Jurnal Elektronik Ilmu Komputer Udayana p-ISSN: 2301-5373
Volume 8, No 4. May 2020 e-ISSN: 2654-5101

Perancangan dan Implementasi Sistem Manajemen Proyek Perangkat Lunak Menggunakan Teknologi *Single Page Application*

Aditya Wikardiyan[1], I Made Widiartha[2], Luh Arida Ayu Rahning Putri[3]

[1,2,3]Teknik Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Udayana
Bukit Jimbaran, Badung, Bali, Indonesia

[1]adityawikardiyan@gmail.com
[2]madewidiartha@unud.ac.id
[3]rahningputri@unud.ac.id

Abstrak

Sistem Manajemen Proyek Perangkat Lunak (SMPPL) dibangun berbasis website bertujuan untuk mendukung pengerjaan proyek secara berkolaborasi agar komunikasi antar user dengan anggota tim dapat berjalan secara efisien dan realtime. Sistem ini menerapkan teknologi Single Page Application (SPA). Penerapan Single Page Application (SPA) pada sistem dapat menghemat penggunaan bandwidth dan meningkatkan kecepatan memuat website karena hanya mengambil bagian-bagian tertentu sesuai kebutuhan dari setiap halaman website tanpa memuat keseluruhan resource saat adanya interaksi berpindah halaman yang dilakukan user. Berdasarkan hasil pengujian penggunaan bandwidth dari lima halaman menu sistem, penerapan SPA pada SMPPL dapat membantu menghemat penggunaan bandwidth hingga 776.4KB dari 830KB total (93.54%).

Kata Kunci : *Single Page Application, website, kerja kelompok, Sistem Manajemen Proyek Perangkat Lunak, SMPPL*

1. Pendahuluan

Pekerjaan secara berkelompok memerlukan koordinasi yang tepat, sehingga tujuan dari pekerjaan dapat tercapai secara maksimal dan sesuai dengan yang diharapkan oleh setiap anggota. Keterlibatan teknologi informasi dapat memberi kemudahan bagi seluruh anggota dalam sebuah kelompok kerja dengan lebih efisien dan efektif, tetapi meskipun begitu *platform* yang tersedia menjadi tersebar dan tidak terpusat pada satu *platform* saja dikarenakan sistem yang telah ada belum mengandung komunikasi dari sesama *user*, yang menyebabkan saat *user* ingin berkomunikasi dengan anggota tim harus menggunakan aplikasi komunikasi lain yang menjadikan hal ini kurang efisien.

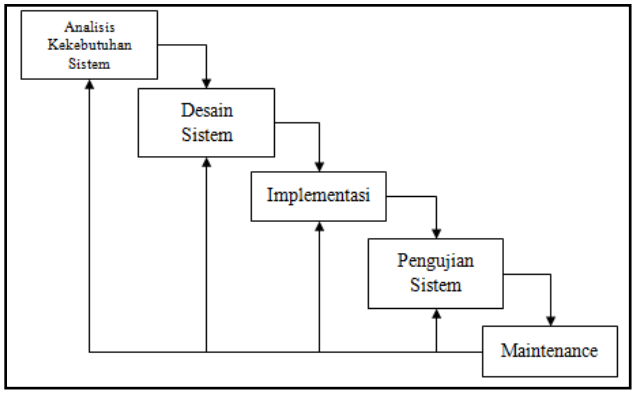
Kurangnya efisiensi dalam mengerjakan suatu pekerjaan tim akan berdampak pada proses pengerjaan. Dalam penelitian yang dilakukan oleh Paramita [1] dikatakan bahwa salah satu faktor umum penyebab kegagalan dari suatu proyek TI adalah kurangnya komunikasi dan kolaborasi antara manajer proyek, tim proyek san semua pihak yang terlibat. Menyelesaikan masalah dari hal-hal tersebut maka diperlukan Sistem Manajemen Proyek Perangkat Lunak (SMPPL) yang sifatnya *online* berbasis *website* sehingga dapat dengan mudah diakses dari mana saja dan mendukung komunikasi antar *user* secara efisien. Selain kemudahan komunikasi, akses data secara *realtime* juga dibutuhkan untuk membantu mempercepat proses penyelesaian proyek. Salah satu bentuk dukungan SMPPL untuk mempercepat kegiatan penyelesaian proyek adalah dengan menerapkan teknologi *Single Page Application*.

Single Page Application (SPA) adalah istilah untuk aplikasi berbasis *website*, yang menggunakan satu halaman *web* saja sebagai tampilan dari aplikasinya. Semua interaksi atau pun penyajian data tidak akan membuat halaman secara utuh dimuat ulang, tetapi hanya bagian-bagian tertentu saja yang di*update* dari *server* atau dari hasil proses aplikasi di sisi *client*. Dengan memanfaatkan AJaX dan JavaScript maka permintaan atau *request* dari sisi *client* ke *server* akan diproses dibalik layar (*background process*), lalu *server* akan merespon dengan komponen-komponen *website* apa yang dibutuhkan client untuk

ditampilkan [2]. Keuntungan dari penggunaan teknologi ini menjadikan halaman web yang dibuat menjadi lebih ringan dan lebih cepat ketika diakses disbanding *web conventional*. Hal ini dikarenakan *Website conventional* memiliki masalah pada *user* dan *bandwidth* yang besar saat *user* melakukan akses ataupun berpindah halaman, sehingga seringkali kecepatan memuat *website* menjadi lambat. Hal ini terjadi karena pengguna akan memuat berkas dan *script* (CSS atau JavaScript) serta konten yang terkandung di dalamnya dari awal setiap terjadi perpindahan halaman. Penggunaan teknologi *Single Page Application* pada *website* sudah menjadi sangat penting, karena dengan kelebihan yang diberikan akan memberikan dampak positif baik dari sisi server dalam mengoptimalkan proses ataupun sisi pengguna untuk menghemat *bandwidth*.

2. Metode Penelitian

Sistem ini dibangun berbasis website dengan menerapkan teknologi *Single Page Application* (SPA). Penelitian ini dikembangkan dengan menggunakan metode *Waterfall*. Tahapan dari metode ini dimulai dari tahapan Analisis Kebutuhan Sistem, Desain Sistem, Implementasi, Pengujian Sistem dan *Maintenance*.



Gambar 1 Metode Waterfall

2.1 Analisa Kebutuhan Sistem

Pada tahap analisis kebutuhan sistem dilakukan proses pencarian informasi dan data mengenai kebutuhan sistem yang akan dibangun. Pencarian informasi dilakukan dengan metode observasi dan wawancara dalam bentuk kuisioner. Kuisioner ini diisi oleh calon *user* yang berasal dari lingkungan *Start-up* (perusahaan rintisan) IT berkantor di DILo (*Digital Innovation Lounge*) Denpasar beralamat di Jalan Wage Rudolf Supratman No. 302, Tohpati, Denpasar. Hasil dari pencarian informasi diperoleh fitur yang dibutuhkan oleh calon *user* digambarkan pada tabel 1.

Tabel 1 Kebutuhan *User* Berdasarkan Hasil Kuisioner

No	Kebutuhan
1	*User* membutuhkan sistem manajemen proyek perangkat lunak yang dapat memudahkan *user* untuk melakukan pertukaran berkas (*file)* dan folder. Serta sekaligus memberikan versi pada berkas secara otomatis.
2	*User* membutuhkan fitur yang memudahkan *user* untuk berkomunikasi ketika berkolaborasi mengerjakan proyek perangkat lunak, dalam hal ini fitur tersebut adalah fitur *chatting*.

Dari tabel di atas selanjutnya akan dikembangkan ke dalam kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional adalah kebutuhan yang mendefinisikan apa yang sistem harus kerjakan, fungsi yang disediakan sebuah sistem, bagaimana sistem bereaksi terhadap input dan situasi tertentu [3]. Adapun kebutuhan fungsional sistem dapat dilihat pada tabel 2.

Tabel 2 Kebutuhan Fungsional

Kode	Deskripsi Kebutuhan	Target *User*
KF1	Sistem menyediakan fasilitas registrasi bagi *user* baru.	*User*
KF2	Sistem menyediakan fasilitas *login* dengan cara memasukkan *username* dan *password.*	*User*
KF3	Sistem dapat mempermudah *user* dalam melakukan pengaturan *Profile*, dengan rincian berikut: a. Dapat mengganti dan menghapus *avatar* (foto profil) b. *User* sistem dapat menambahkan dan mengedit kemampuan (*skill)* dan data diri. c. *User* dapat mengubah nama lengkap, *e-mail* dan *password*.	*User*
KF4	Sistem menyediakan fitur cari teman agar *user* dapat menambahkan teman atau relasi.	*User*
KF5	Dalam sistem ini, *user* dapat membuat dan menambah *project group* baru untuk mengerjakan proyek perangkat lunak secara berkolaborasi, dengan langkah berikut: a. Mengisi nama proyek b. Menambahkan anggota tim ke dalam *project.* c. Mengisi jumlah memori untuk penyimpanan file ataupun folder yang akan digunakan dalam *project*.	*User*
KF6	Ketua tim sebuah proyek dapat membuat divisi dalam *project group* untuk memudahkan pembagian tugas kepada anggota tim yang tepat terkait dengan proyek tersebut sesuai tugas dan keahlian masing-masing anggota.	*User*
KF7	*User* dapat menggunggah berkas ataupun folder ke dalam *project group* yang dibuat.	*User*
KF8	*User* dapat melihat perkembangan *project* atau ruang divisi, dengan rincian sebagai berikut: a. *User* yang tergabung dalam sebuah *project* dapat menentukan persentase perkembangan *project* atau ruang divisi. b. Administrator Proyek dapat menentukan batas waktu selesainya sebuah proyek.	*User*
KF9	Sistem menyediakan menu *Dashboard Analytic* yang berisi statistic total *user*, banyaknya *project group*, dan jumlah pengunjung web sistem.	Admin
KF10	Sistem dapat menentukan estimasi biaya proyek perangkat lunak menggunakan UCP (*Use Case Points*), dimana fitur ini dapat diakses oleh ketua tim setiap *project*.	*User*
KF11	Sistem dapat menentukan Versi sebuah *file source code* ketika file tersebut diunggah, fitur ini akan memudahkan *user* jika ingin melakukan *rollback* ke versi *file source code* sebelumnya.	*User*
KF12	Untuk melakukan komunikasi antar *user* sistem menyediakan fitur Grup *chat* dan *private chat*.	*User*
KF13	Pada sistem ini disediakan fitur *Manage User*, yang berfungsi untuk mengatur *user* yang terdaftar di sistem, seperti menangguhkan (*suspend*) *user*, memblokir *user*, ataupun menghapus *user*.	Admin
KF14	Sistem menyediakan fasilitas *logout*	*User*

Sedangkan kebutuhan non fungsional merupakan kebutuhan yang tidak secara langsung berkaitan dengan layanan atau fungsi spesifik pada sistem kepada penggunanya [3]. Contohnya seperti keandalan sistem ataupun kecepatan proses dari sistem. Kebutuhan non fungsional pada sistem ini adalah sebagai berikut:

a. *User-Friendly*

Sistem yang dibangun dapat menampilkan antarmuka yang mudah dipahami oleh *user*, sehingga *user* dapat dengan mudah mengerti fungsi-fungsi dari fitur yang disediakan oleh sistem dan memberikan kenyamanan kepada *user* saat menggunakan sistem.

b. Kecepatan

Dengan memanfaatkan teknologi SPA (*Single Page Application),* sistem yang dibangun diharapkan dapat memberikan kecepatan proses.

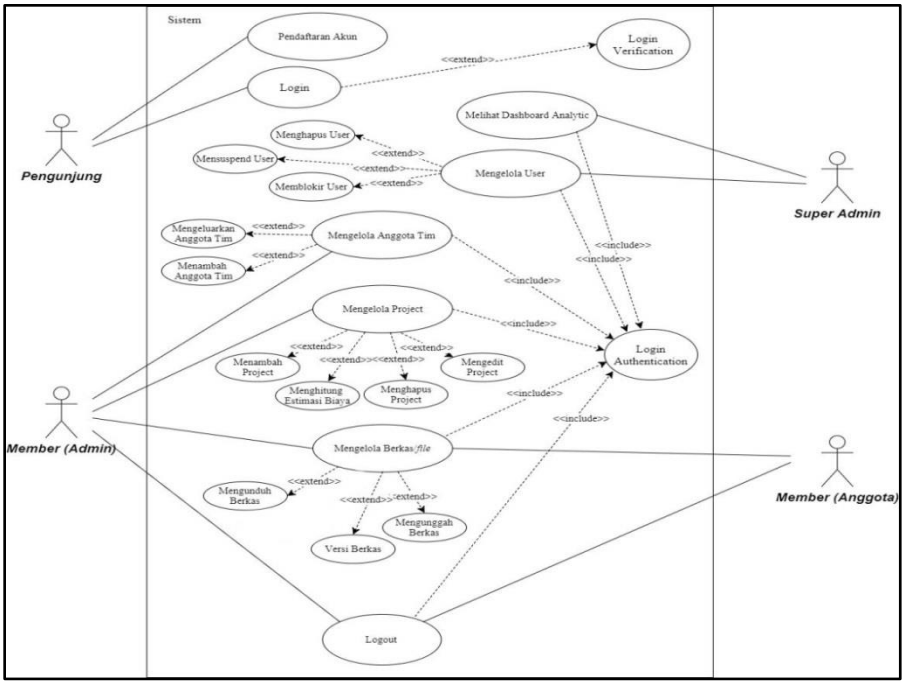
c. Keamanan

Sistem yang dibangun dapat menjaga keamanan akses terhadap data *user*, sehingga setiap data proyek yang ada pada sistem hanya dapat diakses oleh ketua dan anggota tim dari proyek tersebut.

2.2 Desain Sistem

a. *Use Case Diagram*

Use Case Diagram menggambarkan pada sistem ini terdiri fungsionalitas yang diharapkan dari suatu sistem. Use case diagram merepresentasikan sebuah interaksi antara actor dengan sistem, seperti yang digambarkan dalam Gambar 2 terkait interaksi antara Pengunjung, Member, dan Super Admin dengan sistem.

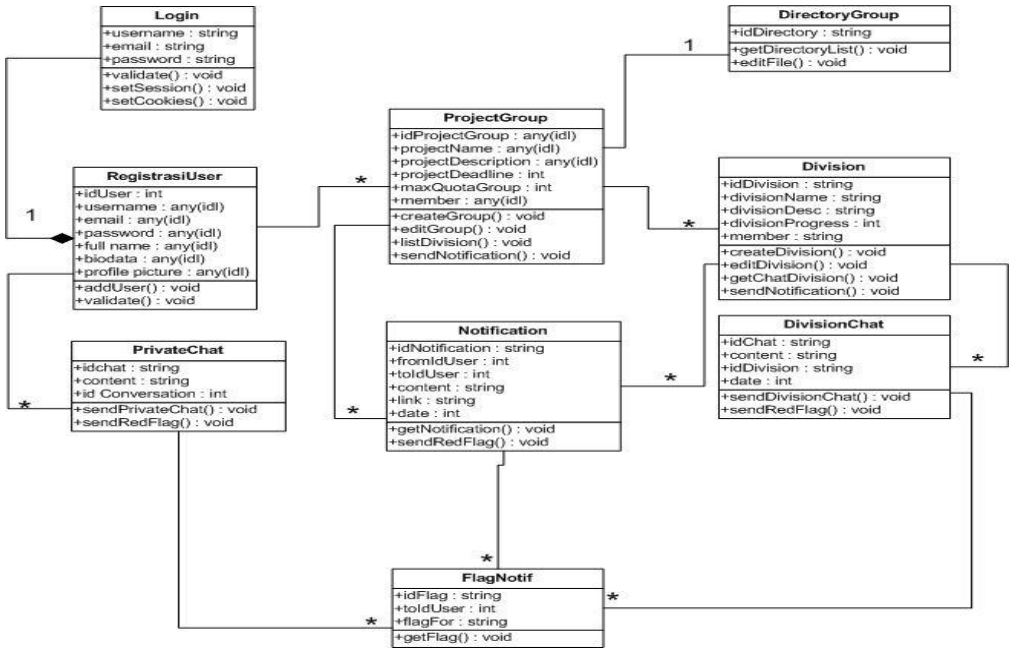


Gambar 2 *Use Case Diagram*

b. *Class Diagram*

Class Diagram dari Sistem Manajemen Proyek Perangkat Lunak memiliki beberapa *class* yang terdiri atas Login, RegistrasiUser, PrivateChat, DirectoryGroup, ProjectGroup, Notification, Division, DivisionChat, dan FlagNotif. Masing – masing *class* memiliki atribut dan *method* yang nantinya akan diimplementasikan ke dalam program. Adapun penjelasan *class diagram* terdapat pada tabel 3.

p-ISSN: 2301-5373
e-ISSN: 2654-5101



Gambar 3 *Class Diagram*

Tabel 3 Hubungan *Class Diagram* dengan *Use Case*

Kelas	*Use Case*	Fungsi
Login	*Login*	Menangani verifikasi *login user*
	Login Verification	
Registrasi *User*	Pendaftaran Akun	Menangani pendaftaran akun *user*
Project Group	Mengelola *project*	Menangani manajemen *project group* yang akan dilakukan *user*
	Menambah *project*	
	Menghitung estimasi biaya	
	Menghapus *project*	
	Mengedit *project*	
Directory Group	Mengelola berkas	Menangani pengelolaan *file* dan *folder* dalam *project group*
	Mengunduh berkas	
	Versi berkas	
	Mengunggah berkas	
Division	Mengelola anggota tim	Mengangani manajemen divisi pada *project group*
	Mengelola *project*	
	Mengeluarkan dan menambah anggota tim	

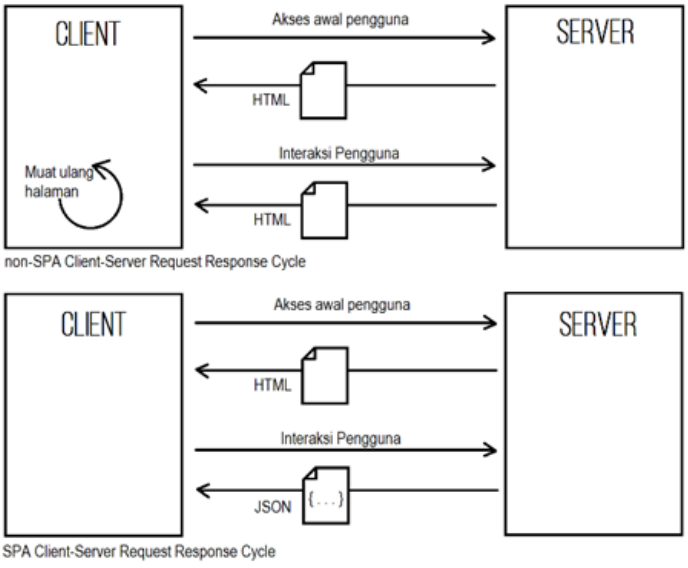
c. Penerapan Single Page Application

Single Page Application (SPA) adalah istilah untuk aplikasi berbasis *website*, yang menggunakan satu halaman web saja sebagai tampilan dari aplikasinya. Semua interaksi atau pun penyajian data tidak akan membuat halaman secara utuh dimuat ulang, tetapi hanya bagian-bagian tertentu saja yang di*update* dari *server* atau dari hasil proses aplikasi di sisi *client*. Keuntungan dari penggunaan teknologi ini menjadikan halaman web yang dibuat menjadi lebih ringan dan lebih cepat ketika diakses [2].

Dengan memanfaatkan AJaX dan *JavaScript* maka permintaan atau *request* dari sisi *client* ke *server* akan diproses dibalik layar (*background process*), lalu *server* akan merespon dengan komponen-komponen website apa yang dibutuhkan *client* untuk ditampilkan. Perbedaan SPA dengan request konten menggunakan AJaX adalah pada saat pengguna melakukan interaksi untuk berpindah halaman maka SPA melakukan perubahan URL website pada browser, untuk itu SPA membutuhkan bantuan HTML5 History API untuk memungkinkan penyimpanan history alamat website dan keperluan mengubah-ubah halaman URL website untuk menyesuaikan dengan halaman yang saat ini dibuka oleh pengguna. Salah satu keuntungan dari mengubah URL website adalah agar tautan tersebut menjadi *shareable* atau mudah untuk dibagikan kepada pengguna lain. Berikut adalah beberapa point penting dalam SPA [2]:

1. *Individual Components* : Seluruh halaman web dibagi menjadi komponen yang lebih kecil dan saling berinteraksi.
2. *Replaced/Updated* : Komponen/Bagian dari halaman web digantikan/diperbarui dengan perubahan sesuai dengan permintaan pengguna.
3. *Refreshing/Reloading* : Halaman web tidak pernah disegarkan atau dimuat ulang, tetapi konten baru dimuat dibeberapa bagian sesuai dengan data baru.
4. *User Actions* : SPA berkewajiban menangani semua interaksi dari pengguna (tombol klik, masukan dari *keyboard*, dll) dengan cepat dan mengarah ke *user interface* yang sangat fluida (tidak menolak berubah bentuk).

SPA menyediakan cara yang lebih fleksibel dan elegan berkaitan dengan data. Menyegarkan (*refresh*) bagian tertentu atau bagian dari halaman tanpa memuat ulang seluruh halaman adalah tujuan utama layanan SPA, tetapi semua fleksibilitas ini membutuhkan *user interface* yang lebih interaktif dan hal ini menyebabkan *user experience* yang lebih baik [2]. *Gambar 5 menampilkan Client-Server Request Response Cycle* pada SPA dan non-SPA.



Gambar 4 *Client-Server Request Response Cycle*

Alur jika menggunakan SPA:

- Pengguna/klien mengakses halaman web (pengguna pertama kali mengakses halaman web).
- *Server* merespon berupa halaman HTML dengan konten eksternal berupa gambar, CSS, JavaScript, dll.
- Jika pengguna berinteraksi, hal ini akan menjadikan sisi klien meminta konten ke *server.*
- *Server* merespon dengan format JSON melalui AJaX yang mengandung hanya permintaan pengguna/klien tersebut.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

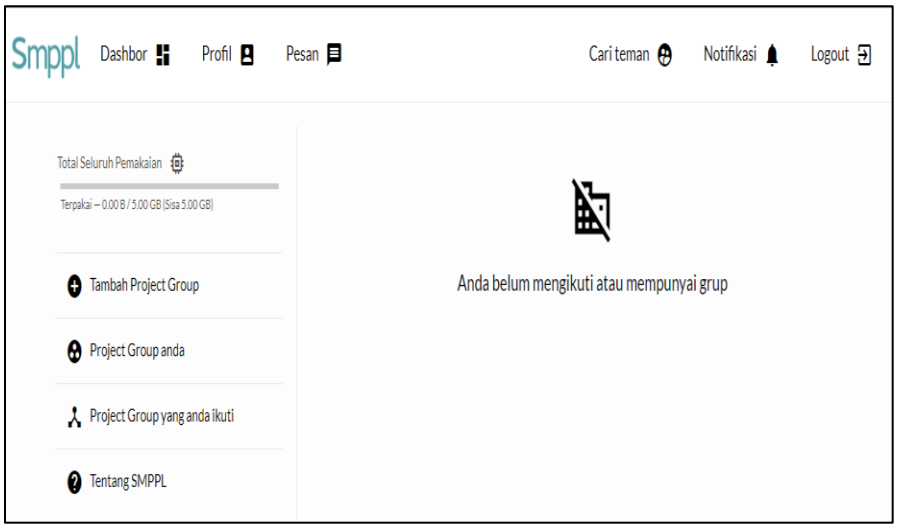
- Pada sisi klien tidak akan memuat ulang halaman dari awal, tetapi akan menggantikan/ memperbarui bagian-bagian tertentu saja sesuai permintaan pengguna/klien. Hal ini akan meminimalkan penggunaan bandwidth dan meningkatkan kecepatan akses.

Alur jika tidak menggunakan SPA,

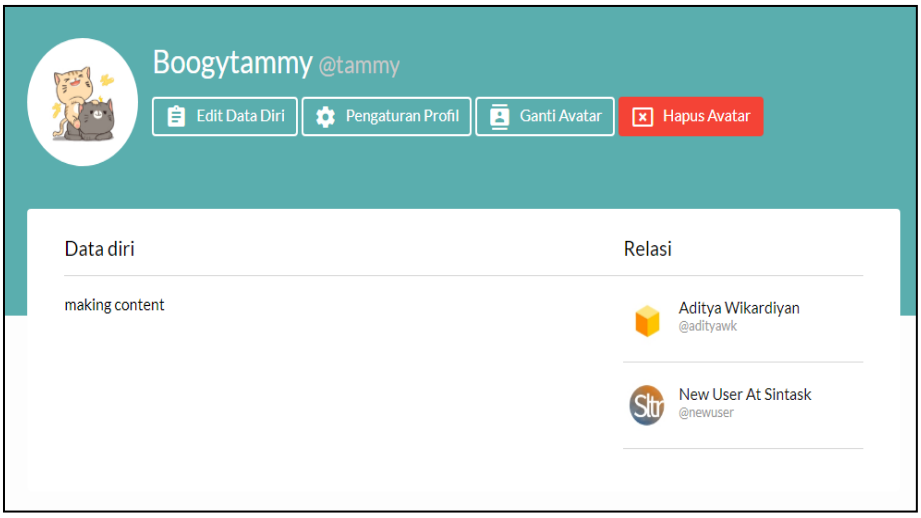
- Pengguna/klien mengakses halaman web (pengguna pertama kali mengakses halaman web).
- *Server* merespon berupa halaman HTML dengan konten eksternal berupa gambar, CSS, JavaScript, dll.
- Jika pengguna berinteraksi dengan mengklik atau masukan dari keyboard, hal ini akan menjadikan sisi klien meminta konten ke *server* dan merespon dengan memberikan seluruh konten pada halaman HTML baru.
- Pada sisi pengguna/klien akan memuat ulang halaman web, sehingga mengakibatkan seluruh konten dimuat kembali dari awal. Hal ini mengakibatkan penggunaan bandwidth menjadi lebih besar dibandingkan menggunakan SPA.

3. Hasil dan Pembahasan

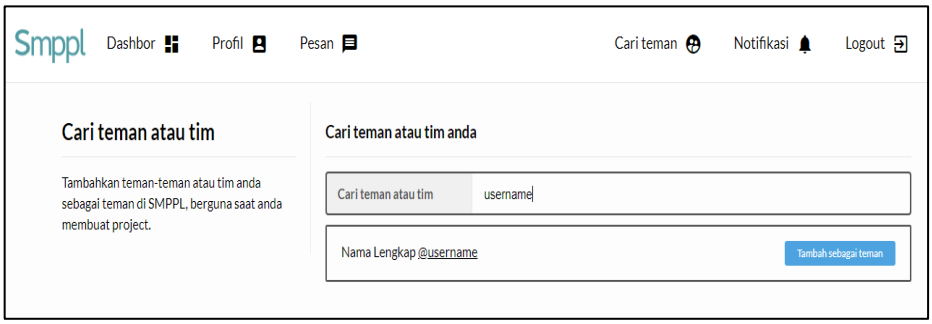
3.1 Implementasi Antarmuka



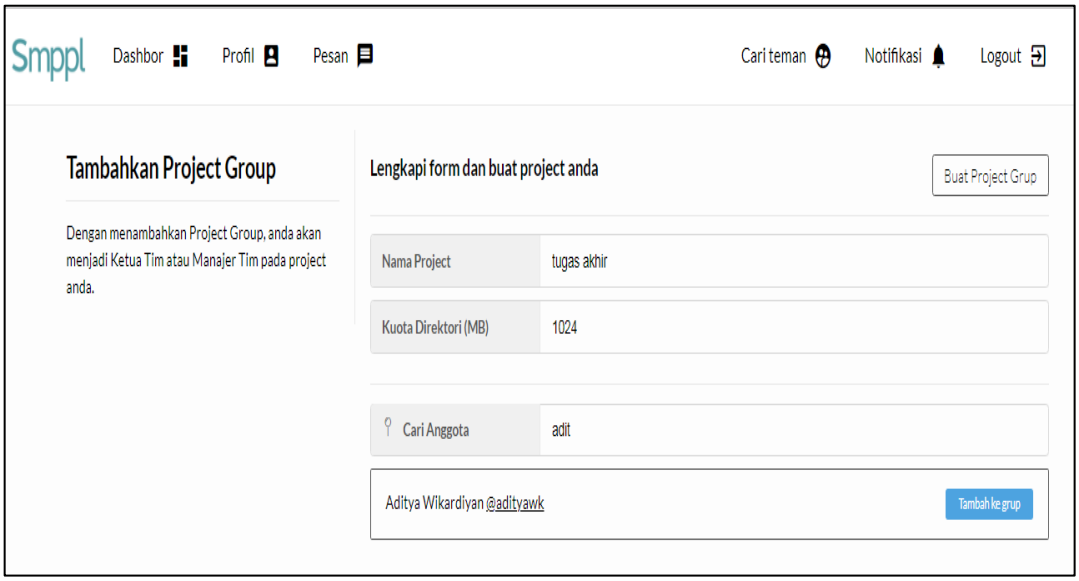
Gambar 5 Tampilan awal *atau dashbor*



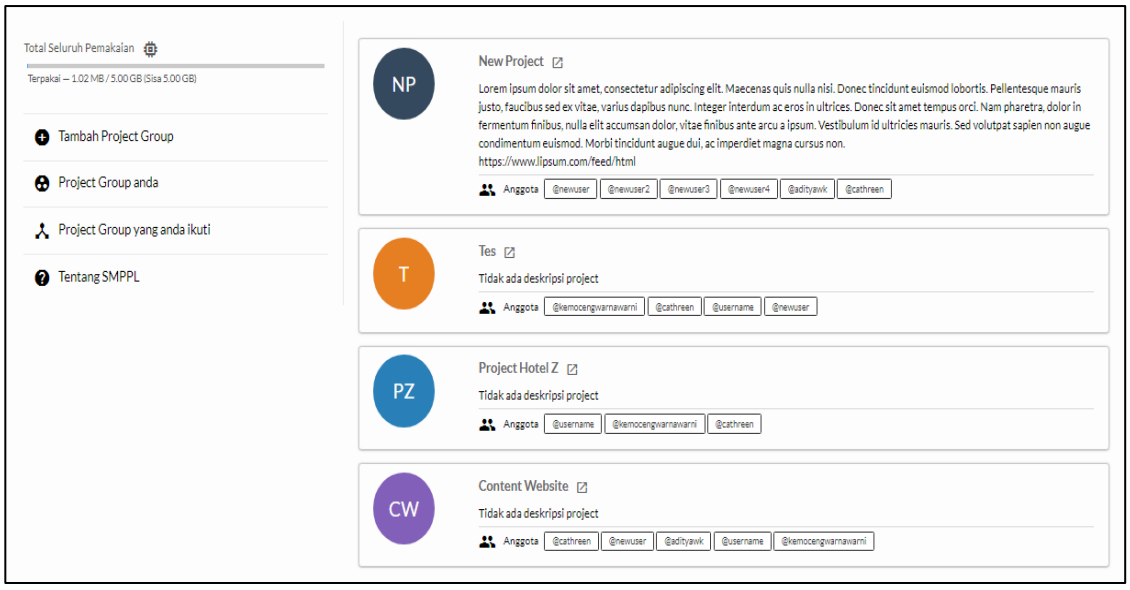
Gambar 6 Profil



Gambar 7 Cari Teman



Gambar 8 Menu Tambah *Project*



Gambar 9 Menu Daftar *Project* Saya

Gambar 5 hingga Gambar 9 merupakan implementasi antarmuka dari Sistem Manajemen Proyek Perangkat Lunak. Adapun implementasi sistem yang ditampilkan pada bagian ini antara lain, tampilan awal atau *dashbor*, tampilan profil *user*, menu cari teman untuk menemukan teman atau calon anggota tim, menu tambahkan *project* dan terakhir menu daftar *project* saya.

3.2 Pengujian dan Evaluasi

a. Hasil Penggunaan *Bandwidth*

Penerapan *Single Page Application* pada sisi *client* tidak akan memuat ulang halaman dari awal, tetapi akan menggantikan/memperbarui bagian – bagian tertentu saja sesuai permintaan pengguna atau *client*. Hal ini akan meminimalkan penggunaan *bandwidth* dan meningkatkan kecepatan akses. Berbeda dengan sistem tanpa SPA, pada sisi *client* akan memuat ulang halaman web, sehingga mengakibatkan seluruh konten dimuat ulang kembali dari awal. Hal ini mengakibatkan penggunaan *bandwidth* menjadi lebih besar dibandingkan menggunakan SPA serta menyebabkan memperlambat kecepatan memuat *website*.

Berdasarkan penjabaran di atas, berikut ini merupakan hasil penggunaan *bandwidth* dari SMPPL dengan SPA maupun SMPPL tanpa penerapan SPA. Terdapat 5 buah halaman utama yang digunakan untuk pengujian hasil penggunaan *bandwith,* yaitu Dashbor, Profil, Cari Teman, Tambah Project dan Daftar Project Saya.

Tabel 4 Hasil Pengujian Penggunaan *Bandwidth*

Nama halaman	SPA	TANPA SPA
Dasbor	11.7 KB	158 KB
Profil	11.2 KB	205 KB
Cari Teman	8.6 KB	155 KB
Tambah Project	10.3 KB	156 KB
Daftar Project Saya	11.8 KB	156 KB

Dari tabel di atas diperoleh kesimpulan bahwa penggunaan SPA dapat menghemat penggunaan *bandwidth* hingga kurang lebih sebesar 776.4KB dari 830KB atau sekitar 93.54%

b. Stress Testing

Pengujian stress pada sistem dilakukan mengetahui waktu tunggu (*loading time*) untuk menjalankan suatu proses yang ada pada sistem yang dibangun Berikut ini merupakan hasil pengujian *stress testing* sistem dengan menggunakan Apache JMeter. Dimana skenarionya adalah *loop* yang digunakan sebanyak 5, *threads* yang digunakan dari 100 hingga 1000 serta *ramp up* sebanyak 100.

Threads mereprentasikan jumlah *virtual user* yang digunakan dalam pengujian. *Loop* berisi jumlah pengulangan pengaksesan website dalam pengujian yang dilakukan oleh seorang user. *Ramp Up* merupakan waktu yang diperlukan JMeter untuk memulai eksekusi dari setiap *thread/user* [7].

Tabel 5 Hasil *Stress Testing*

No	Threads/User	Average (dalam satuan ms)	Error (dalam satuan %)	Throughput (dalam satuan kb/s)
1	100	720	0	4.8
2	200	963	0	9.3
3	300	1780	0	12.17
4	400	1385	0	16.5
5	500	1149	0	23.6
6	600	4319	10.25	23.7
7	700	5514	12.7	27.3
8	800	10245	29.02	25.6
9	900	12328	34.99	25
10	1000	14326	29.37	29.37

Pada pengujian ini,semakin banyak *user* yang melakukan akses ke sistem, maka semakin lama waktu tunggu yang dihasilkan. Berdasarkan data yang didapat dari *Google Test My Site* yang dapat diakses di https://www.thinkwithgoogle.com/feature/testmysite, ketika *website* merespon permintaan *user* dengan waktu lebih dari 2,5 detik, maka *website* tersebut dikatakan memiliki respon yang lama [8].

Berdasarkan tabel 7, sistem mulai menghasilkan *error* serta waktu respon mulai lebih lambat ketika yang melakukan *request* ke sistem sebanyak 600 *user* dengan waktu respon yang dapat dilihat pada kolom *average* sekitar 4,3 detik. Sehingga dapat dikatakan bahwa sistem akan merespon permintaan *user* dengan cepat atau dibawah 2,5 detik, ketika *user* yang mengakses sistem berada diambang 500 *user*.

4. Kesimpulan

Sistem Manajemen Proyek Perangkat Lunak menerapkan penggunaan *Single Page Application* yang bertujuan agar proses memuat halaman dapat dilakukan lebih cepat, mengingat sistem ini ditujukan untuk melakukan pekerjaan proyek secara berkelompok atau kolaborasi. Penerapan *Single Page Application* pada sisi *client* tidak akan memuat ulang halaman dari awal, tetapi akan menggantikan/memperbarui bagian – bagian tertentu saja sesuai permintaan pengguna atau *client*. Hal ini akan meminimalkan penggunaan *bandwidth* dan meningkatkan kecepatan akses. Berdasarkan hasil pengujian penggunaan *bandwidth*, terhadap lima buah halaman menu yang terdapat pada sistem, SMPPL dengan SPA dapat menghemat penggunaan *bandwidth* hingga sebesar 776.4KB dari 830KB total tanpa SPA atau lebih menghemat *bandwidth* sekitar 93.54%.

Berdasarkan hasil pengujian *stress testing,* dengan adanya penerapan teknologi *Single Page Application* (SPA) pada sistem ini, hasil tes memberikan *average* atau rata – rata waktu respon selama 1,1 detik tanpa menghasilkan *error* ketika diakses oleh 500 *user*. Sedangkan jika diakses oleh 600 *user* ke atas, waktu respon yang dihasilkan semakin lama. Berdasarkan data yang didapat dari *Google Test My Site*, ketika *website* merespon permintaan *user* dengan waktu lebih dari 2,5 detik, maka *website* tersebut dikatakan memiliki respon yang lama. Maka dengan kata lain, sistem ini dapat memberikan respon yang cepat ketika *user* yang mengakses sebanyak 500 *user*.

Daftar Pustaka

[1]. Paramita, Dewi. 2015. “Rancang Bangun Sistem Informasi Kolaboratif Berbasis *Web* Untuk Manajemen Proyek Teknologi Informasi”. Jurnal Buana Informatika Vol. 6, No. 3.

[2]. Jadhav, Madhuri A., Sawant, Balkrishna R., Deshmukh, Anushree., 2015. “Single Page Application using AngularJS”**, International Journal of Computer Science and Information Technologies,** Vol.6, No 3, ISSN: 0975-9646.

[3]. Sommerville, I. (2011). *SOFTWARE ENGINEERING* (9 ed.). (M. Horton, M. Hirsch, M. Goldstein, C. Bell, & J. Holcomb, Eds.) USA: Pearson Education,Inc.

[4]. Apache Software Foundation. 2019. *Apache Jmeter User Manual Glossary.* https://jmeter.apache.org/usermanual/glossary.html. Diakses 27 April 2020.

[5]. Think With Google. 2017. *Understanding The Speed Overview*. https://www.thinkwithgoogle.com/feature/testmysite/faq/. Diakses 27 April 2020.