],
    ];
    ...
    $SearchInDb = User::where('email', $Normalized['mail'])->first();
    ...
    if($SearchInDb){
        Auth::login($SearchInDb);
    }else{
        Auth::loginUsingId(User::create($Normalized)->id);
    }
    ...
    return redirect(route('user.dashboard'));
}
...
```

Gambar 3.12 Fungsi *login* UII

Dengan demikian ShortyURL yang dikembangkan telah memiliki sistem SSO menggunakan Shibboleth SP.

3.3 Implementasi *Shortener*

Di sini penulis menggunakan Composer selaku *package manager* untuk memasang *package* Short URL dari *ashallendesign* pada Laravel dengan perintah sebagai berikut:

```
composer require ashallendesign/short-url;
```

Setelah dipasang penulis melakukan konfigurasi dari *package* Short URL dan menambahkan tabel yang digunakan untuk menyimpan data pengunjung menggunakan Artisan dengan perintah sebagai berikut:

```
php artisan vendor:publish --provider="AshAllenDesign\ShortURL\Providers\ShortURLProvider";
php artisan migrate;
```

Selanjutnya penulis menghilangkan *prefix* bawaan pada pengaturan Short URL agar *user* langsung mengakses URL yang telah dipendekkan tanpa terganggu oleh *prefix*, hal ini bisa membuat hasil URL yang disingkat lebih pendek jika dibandingkan sama URL dengan *prefix* ketika diakses oleh *user*. Hal ini bisa dicapai dengan merubah konfigurasi *disable_default_route* pada *config/short-url.php* menjadi *true* dan tambahkan indikator segmen pada URI

dengan nama *shortURLKey* di akhir *file* konfigurasi *routes/web.php*, indikator tersebut nantinya akan disuntikkan melalui *callback* ke *ShortURLController* yang merupakan *controller* bawaan pustaka Short URL dan terletak di dalam *vendor/ashallendesign/short-url/src/Controllers.*

Setelah melakukan konfigurasi yang dibutuhkan penulis menyediakan halaman untuk menangani *input* dari *user* ketika ingin mempersingkat URL maupun mengubahnya, dengan cara menambahkan *controller* menggunakan Artisan melalui perintah sebagai berikut:

```
php artisan make:controller ShortController -r;
```

Selanjutnya penulis menyediakan rute pada berkas *routes/web.php* menggunakan metode yang sesuai dengan Controller untuk menangani tindakan penambahan atau perubahan yang dilakukan *user*, adapun gambar pada rute tersebut digambarkan seperti Gambar 3.13 berikut:

```
Route::post('/short', [App\Http\Controllers\ShortController::class, 'store']);
Route::post('/shortUpdate', [App\Http\Controllers\ShortController::class, 'update']);
```

Gambar 3.13 Short URL *web.php*

Pada metode *store* yang ada di dalam Controller *ShortController*, penulis melakukan *instantiation builder* dari *package* Short URL dan mengisi parameter menggunakan *chaining method*, sesuai dengan *input user* yang dapat di ambil pada variabel *request* untuk membuat atau mempersingkat URL, adapun fungsi yang digambarkan pada Gambar 3.14 adalah seperti berikut:

```
...
public function store(Request $request)
{
    ...
    $builder = new \AshAllenDesign\ShortURL\Classes\Builder();
    $shortURLObject = $builder->destinationUrl($request->long_url)
                              ->urlKey($request->url)
                              ->redirectStatusCode(302)
                              ->make();

    short_urls::where('id', $shortURLObject->id)->update([
        'default_short_url' => url('/'. $request->url);
    ]);
    ...
}
...
```

Gambar 3.14 Fungsi membuat URL singkat baru

Masih dengan Controller yang sama namun kali ini penulis merubah metode *update* untuk menangani perubahan data yang dilakukan *user*, adapun tangkapan layar yang digambarkan pada Gambar 3.15 seperti berikut:

```
...
public function update(Request $request)
{
    $shortURL = \AshAllenDesign\ShortURL\Models\ShortURL::find($request->id);
    $data_update = [];

    if(!$shortURL->visits->count()){
        $data_update += [
            'url_key'       => $request->url,
        ];
    }
    $data_update += [
        'destination_url'   => $request->long_url,
        'title'             => $request->title
    ];

    short_urls::where('id', $request->id)->update($data_update);
    ...
}
```

Gambar 3.15 Fungsi mengubah data URL yang telah disingkat

3.4 Pengujian

Dalam pengujian dengan metode *white box* seluruh uji coba dibantu oleh pihak ketiga bernama PHPUnit yang telah diintegrasikan dengan *framework* Laravel, pengujian ini dilakukan untuk mengindikasi kesalahan yang terdapat pada saat pengembangan.

3.4.1 White Box

Penulis melakukan pengujian dengan metode *white box* mengikuti rancangan yang ada pada daftar berikut:

a. *is mysql connected* menguji apakah konektivitas *database* dan aplikasi berjalan dengan semestinya melalui potongan kode yang dapat digambarkan pada Gambar 3.16 sebagai berikut:

```
use Illuminate\Support\Facades\DB;
...
public function test_is_mysql_connected()
{
    DB::connection('mysql')->getPDO();
    $this->assertTrue(true);
}
```

Gambar 3.16 Uji coba fungsi *is mysql connected*

b. *is shorturl package loadable* menguji pustaka ShortURL yang digunakan untuk menangani tautan pendek tersebut bisa diakses atau tidak, dapat digambarkan pada Gambar 3.17 sebagai berikut:

```
public function test_is_shorturl_package_loadable()
{
    new \AshAllenDesign\ShortURL\Models\ShortURL();
    $this->assertTrue(true);
}
```

Gambar 3.17 Uji coba *is shorturl package loadable*

c. *is geolocation website accessible* menguji apakah *website* yang digunakan untuk mendapatkan detail lokasi berdasarkan IP dapat diakses atau tidak, dapat digambarkan pada Gambar 3.18 sebagai berikut:

```php
public function test_is_geolocation_webite_accessible()
{
    unserialize(file_get_contents('http://www.geoplugin.net/php.gp?ip=127.0.0.1'));
    $this->assertTrue(true);
}
```

Gambar 3.18 Uji coba *is geolocation website accessible*

d. *is shorted url get redirected* menguji apakah tautan yang dipendekkan ketika diakses akan diarahkan kembali ke tautan asli, adapun tangkapan layar yang dapat digambarkan pada Gambar 3.19 sebagai berikut:

```php
use App\Models\short_urls;
...
public function test_is_shorted_url_get_redirected()
{
    $short_urls = short_urls::limit(3)->get();
    foreach ($short_urls as $short_urls_data ) {
        $response = $this->get('/' . $short_urls_data->url_key);
        $response->assertRedirect($short_urls_data->destination_url);
    }
}
```

Gambar 3.19 Uji coba *is shorted url get redirected*

e. *is unauthorize get redirected from dashboard* menguji apakah pengguna akan diarahkan kembali ke halaman *log-in* jika belum melakukannya tetapi mencoba untuk mengakses halaman dasbor, adapun potongan kode program uji coba tersebut dapat digambarkan pada Gambar 3.20 sebagai berikut:

```php
public function test_is_unauthorize_get_redirected_from_dashboard()
{
    $response = $this->get(route('admin.dashboard'));
    $response->assertRedirect();
    $response = $this->get(route('analytic.dashboard'));
    $response->assertRedirect();
}
```

Gambar 3.20 Uji coba *is unauthorize get redirected from dashboard*

Setelah program untuk pengujian selesai dibuat, penulis menjalankan PHPUnit melalui Artisan yang telah di sederhanakan oleh Laravel dengan perintah *php artisan test*.

3.4.2 Black Box

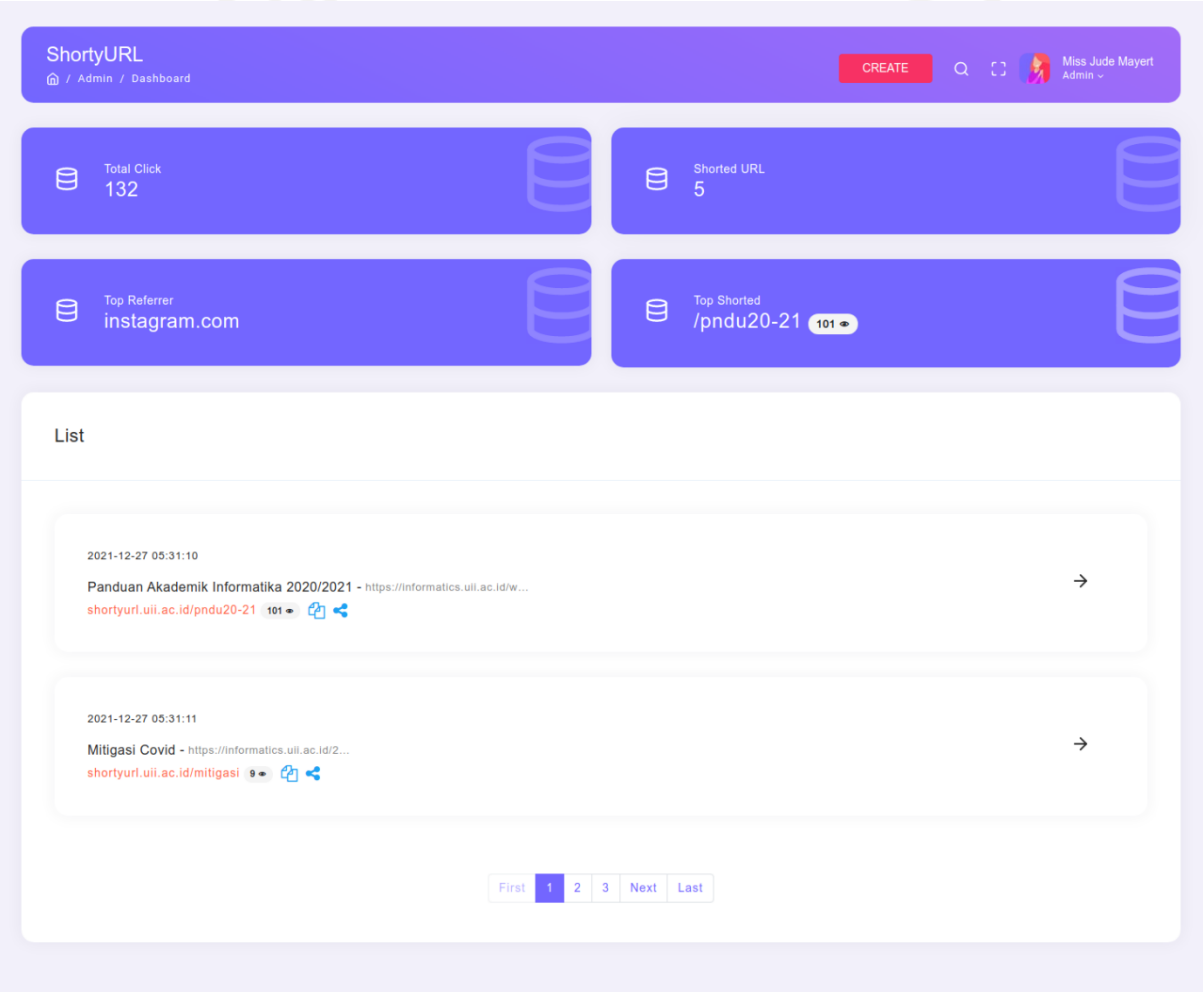
Penulis melakukan pengujian dengan metode *black box* yang mengikuti rancangan seperti pada daftar berikut:

a. *Log-in*
b. *Log-out*
c. Membuat tautan pendek
d. Membuka tautan pendek
e. Mengubah data tautan pendek
f. Membuka dasbor
g. Membuka dasbor analisis
h. Melihat rujukan tertinggi
i. Melihat tautan pendek terpopuler
j. Melihat jumlah pengunjung
k. Melihat jumlah tautan pendek
l. Melihat lokasi pengunjung

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

Pengerjaan ShortyURL telah mencapai final mengikuti prototipe yang telah dibuat sebelumnya sebagai gambaran. Hasil pada bagian *dashboard* akan terlihat seperti tangkapan layar yang digambarkan pada Gambar 4.1 berikut:



Gambar 4.1 Hasil *dashboard*

Adapun hasil implementasi yang telah dicapai sesuai dengan penjelasan pada bab sebelumnya selain *dashboard* akan dijelaskan pada bagian selanjutnya.

4.1.1 Implementasi SSO

Hasil dari konfigurasi Shibboleth SP dapat dilihat menggunakan perintah *shibd -t* di *terminal* yang digambarkan pada Gambar 4.2 berikut:

```
➜ ~  shibd -t
overall configuration is loadable, check console or log for non-fatal problems
➜ ~
```

Gambar 4.2 Hasil keberhasilan konfigurasi Shibboleth SP.

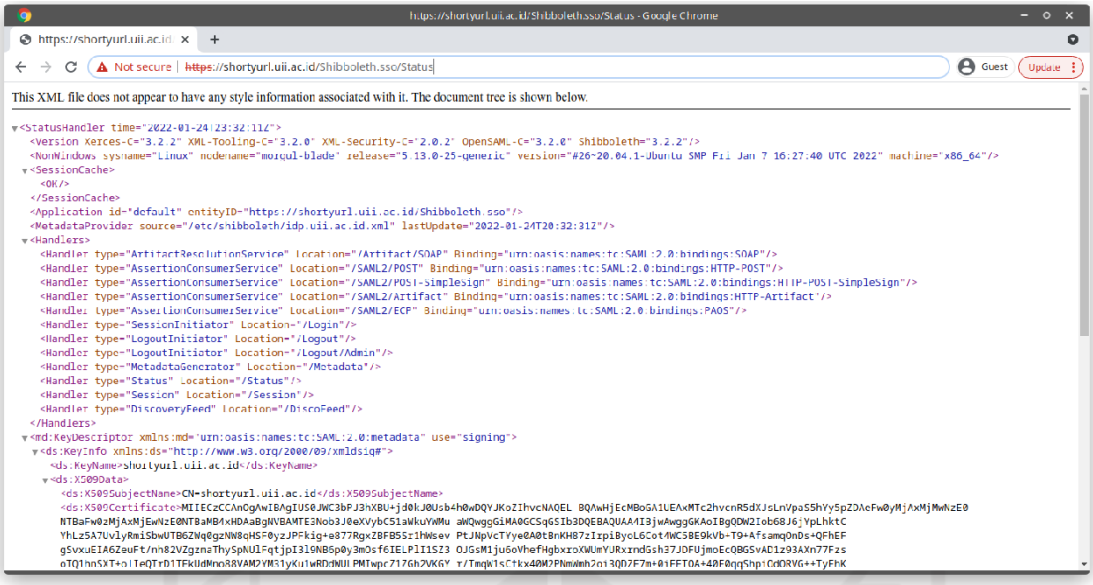
Hasil dari Shibboleth SP yang berjalan bisa dilihat menggunakan perintah *systemctl status shibd* di *terminal* yang dapat digambarkan pada Gambar 4.3 berikut:

```
➜ ~  systemctl status shibd
● shibd.service - Shibboleth Service Provider Daemon
     Loaded: loaded (/lib/systemd/system/shibd.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-01-26 17:27:09 WIB; 7h ago
       Docs: man:shibd(8)
             https://wiki.shibboleth.net/confluence/display/SP3/Home
   Main PID: 768 (shibd)
      Tasks: 6 (limit: 1087)
     Memory: 4.4M
     CGroup: /system.slice/shibd.service
             └─768 /usr/sbin/shibd -f -F

Jan 26 17:27:08 morgul-blade systemd[1]: Starting Shibboleth Service Provider Daemon...
Jan 26 17:27:09 morgul-blade systemd[1]: Started Shibboleth Service Provider Daemon.
➜ ~
```

Gambar 4.3 Hasil status berjalannya Shibboleth SP.

Hasil dari informasi yang diringkas oleh Shibboleth SP dapat dilihat pada tautan *https://shortyurl.uii.ac.id/Shibboleth.sso/Status* yang digambarkan seperti pada Gambar 4.4 berikut:



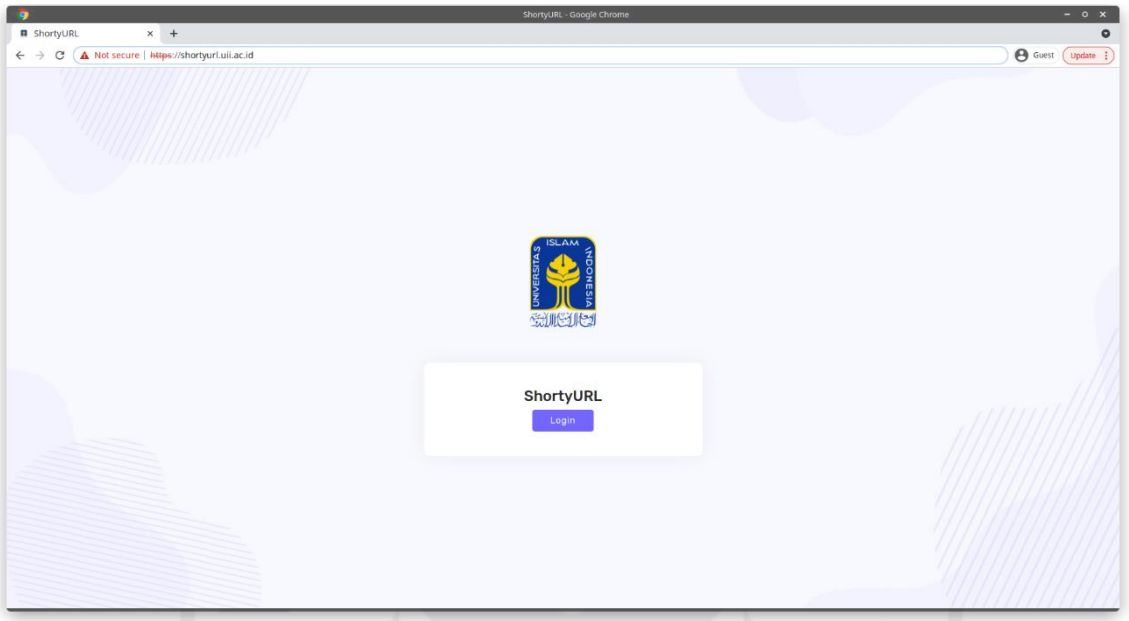
Gambar 4.4 Hasil Shibboleth SP *status*

Selanjutnya merupakan hasil konfigurasi Apache bisa dilihat menggunakan perintah *apache2ctl -S* di *terminal* yang digambarkan pada Gambar 4.5 seperti berikut:

```
➜ ~  apache2ctl -S
VirtualHost configuration:
*:443                  shortyurl.uii.ac.id (/etc/apache2/sites-enabled/shortyurl.conf:2)
ServerRoot: "/etc/apache2"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/var/log/apache2/error.log"
Mutex default: dir="/var/run/apache2/" mechanism=default
Mutex mpm-accept: using_defaults
Mutex watchdog-callback: using_defaults
Mutex rewrite-map: using_defaults
Mutex ssl-stapling-refresh: using_defaults
Mutex ssl-stapling: using_defaults
Mutex ssl-cache: using_defaults
PidFile: "/var/run/apache2/apache2.pid"
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="www-data" id=33 not_used
Group: name="www-data" id=33 not_used
➜ ~
```

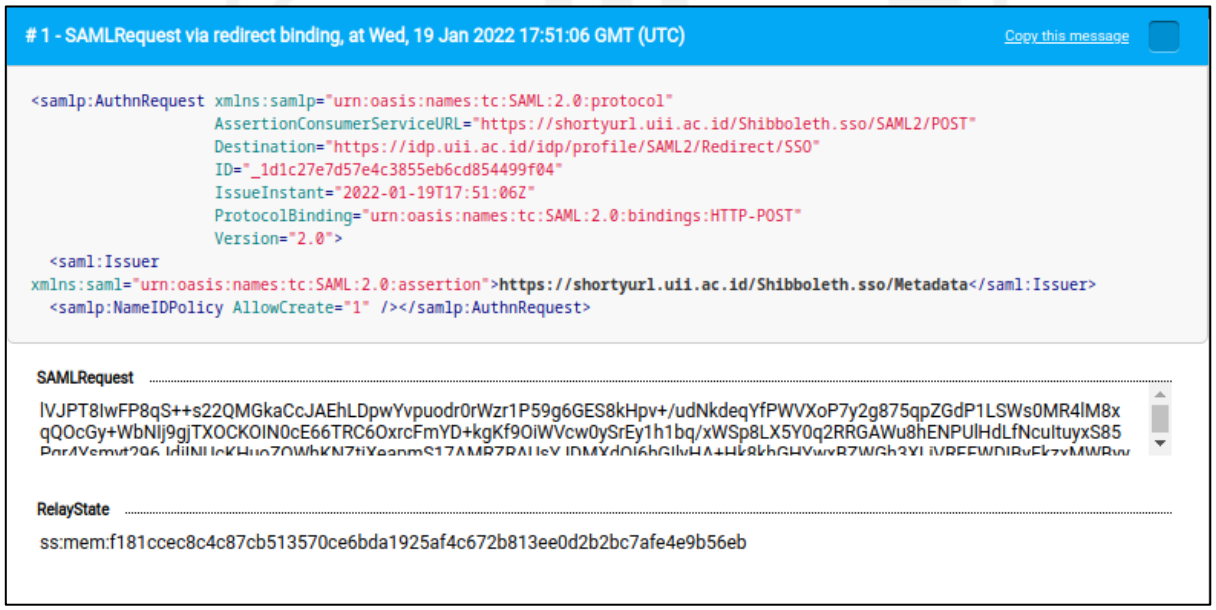
Gambar 4.5 Hasil dari konfigurasi antarmuka pada Apache

Selanjutnya merupakan hasil antarmuka yang telah dibuat sebagai gerbang sistem SSO terletak pada halaman awal aplikasi ShortyURL dan SSL yang diterapkan juga meliputi keseluruhan aplikasi. Adapun tangkapan layar dapat digambarkan pada Gambar 4.6 seperti berikut:



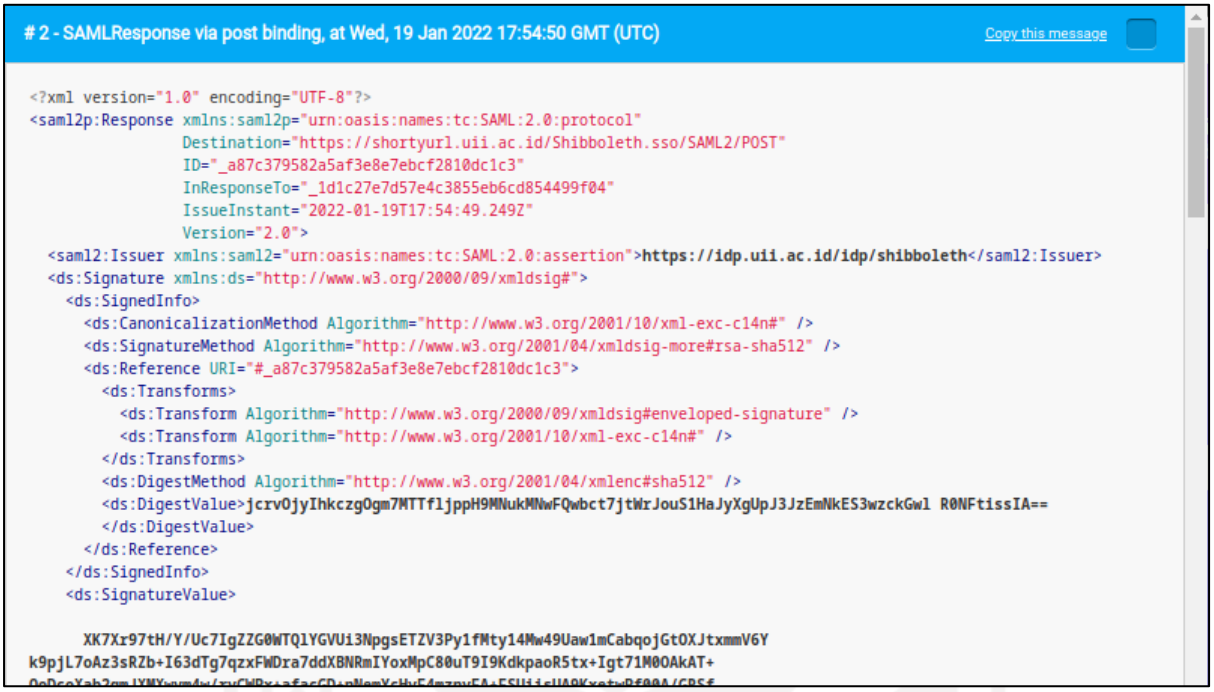
Gambar 4.6 Hasil halaman awal ShortyURL

Sistem SSO yang menggunakan SAML berhasil diimplementasikan pada ShortyURL, adapun hasil dari *SAML Request* yang ditangkap dapat dilihat menggunakan *SAML Message Decoder* dan digambarkan pada Gambar 4.7 seperti berikut:



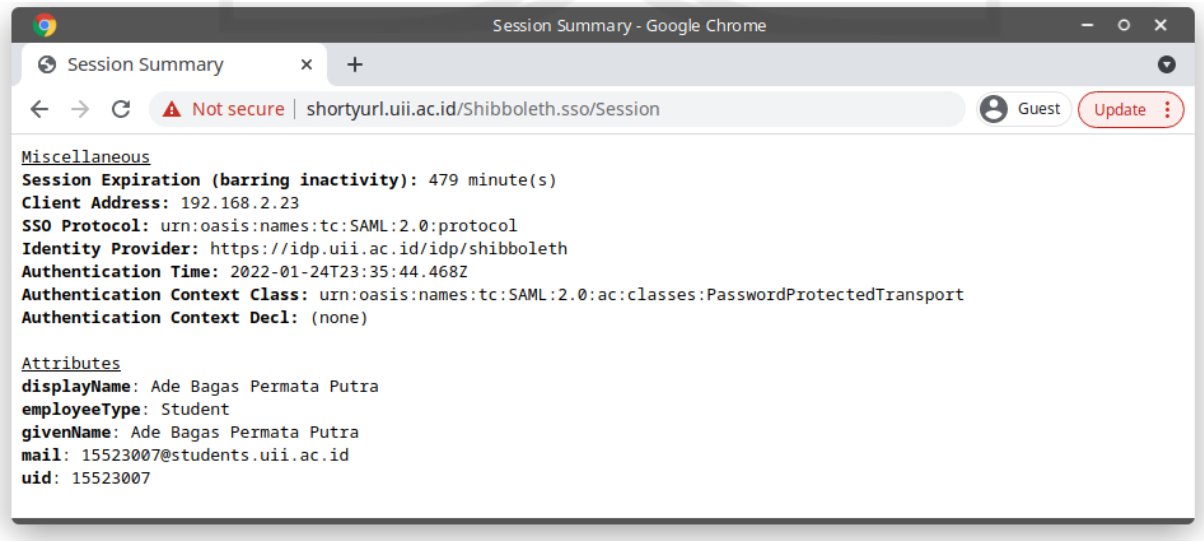
Gambar 4.7 Hasil SAML Request

Selanjutnya hasil dari *SAML Respond* yang didapatkan setelah melakukan *SAML Request* dapat dilihat menggunakan *SAML Message Decoder* dan digambarkan pada Gambar 4.8 seperti berikut:



Gambar 4.8 Hasil SAML Response

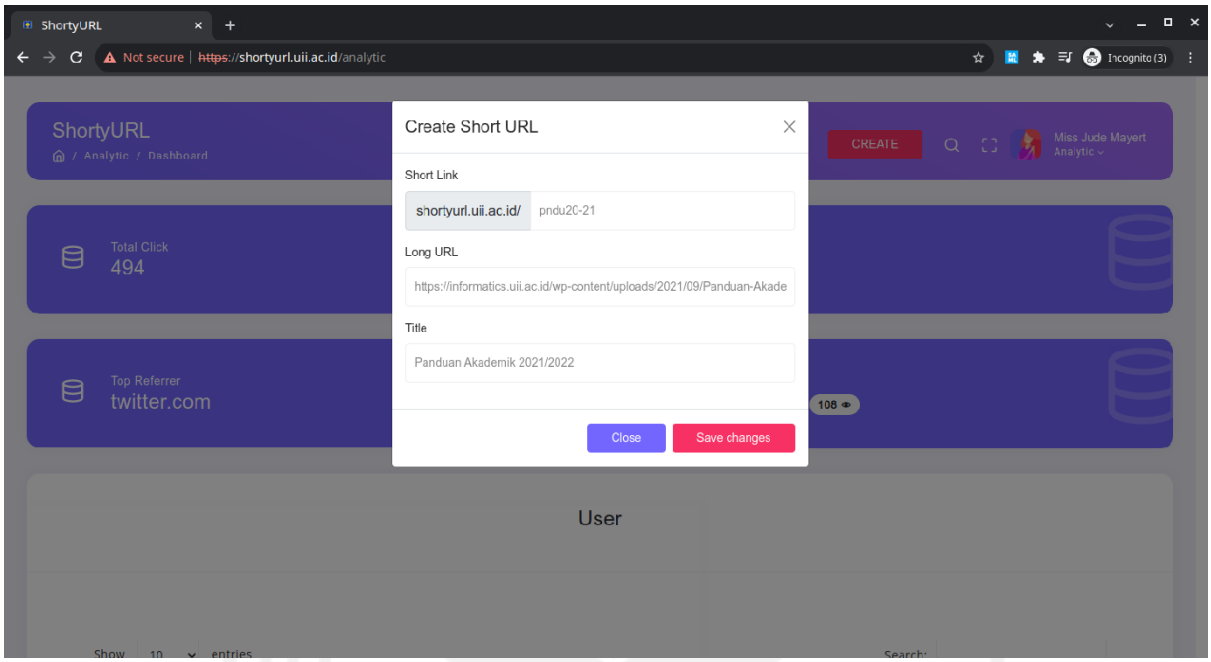
Hasil *attribute* yang terima oleh Shibboleth dapat dilihat pada tautan *https://shortyurl.uii.ac.id/Shibboleth.sso/Session* setelah *login*, dapat digambarkan pada Gambar 4.9 seperti berikut:



Gambar 4.9 Hasil *Attribute* yang diterima oleh SP

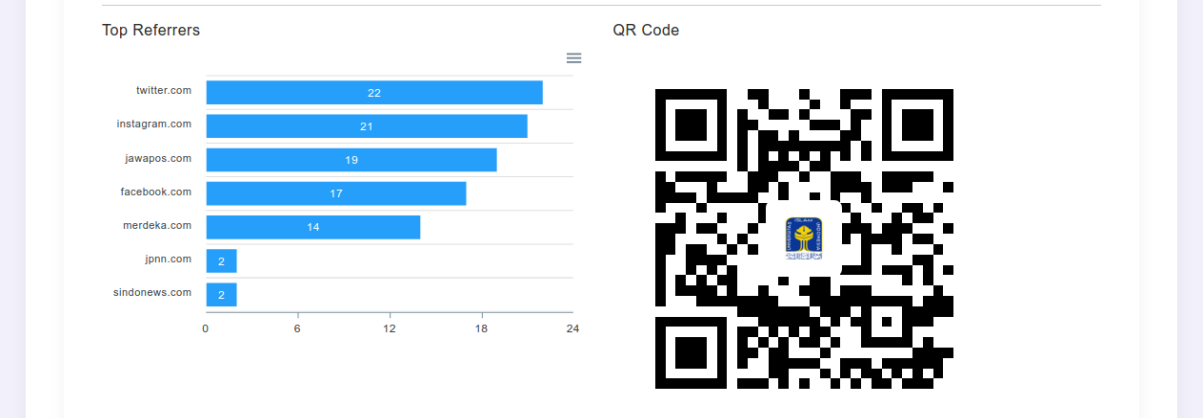
4.1.2 Implementasi *Shortener*

Hasil dari tampilan untuk pembuatan URL singkat yang merujuk pada *prototype* digambarkan pada Gambar 4.10 seperti berikut:



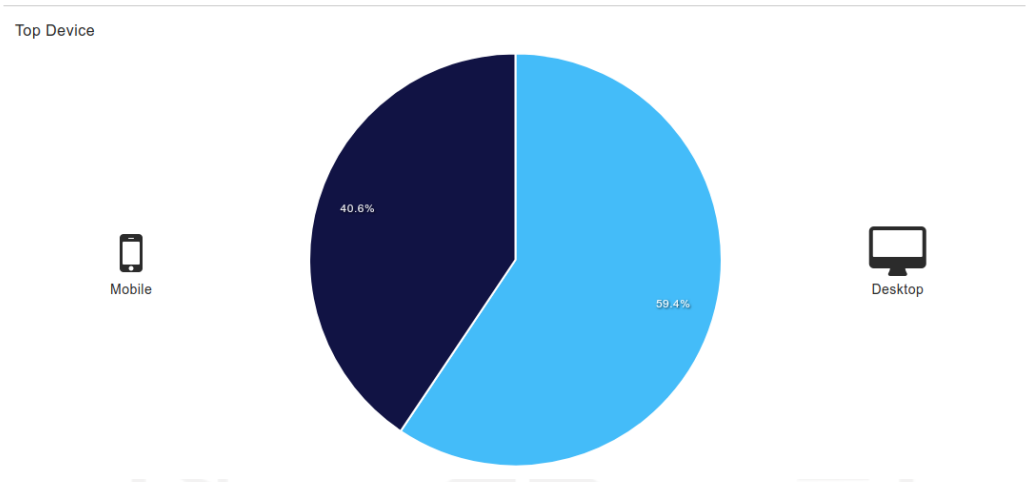
Gambar 4.10 Hasil formulir membuat URL singkat

Hasil dari tampilan untuk melihat kelengkapan informasi detail *Top Referrers* mengenai tautan spesifik, menjadi satu bagian dengan *QR Code* yang digambarkan pada Gambar 4.11 seperti berikut:



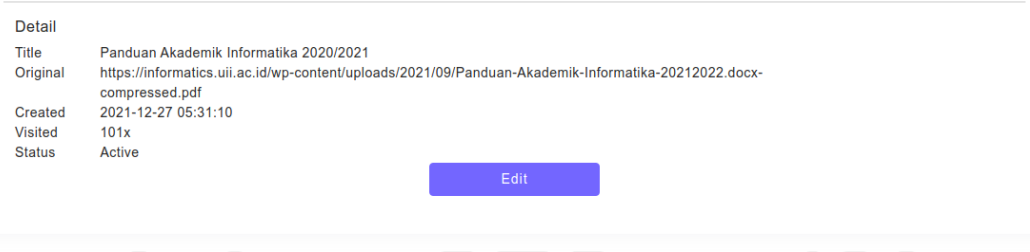
Gambar 4.11 Hasil detail *Top Referrers* dan *QR Code*.

Hasil dari tampilan untuk melihat detail *Top Device* pada tautan digambarkan pada Gambar 4.12 seperti berikut:



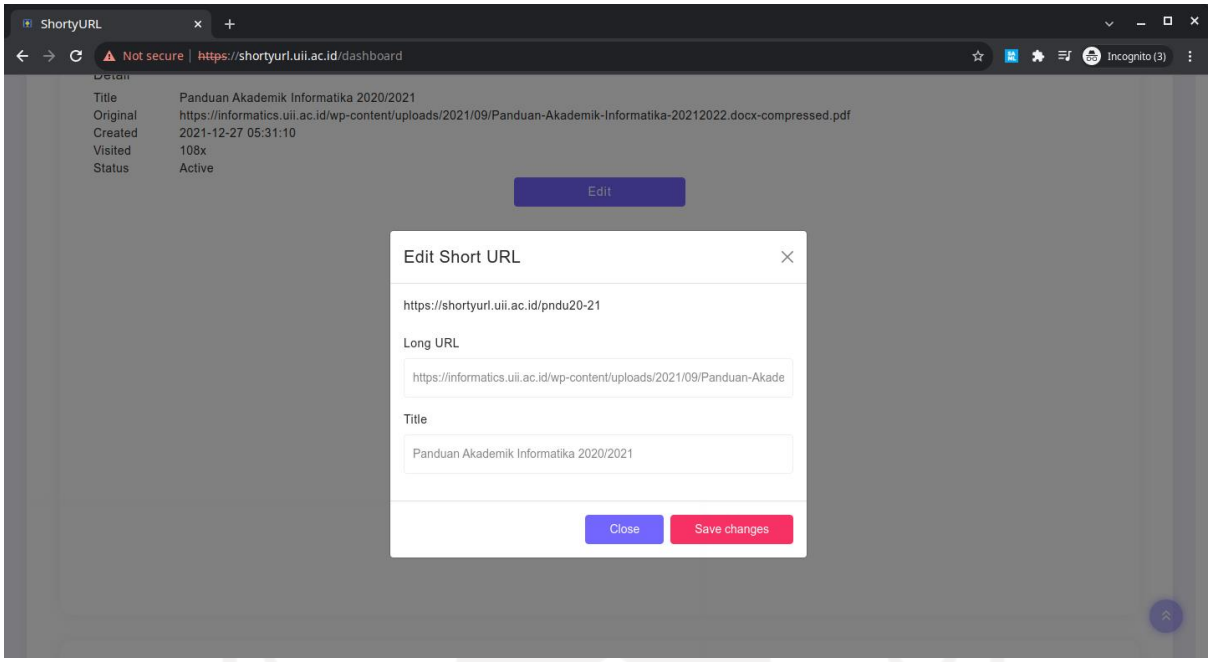
Gambar 4.12 Hasil detail *Top Device*

Hasil mengenai informasi tautan secara rinci yang ditampilkan pada detail dapat digambarkan pada Gambar 4.13 seperti berikut:



Gambar 4.13 Hasil detail informasi tautan secara rinci.

Selanjutnya merupakan hasil akhir dari tampilan yang dibentuk untuk melakukan perubahan data URL pendek digambarkan pada Gambar 4.14 seperti berikut:



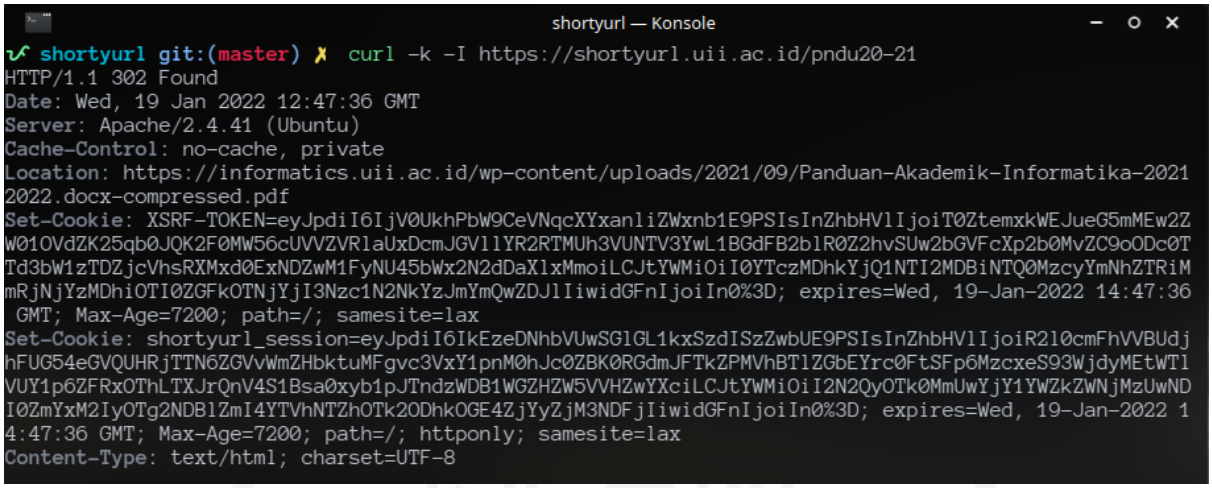
Gambar 4.14 Hasil formulir mengubah data URL singkat

Selanjutnya merupakan hasil dari pustaka Short URL yang berhasil diimplementasi dapat dilihat pada berkas *composer.json* yang di dalamnya tertera *ashallendesign/short-url* beserta versi yang terpasang digambarkan pada Gambar 4.16 berikut:

```
{
    "require": {
        "ashallendesign/short-url": "^5.2",
        ...
    },
    ...
}
```

Gambar 4.15 Hasil pemasangan shorturl.

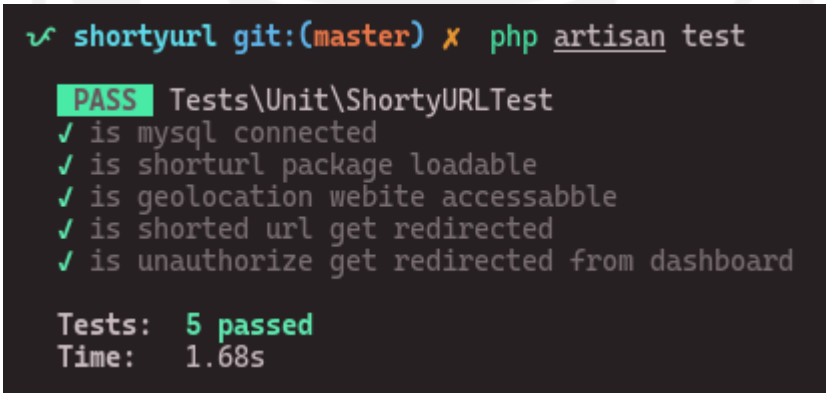
Redirect yang merupakan hasil dari Short URL dapat dilihat menggunakan perintah *curl* di *terminal* dan digambarkan pada Gambar 4.16 seperti berikut:



Gambar 4.16 Hasil *shortener*

4.1.3 White Box

PHPUnit yang dijalankan untuk melakukan uji coba menghasilkan *output* yang mengandung nama *unit test* serta nama *test* yang dijalankan, tangkapan layar yang dihasilkan dapat digambarkan pada Gambar 4.17 berikut:



Gambar 4.17 Hasil uji coba menggunakan PHPUnit

4.1.4 Black Box

Pengujian tanpa melibatkan program internal dan hanya mengandalkan *input* serta *output* yang akan menjadi parameter keberhasilan telah dilakukan, adapun hasil yang dapat diuraikan pada Tabel 4.1 adalah sebagai berikut:

Tabel 4.1 Hasil *black-box*.

Nama	Status	Keterangan
Log-in	Ok	*User* dapat melakukan *log-in*

Membuat tautan pendek	Ok	*User* dapat membuat tautan pendek
Membuka tautan pendek	Ok	Siapa pun dapat membuka tautan pendek
Mengubah data tautan pendek	Ok	*User* dapat mengubah data pada tautan pendek
Membuka dasbor	Ok	*User* dapat membuka dasbor
Membuka dasbor analisis	Ok	*User* dapat membuka dasbor analisis
Melihat rujukan tertinggi	Ok	*User* dapat melihat rujukan tertinggi
Melihat tautan pendek terpopuler	Ok	*User* dapat melihat tautan pendek terpopuler
Melihat jumlah pengunjung	Ok	*User* dapat melihat jumlah pengunjung
Melihat jumlah tautan pendek	Ok	*User* dapat melihat jumlah tautan pendek
Melihat lokasi pengunjung	Ok	*User* dapat melihat lokasi pengunjung
Log-out	Ok	*User* dapat melakukan *log-out*

4.2 Pembahasan

Prototipe yang dibuat menggunakan Figma berbasis aplikasi *website*, membuat pengerjaan dalam pengembangan *dashboard* semakin mudah karena adanya gambaran yang diberikan untuk pengembang. Tidak hanya gambaran melainkan informasi seperti tata letak, warna, dan *font* maupun detail lainnya, adapun hasil yang didapatkan dapat dilihat pada Gambar 3.

4.2.1 Implementasi SSO

Beragam cara untuk melakukan pemeriksaan terhadap jalannya Shibboleth SP, diantaranya menggunakan *shibd*, tautan, dan *systemctl* juga bisa. Dimulai dengan menekan tombol *login* pada halaman awal ShortyURL, transaksi SAML dapat dibaca oleh pihak ketiga yang dipasang pada aplikasi *Google Chrome* bernama *SAML Message Decoder*. Pada Gambar 4.7 terlihat detail informasi dari SP yang diajukan untuk IdP, adapun daftar dari *property* serta penjelasan yang ada pada *SAML Request* tersebut adalah sebagai berikut:

a. *AssertionConsumerServiceURL* berisikan tempat di mana nantinya IdP mengirim data balasan.
b. *Destination* berisikan *endpoint* pada IdP yang digunakan untuk melakukan SSO.
c. *ID* merupakan identitas dari SP.
d. *IssueInstant* merupakan data dari tanggal pengiriman *SAML Request*.

e. *ProtocolBinding* Merupakan informasi tentang protokol apa yang digunakan untuk melakukan *binding.*
f. *Version* merupakan versi dari SAML yang terjadi.

Setelah *SAML Request* berhasil dilaksanakan dan proses *authentication* berhasil dipenuhi, IdP akan memberikan *SAML Response* setelahnya, adapun beberapa penjelasan dari *property* penting yang akan dijelaskan sebagai berikut:

a. *Destination* merupakan *endpoint* yang disediakan SP untuk IdP.
b. *InResponseTo* merupakan identitas dari SP.
c. *IssueInstant* merupakan data dari tanggal pengiriman *SAML Response*.

Dengan pertukaran SAML yang berhasil dilaksanakan, informasi dari IdP dikemas melalui *attribute* kini sudah bisa digunakan untuk kepentingan layanan ShortyURL, seperti penentuan halaman yang boleh digunakan.

4.2.2 Implementasi Shortener

Setelah *login* berhasil dilakukan, *user* dapat membuat URL pendek dengan menekan tombol *create* pada *navbar* sebagai pemicu *pop-up* yang berisikan formulir di dalamnya. Isi dari formulir tersebut akan di teruskan hingga *ShortController* setelah menekan tombol *Save changes*. Setelah data berhasil ditangkap oleh *ShortController* maka metode *store* akan berjalan, adapun daftar *event* yang terjadi di dalamnya adalah:

a. Instansiasi *builder* Short URL.
b. Memeriksa apakah data URL singkat dari formulir sesuai dengan ketentuan, URL panjang terisi, dan *title* terisi.
c. Memeriksa apakah data URL singkat telah ditetapkan sebelumnya.
d. Menambahkan data ke dalam *database* merupakan tahapan terakhir sebelum halaman dikembalikan pada halaman pengisian formulir beserta pemberitahuan apakah gagal atau berhasil.

Selanjutnya merupakan *pop-up edit* yang dapat dipicu melalui tombol *edit* pada bagian detail, kurang lebih sama seperti pembuatan URL singkat baru, namun di sini URL singkat yang telah di tetapkan hanya boleh diganti apabila belum dibuka atau belum ada pengunjung sama sekali. Acara yang terkandung pada metode *update* dalam *ShortController* juga mirip seperti metode *create*, namun yang berbeda terletak pada bagian perubahan *database* yang di mana pada metode *create* adalah menambahkan bukan melakukan perubahan.

Ketika pengunjung membuka URL yang disingkat, *event* dari pustaka *Short URL* berjalan melakukan pencatatan. Setelah pencatatan berhasil, pengunjung langsung dialihkan pada halaman yang dituju sebenarnya. CURL di sini bertindak sebagai pengunjung, dapat dilihat ketika pengunjung menuju *https://shortyurl.uii.ac.id/pndu20-21* dibalas dengan *response code* 302 yaitu *found* dan *Location* merupakan halaman yang akan dituju oleh pengunjung.

4.2.3 Pengujian

Dalam pengujian dengan metode *white box* seluruh uji coba dibantu oleh pihak ketiga bernama PHPUnit yang telah diintegrasikan dengan *framework* Laravel, pengujian ini dilakukan untuk mengindikasi kesalahan yang terdapat pada saat pengembangan. Untuk pengujian *black box* yang dilakukan berfokus pada fungsionalitas *end-user* ketika menjelajahi ShortyURL dan hasil pada pengujian tersebut menunjukkan bahwa seluruh fungsionalitas berjalan dengan semestinya.

BAB V
SIMPULAN DAN SARAN

5.1 Simpulan

ShortyURL berhasil memanfaatkan layanan sistem SSO yang menggunakan SAML dengan cara mengintegrasikan Shibboleth IdP, dan Shibboleth SP yang telah diintegrasikan juga dengan *framework* Laravel sebagai antarmuka *dashboard*. Layanan pendukung UII berhasil dilengkapi dengan cara membuat aplikasi ShortyURL guna mempersingkat tautan berbasis website dan juga mengurangi pemakaian layanan pihak ketiga yaitu bit.ly dan masuk akal untuk diingat juga dibagikan, tautan yang dipendekkan juga tersedia dalam bentuk QR Code. Dengan cara menghadirkan data untuk kepentingan analisa dalam bentuk *dashboard*, ShortyURL berhasil melengkapi layanan yang dibutuhkan serta membantu dalam pengambilan keputusan terhadap platform apa yang akan digunakan nantinya untuk memperkuat *branding*, di dalam halaman yang berisikan kesimpulan dari keseluruhan data yang bisa digunakan untuk menentukan tempat penyebaran tautan pendek yang tepat, adapun beberapa yang tersedia adalah sebagai berikut:

a. *Top Referrer*.
b. *Top Link*.
c. *Top Visitor*.
d. Total tautan pendek.

Daftar tersebut merupakan sebagian data yang terdapat pada halaman *analytic* dan hanya yang memiliki izin yang boleh masuk, untuk pengguna biasa bisa menikmati data tersebut tetapi tidak seutuhnya seperti pada halaman *analytic* yang datanya diambil berdasarkan kalkulasi keseluruhan pengguna, melainkan data yang diambil terbatas untuk lingkup pengguna itu sendiri.

5.2 Saran

Untuk saran selanjutnya gunakan versi terbaru dari produk Shibboleth Consortium yang sudah mencapai versi 4 pada saat ShortyURL ini dikembangkan. Gunakan fitur *Queued Event Listener* yang dihadirkan oleh Laravel ketika memeriksa *geolocation* agar pengunjung tidak menunggu terlalu lama ketika membuka tautan singkat.

DAFTAR PUSTAKA

Antoniades, D., Polakis, I., Kontaxis, G., Athanasopoulos, E., Ioannidis, S., Markatos, E. P., & Karagiannis, T. (2011). Web: The web of short URLs. *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, 715–724. https://doi.org/10.1145/1963405.1963505

Choi, D., Han, J., Chun, S., Rappos, E., Robert, S., & Kwon, T. T. (2018). Bit.ly/practice: Uncovering content publishing and sharing through URL shortening services. *Telematics and Informatics*, *35*(5), 1310–1323. https://doi.org/10.1016/j.tele.2018.03.003

Dockins, K. (2017). Design Patterns in PHP and Laravel. In *Design Patterns in PHP and Laravel*. https://doi.org/10.1007/978-1-4842-2451-9

Doupé, A., Boe, B., Kruegel, C., & Vigna, G. (2011). Fear the EAR: Discovering and mitigating execution after redirect vulnerabilities. *Proceedings of the ACM Conference on Computer and Communications Security*, 251–261. https://doi.org/10.1145/2046707.2046736

Graves, R., & Vandenbrink, R. (2014). Implementing a Shibboleth SSO Infrastructure. *SANS Reading Room*, 1–36.

Kingsnorth, S. (2016). *Digital Marketing Strategy*.

Kotler, P., Kartajaya, H., & Setiawan, I. (2017). *Marketing 4.0: Moving From Traditional to Digital*.

Kukic, A. (2011). *The Definitive Guide to Single Sign-On (SSO)*. *August*, 74. https://resources.auth0.com/definitive-guide-to-single-sign-on/

Laravel LLC. (2022). *Directory Structure*. https://laravel.com/docs/8.x/structure

Liu, G., Gao, X., & Wang, H. (2021). An investigation of identity-account inconsistency in single sign-on. *The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021*, 105–117. https://doi.org/10.1145/3442381.3450085

Lockhart, H., Hardjono, C. T., & Cantor, C. S. (2012). *OASIS Security Services (SAML) TC*. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

Machek, Z. (2014). *PHPUnit Essentials*.

Manik, E., & Sidharta, I. (2017). The impact of academic service quality on student satisfaction. *Mpra*, *March*(80878), 1–7.

Morgan, R. L. B., Cantor, S., Carmody, S., Hoehn, W., & Klingenstein, K. (2004). Federated

Security: The Shibboleth Approach. *EDUCAUSE Quarterly*, *27*(4), 12–17.

Muktar, C. (2020). *Bitly Enterprise Pricing (2022)*. https://linklyhq.com/blog/bitly-enterprise-pricing-2020

OASIS. (2008). Security Assertion Markup Language (SAML) V2 . 0 Technical Overview. *OASIS Security Service TC*.

Pence, H. E. (2014). What is Big Data and Why is it Important? *Journal of Educational Technology Systems*, *43*(2), 159–171. https://doi.org/10.2190/et.43.2.d

Radha, V., & Reddy, D. H. (2012). A Survey on Single Sign-On Techniques. *Procedia Technology*, *4*, 134–139. https://doi.org/10.1016/j.protcy.2012.05.019

Scavo, T., & Cantor, S. (2005). *Shibboleth Architecture Technical Overview*. http://www.internet2.edu/products-services/trust-identity-middleware/shibboleth/

Shaharanee, I. N. M., Jamil, J. M., & Rodzi, A. S. S. M. (2016). The application of Google Classroom as a tool for teaching and learning. *Journal of Telecommunication, Electronic and Computer Engineering*, *8*(10), 5–8.

Shibboleth Consortium. (2009). *Shibboleth Concepts*. Shibboleth Consortium. https://shibboleth.atlassian.net/wiki/spaces/CONCEPT/pages/928645504/Home

Simco, G. (2001). The internet 2 middleware initiative. *Internet and Higher Education*, *4*(1), 77–84. https://doi.org/10.1016/S1096-7516(01)00049-5

Singh, R., & Awasthi, S. (2020). Updated Comparative Analysis on Video Conferencing Platforms- Zoom , Google Meet , Microsoft Teams , WebEx Teams and GoToMeetings. *Easy Chair: The World for Scientist*, 1–9.

Spicer, D. Z., Deblois, P. B., & Issues, C. (2004). Fifth Annual EDUCAUSE Survey Identifies Current IT Issues. *EDUCAUSE*, *2*, 8–22.

Stauffer, M. (2019). Laravel: Up and Running A Framework for Building Modern PHP Apps. In *O'Reilly* (Vol. 44, Issue 8).

Wilson, Y., & Hingnikar, A. (2019). Solving Identity Management in Modern Applications. In *Solving Identity Management in Modern Applications*. Apress. https://doi.org/10.1007/978-1-4842-5095-2