OTOMASI INFRASTRUCTURE AS CODE (IAC) DENGAN MODEL KOLABORASI CHATOPS MENGGUNAKAN HUBOT

Disusun Oleh:

N a m a : Wahyuni Puji Lestari

NIM : 14523106

PROGRAM STUDI TEKNIK INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2018

HALAMAN PENGESAHAN DOSEN PEMBIMBING

OTOMASI INFRASTRUCTURE AS CODE (IAC) DENGAN MODEL KOLABORASI CHATOPS MENGGUNAKAN HUBOT

TUGAS AKHIR

Disusun Oleh:

N a m a	: Wahyuni Puji Lestari
NIM	: 14523106

Yogyakarta, 10 Oktober 2018

Pembimbing,

(Ari Sujarwo S.Kom., MIT. (Honours))

HALAMAN PENGESAHAN DOSEN PENGUJI

OTOMASI INFRASTRUCTURE AS CODE (IAC) DENGAN MODEL KOLABORASI CHATOPS MENGGUNAKAN HUBOT

TUGAS AKHIR

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

Yogyakarta, 5 Oktober 2018

Tim Penguji

Ari Sujarwo S.Kom., MIT. (Honours))

Anggota 1

Hendrik, S.T., M.Eng.

Anggota 2

Hanson Prihantoro Putro, S.T., M.T.

Mengetahui,

Ketua Program Studi Teknik 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 : Wahyuni Puji Lestari

NIM : 14523106

Tugas akhir dengan judul:

OTOMASI INFRASTRUCTURE AS CODE (IAC) DENGAN MODEL KOLABORASI CHATOPS MENGGUNAKAN HUBOT

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian 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 resiko dan konsekuensi apapun.

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

Yogyakarta, 3 Oktober 2018

(Wahyuni Puji Lestari)

HALAMAN PERSEMBAHAN

Puji syukur atas segala nikmat dan karunia yang Allah SWT berikan.

Sholawat beserta salam kepada Nabi Muhammad SAW sebagai pemberi syafaat kepada seluruh umat manusia.

Kepada kedua orang tua penulis, Ibu Sulastri dan Bapak Sujiman.

Terimakasih atas doa, kasih sayang, serta dukungan yang diberikan sepenuh hati.

Adik tersayang, terima kasih selalu mendukung dan menjadi
penyemangat untuk lebih kuat.

Terima kasih untuk Bapak Ari Sujarwo S.Kom., MIT. (Honours) selaku dosen pembimbing yang selalu memberikan nasihan dan arahan selama pengerjaan skripsi.

Seluruh keluarga besar BSI Universitas Islam Indonesia
yang selalu memberi dukungan dan membantu penulis dalam penyelesaian sistem.

HALAMAN MOTO

"Sesungguhnya Allah tidak akan merubah nasib suatu kaum kecuali kaum itu sendiri yang mengubah nasibnya"
(Ar Rad : 11)

The greatest discovery of all time is that a person can change his future
by merely changing his attitude

-Oprah Winfrey-

Kegagalan juga menyenangkan, hiduplah dengan kepercayaan bahwa cobaan itu berguna untuk menempa diri sendiri.

-Jiraya-

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillahi Robbil 'Alamin, puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, karunia dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "OTOMASI INFRASTRUCTURE AS CODE (IAC) DENGAN MODEL KOLABORASI CHATOPS MENGGUNAKAN HUBOT".

Laporan ini disusun sebagai salah satu persyaratan dalam rangka menyelesaikan pendidikan pada jenjang Starta 1 di Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Tugas akhir ini dapat terselesaikan atas bantuan, dukungan, dan bimbingan yang diberikan dari berbagai pihak, maka dari itu penulis mengucapkan terimakasih kepada:

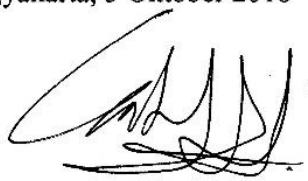
1. Allah SWT yang selalu melimpahkan rahmat, hidayah, serta memberikan kenudahan bagi penulis.
2. Kedua orang tua penulis, Ibu Sulastri dan Bapak Sujiman yang telah mendoakan dan memberikan dukungan moril maupun materil dam proses pembuatan tugas akhir ini, serta adik tersayang Abdul Ghofur yang selalu menjadi penyemangat bagi penulis.
3. Bapak Hendrik, S.T, M.Eng., selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Program Studi Teknik Informatika – Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Ari Sujarwo S.Kom., MIT. (Honours), selaku dosen pembimbing tugas akhir yang telah membagi ilmu dan dengan sabar memberikan waktunya untuk membimbing penulis dalam menyelesaikan tugas akhir.
6. Bapak dan Ibu dosen Jurusan Informatika yang telah membagi ilmu kepada penulis.
7. Tim DevOps 1 yaitu Mas Amien, Mas Agast, Mas Yos, Mas Hanafi dan Mbak Linda yang telah banyak membantu dalam proses pengerjaan tugas akhir penulis, serta seluruh teman-teman di Badan Sistem Informasi UII yang selalu memberi semangat bagi penulis.
8. Tim Dev 4 yaitu Mas Manggala, Mbak Gita, Mas Reza, Mas Fahmi, Mas Fikri serta Zatin, Haris, Arul, Mbak Aan, Mas Rofi yang sudah turut membantu penulis dalam pengujian sistem.

9. Kepada teman-teman Anak Sholeha tercinta yang sudah mendengarkan penulis selama ini dan menjadi penyemangat.
10. Kepada teman-teman Graha Asri Squad, Synaptic dan Magnifiqo yang selalu memberi semangat dan mendoakan penulis.
11. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah membantu, terimakasih atas bantuan dan do'anya.

Tugas akhir ini tidak lepas dari kekurangan maupun ketidaksempurnaan, oleh karena itu kritik dan saran yang membangun sangat penulis harapkan agar menjadi lebih baik lagi. Semoga tugas akhir ini dapat memberikan suatu manfaat bagi kita semua.Aamiin.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 3 Oktober 2018



(Wahyuni Puji Lestari)

SARI

Perkembangan teknologi informasi semakin pesat, hal ini juga mempengaruhi pengadaptasian teknologi dalam lingkungan pendidikan tinggi. Teknologi *cloud* dan virtualisasi yang diterapkan menjadi semakin kompleks dan sulit untuk dikelola. *Infrastructure as code* muncul menjadi solusi dalam permasalahan ini. Konsep teknologi DevOps kemudian terbentuk dalam pengelolaan infrastruktur, arsitektur maupun budaya kerja tim dalam pengembangan perangkat lunak di masa modern ini. Dalam DevOps banyak *tools* yang terlibat dan saling berkolaborasi, setiap *tools* memiliki peran tersendiri namun membutuhkan waktu dalam pengelolaannya. Tim DevOps memberikan perintah atau tugas secara berulang-ulang menjadi salah satu penyebab kurang optimalnya penggunaan tools tersebut. Selain itu Slack sebagai media chat yang digunakan oleh tim baru dimanfaatkan untuk diskusi percakapan sehari-hari, belum melibatkan tim untuk dapat mengakses collaboration tools melalui chat.

Chatbot dibangun untuk mengatasi masalah ini, mengefisensi waktu kerja serta mengotomasi perintah sehingga *workflow* kerja lebih efektif. Teknologi ChatOps dibangun dengan menempatkan bot untuk mengelola operasi teknis dan bisnis ke *chat* klien. ChatOps dibangun menggunakan Hubot yang dimodifikasi sesuai dengan skenario yang akan diterapkan di Gitlab. Skenario dirancang sesuai dengan hasil analisis yang dilakukan dalam tahap perancangan, kumpulan *rule* skenario ini ditulis dalam bahasa *coffescript.* Chatbot akan diterapkan ke tim DevOps di Badan Sistem Informasi Universitas Islam Indonesia.

Dari penelitian yang telah dilakukan, penulis berhasil membuat sistem ChatOps untuk mengelola Gitlab dalam perilisan aplikasi ke *production*. Sistem ChatOps secara umum sudah memenuhi faktor usabilitas atau *usable*. Sedangkan, dampak implementasi sistem otomasi ChatOps adalah membuat kinerja menjadi lebih baik, sistem ChatOps meningkatkan kerjasama tim untuk menyelesaikan masalah-masalah dalam perilisan. Sistem juga dinilai lebih efektif dari pada sistem lama karena menghemat waktu dalam penyelesaian masalah, sehingga membantu lebih produktif.

Kata kunci: DevOps, otomasi, Chatbot, Hubot, Gitlab

GLOSARIUM

Adapter	antarmuka layanan tempat hubot berjalan
Agile	pendekatan untuk pengembangan perangkat lunak di mana persyaratan dan solusi berevolusi melalui kolaboratif tim
Allowance	kelonggaran atau tunjangan waktu
Branch	garis pengembangan yang independen.
Build, rebuild	membangun atau membangun ulang proyek di Gitlab dengan menjalankan *job* dalam *pipeline*.
Cloud computing	gabungan pemanfaatan teknologi komputer dan pengembangan internet
Collaboration tools	alat, aplikasi, perangkat lunak yang digunakan untuk berkolaborasi dengan berbagai sistem seperti DevOps.
Container	wadah unit perangkat lunak standar di Docker
Downtime	jumlah waktu dimana suatu sistem tidak dapat beroperasi
Microservices	aplikasi terdistribusi yang semua modul-modulnya independen minimal dan berinteraksi melalui pesan.
Monolith	arsitektur tradisional dimana aplikasi perangkat lunak yang terdiri dari modul yang tidak independen
Open source	Perangkat lunak sumber terbuka, kode sumber aslinya dibuat dan tersedia secara bebas dan dapat didistribusikan kembali dan dimodifikasi.
Pipeline	saluran, sekelompok pekerjaan yang dijalankan secara bertahap
Regex	reguler expression, konstruksi bahasa dengan pola tertentu untuk kasus yang kompleks
Requirement	kualifikasi yang harus dimiliki untuk melakukan sesuatu
Rule	sekumpulan kode peraturan berdasarkan skenario Hubot
Tag	menandai penerapan dan rilis
Token	peranti keamanan yang dapat menghasilkan kode rahasia tertentu.
Trigger	pemicu yang digunakan untuk Gitlab agar dapat dijalankan.
Webhook	pemberitahuan acara sederhana melalui HTTP POST

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Banyak perguruan tinggi di Indonesia yang telah memanfaatkan teknologi informasi dalam pengelolaan kegiatan maupun proses administrasi akademik, keuangan dan kepegawaian (Nugroho, 2013). Salah satunya di Universitas Islam Indonesia, Perguruan tinggi ini memiliki badan khusus yang bertanggung jawab mengelola infrastruktur baik *perangkat lunak* maupun perangkat keras yaitu Badan Sistem Informasi (BSI). Selain itu pengadopsian teknologi untuk alat pembelajaran perguruan tinggi, juga diadaptasi oleh negara maju maupun berkembang, misalnya Inggris, Amerika serikat, Jerman, India dan Pakistan (Jenhani, 2011; Sarkar, 2012; Shaik & Khoja, 2011).

Nugroho (2013) menyebutkan bahwa pengolalan teknologi informasi di Indonesia masih kurang optimal, baik dalam hal teknis, infrastruktur serta konten yang di kembangkan. Kurangnya ruang untuk memenuhi permintaan yang tinggi, sumber daya terbatas, biaya administrasi tinggi dan kesulitan mengelola populasi pengguna yang semakin membesar adalah tantangan pengelolaan ICT di perguruan tinggi (Ya ’u Gital & Zambuk, 2011). Salah satu upaya untuk memperbaiki sistem ini adalah dengan mengadopsi teknologi *cloud computing* untuk infrastrukturnya (Jenhani, 2011; Makoza, 2015). Beberapa perguruan tinggi internasional sudah menggunakan *cloud computing* sebagai infrastrukturnya, contohnya Universitas Nairobi Harvard University dan University of Toronto (Sarkar, 2012; Shaik & Khoja, 2011).

Perkembangan teknologi *cloud* dan virtualisasi yang sangat cepat membuat pengelolaannya menjadi sulit, sehingga perlu teknologi manajemen infrastruktur IT baru yang dapat menanganinya (Morris, 2016). *Infrastructure as code* menjadi jawaban dalam permasalahan ini (Morris, 2016), dengan teknologi ini praktik otomasi semakin mudah dikembangkan selain itu *continuous integration* (CI) dan *continuous delivery* (CD) didukung oleh infrastruktur ini. Teknologi *Infrastructure as code* menjadi salah satu kunci dalam konsep teknologi DevOps (Null, 2015). Konsep ini merupakan perluasan dari *agile* dengan mencampurkan pola antara *development* dan *operations* untuk meningkatkan kolaborasi.

Dalam implementasi DevOps, banyak *tools* yang terlibat, antara lain *continuous integration server, configuration management tool, monitoring* dan *platform chat*. Setiap *tools*

memiliki peran masing-masing, namun untuk mengelolanya membutuhkan waktu yang lama (Atalay, 2017; Morris, 2016). Tim DevOps memberikan perintah atau tugas secara berulang-ulang menjadi salah satu penyebab kurang optimalnya penggunaan *tools* tersebut. Selain itu Slack sebagai media chat yang digunakan oleh tim, baru dimanfaatkan untuk diskusi percakapan sehari-hari, belum melibatkan tim untuk dapat mengakses *collaboration tools* melalui chat. ChatOps dibangun untuk otomasi perintah tersebut sehingga lebih efektif (Kyrpychenko, 2018). Konsep teknologi ChatOps adalah menempatkan bot untuk mengelola operasi teknis dan bisnis ke *chat* klien (Beer, 2016). Teknologi ini juga dapat melakukan pengecekan terhadap tugas atau perintah yang diberikan ke *tools* DevOps. Melakukan otomasi pada DevOps dapat mengefesienkan alur kerja (Beer, 2016), meningkatkan produktivitas (Gupta, 2018; Kyrpychenko, 2018) dan juga menghemat biaya (Gupta, 2018).

Dalam penelitian ini akan fokus membangun ChatOps untuk sistem integrasi di BSI. Melakukan penganalisisan kebutuhan, kemudian menerapkannya dalam teknologi ChatOps. Selain melakukan otomasi infrastruktur, ChatOps juga dapat melakukan pengecekan secara berkala terhadap collaboration sistem DevOps yang ada di BSI.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, dapat diperoleh rumusan masalah sebagai berikut: Bagaimana membangun ChatOps dan mengetahui dampak implementasi sistem otomasi ChatOps terhadap kinerja tim di Badan Sistem Informasi Universitas Islam Indonesia.

1.3 Batasan Masalah

Batasan dari penelitian yang dilakukan diantaranya sebagai berikut:

a. *Tools* kolaborasi yang digunakan untuk *continous integration* ChatOps adalah Gitlab.
b. Bot yang digunakan hanya satu jenis.
c. Skenario perancangan dan pengujian sistem disesuaikan dengan penggunaan sistem Gitlab yang sudah ada.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagi berikut: Dapat membangun ChatOps dan mengetahui dampak dari implementasi sistem ChatOps, di Badan Sistem Informasi Universitas Islam Indonesia.

1.5 Manfaat Penelitian

Manfaat dari penelitian yang dilakukan adalah sebagai berikut:

a. Manfaat Bagi Peneliti
    1. Menambah wawasan dan pengalaman peneliti tentang otomasi *infrastructure as a code* di DevOps dan menerapkannya langsung, beserta dengan pengembangannya.
    2. Sebagai salah satu syarat kelulusan strata satu (S1), Program Studi Teknik Informatika – Program Sarjana, Fakultas Teknologi Industri Universitas Islam Indonesia.
b. Manfaat Bagi Universitas
    1. Mengetahui kemampuan mahasiswa dalam mendalami materi teori yang telah diperoleh.
    2. Mengetahui kemampuan mahasiswa dalam menerapkan pengetahuannya dan sebagai bahan evaluasi
c. Manfaat Bagi Masyarakat

    Dapat menjadi sarana informasi dan referensi dalam pengembangan ilmu pengetahuan.

1.6 Metodologi Penelitian

1.6.1 Metodologi Pengumpulan Data

a. Observasi

Merupakan pengumpulan data dan informasi dengan cara meninjau dan mengamati secara langsung sistem integrasi yang digunakan untuk memperoleh data pengamatan yang lebih lengkap.

b. Diskusi

Merupakan pengumpulan data dan informasi dengan berkomunikasi secara kelompok untuk dapat memahami suatu topik yang sedang dibahas. Topik dalam sebuah diskusi dapat berkembang sehingga konsep perancangan lebih matang.

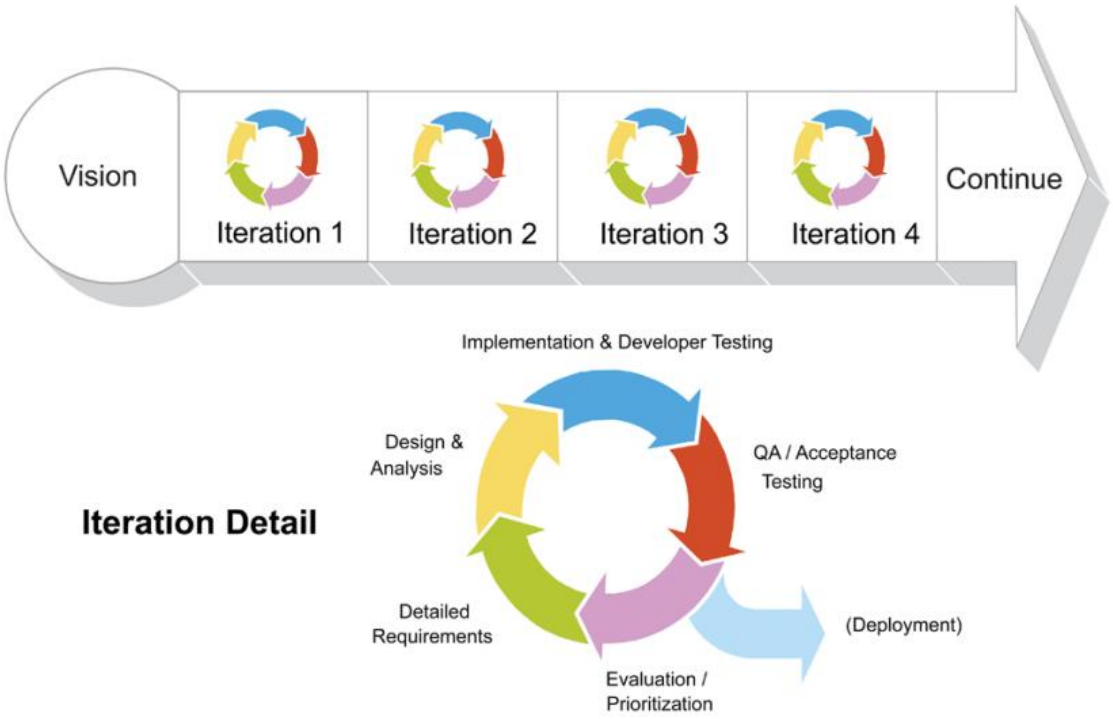
c. Studi Pustaka

Merupakan pengumpulan data dan informasi dengan mencari dan memperoleh data atau informasi yang diperoleh dari sumber tertulis, baik dari buku, jurnal, skripsi ataupun dari situs internet.

1.6.2 Metodologi Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah Scrum. menurut (James & Walter, 2010) Scrum adalah sebuah kerangka kerja untuk pengembangan

produk dimana proses dan teknik yang berbeda dapat diterapkan pada proyek-proyek yang kompleks. Scrum menggunakan iterasi *fixed-length*, yang disebut Sprint, dengan waktu pengerjaannya tidak lebih dari 30 hari. Tim Scrum mencoba untuk membangun produk yang berpotensi *releasable* setiap Sprint. Scrum memadukan semua kegiatan pembangunan dalam setiap iterasi, beradaptasi dengan ditemukan realitas pada interval yang tetap. Seperti yang ditunjukkan pada Gambar 1. 1.



Gambar 1. 1 *Workflow* Scrum

Dapat dilihat dari Gambar 1. 1 bahwa setiap proses dari perancangan, analisis, implementasi, pengujian, evaluasi, pengumpulan *requirement* dapat diulang sampai menjapai tujuan yang diinginkan.

1.7 Sistematika Penulisan

Laporan penelitian ini terbagi menjadi lima bab, yang dapat diuraikan secara singkat sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi uraian tentang Latar Belakang penelitian ini dibuat, Rumusan Masalah, Batasan Masalah, Tujuan, Manfaat, Metodologi Penelitian dan Sistematika Penulisan.

BAB II KAJIAN PUSTAKA

Bab ini berisi uraian tentang pembahasan terhadap kajian dari penelitian-penelitian sebelumnya yang berhubungan dengan penelitian. Penelitian terdahulu nantinya digunakan sebagai pertimbangan sistem yang akan digunakan.

BAB III METODOLOGI

Bab ini membahas bagaimana gambaran pekerjaan secara umum, diagram alur yang menjelaskan bagaimana sistem bekerja, rencana implementasi dan rencana pengujian sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas bagaimana sistem berjalan berdasarkan perancangan implementasi sistem pada kajian sebelumnya. Bab ini juga mengkaji pengujian performa sistem yang dijabarkan secara mendalam.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi urauian tentang kesimpulan-kesimpulan yang didapat serta mengemukakan saran yang penulis dapatkan selama pengerjaan Tugas Akhir.

DAFTAR PUSTAKA

Berisi sumber bacaan yang digunakan penulis yang menjadi acuan penulisan dan pendukung gagasan ide penulis.

LAMPIRAN

Berisi kode program pengembangan sistem dan hasil pengujian sistem.

BAB II
KAJIAN PUSTAKA

2.1 Pentingnya ICT bagi Perguruan Tinggi

Dunia digital dan informasi bergerak semakin cepat, peran ICT dalam pendidikan menjadi lebih penting dan semakin berkembang di abad ke-21 (Sarkar, 2012). Dalam penelitian Lench (2015) juga dijelaskan seberapa pentingnya perkembangan TIK di abad ke 21. ICT mengubah banyak aspek di kehidupan, begitu juga peran ICT dalam pendidikan semakin berkembang. Dalam berbagai penelitian dijelaskan peran perguruan tinggi dalam konteks ekonomi berbasis pengetahuan dan *globalisasi* adalah untuk memberikan individu kemampuan untuk mengubah informasi menjadi pengetahuan yang bermanfaat secara sosial, memodernisasi masyarakat dan meningkatkan taraf hidup (Kong & Li, 2009; Shaikh, Zaffar, 2009). Sehingga ICT dianggap menjadi garis kehidupan untuk pertumbuhan abad ke-21, seperti yang terjadi di India (Sarkar, 2012) dan Pakistan (Shaik & Khoja, 2011).

Meningkatkan kualitas pendidikan dan pengajaran merupakan masalah penting, terutama pada saat penyebaran pendidikan di negara berkembang (Sarkar, 2012). Dua puluh negara telah mengadopsi "*The constitution of the United Nations Educational* (UNESCO)" dalam penerapan ICT mereka. Secara garis besar prinsip-prinsip UNESCO pada ICT dalam pendidikan dapat diringkas sebagai berikut ini (Unesco, 2009):

a. Teknologi lama dan baru perlu digunakan secara seimbang.
b. Memenuhi tujuan pendidikan internasional pada tahun 2015, yang membutuhkan investasi besar di lembaga pelatihan guru.
c. Permintaan untuk akses pendidikan di negara maju dan berkembang tidak dapat dipenuhi, tanpa memangkas jarak dengan mode virtual pembelajaran (universitas virtual).
d. Tujuan pendidikan tidak dapat dipenuhi tanpa adanya isu gender. Jika memungkinkan, diusulkan indikator yang akan mengatasi kebutuhan untuk mengukur gap gender.

ICT sangat berpotensi untuk memperluas kesempatan pendidikan baik formal maupun non-formal, meningkatkan pembelajaran dan penelitian baik dari konstruktivis dan instruktivis teori-teori belajar. Secara umum manfaat ICT di dunia pendidikan sangat banyak, dalam penelitian (Sarkar, 2012) disebutkan bahwa mengadopsi teknologi dapat meningkatkan akses ke pendidikan. Pengaksesan tidak terbatas oleh ruang, waktu dan konten, hal ini berkaitan dengan teknologi virtual learning yang digunakan seperti *Hughes Net Global Educations*

Interaktif platfrorm yang berusaha mencirikan pendidikan di masa depan sebagai pendidikan *Real Time Interaktif* (Sarkar, 2012). Pemanfaatan teknologi juga berpengaruh dalam sistem pengajaran *offline*. Jumlah pertumbuhan masa pendidikan yang tinggi membuat kelas semakin besar dan sulit untuk siswa mendapatkan pengetahuan jika guru hanya audio-visual sebagai media pengantar pembelajaran. Tidak dipungkiri bahwa penggunaan teknologi dapat meningkatkan kognitif siswa dan prestasi, jika digunakan secara tepat dan sesuai (Sarkar, 2012).

Dalam studi (Richardson, 2008; Shaikh, Zaffar, 2009) menyatakan bahwa ICT tidak hanya membantu perguruan tinggi, namun juga mempersempit kesenjangan digital dan membantu meningkatkan kualitas pembelajaran dan hasil pendidikan. ICT mempengaruhi banyak aspek dalam kehidupan tidak hanya pendidikan, melainkan pembangunan, pekerjaan, pertumbuhan ekonomi, administrasi, pengurangan kemiskinan, penelitian dan *globalisasi* (Aypay, 2010; Shaikh, Zaffar, 2009).

Efektivitas, biaya, ekuitas dan keberlanjutan adalah empat isu yang harus ditangani ketika mempertimbangkan dampak keseluruhan dari penggunaan ICT dalam pendidikan. Dalam penelitian (Sarkar, 2012) disebutkan permasalahan dalam pengadopsian ICT dalam dunia pendidikan sangat beragam misalnya, membutuhkan biaya yang tinggi dalam pemasangan, pengoperasian dan pemeliharaan ICT. Banyak negara berkembang yang ketersediaan listrik dan jaringan telepon belum tersedia, sehingga penerapan ICT yang baik masih terganggu. Sarkar (2012) menyatakan bahwa kurangnya fasilitas ICT dan infrastruktur adalah hambatan yang signifikan untuk penggunaan ICT. Dapat disimpulkan bahwa infrastruktur ICT yang kuat di perguruan tinggi adalah hal kritis yang harus diperhatikan dan prasyarat bagi pembangunan berbasis pengetahuan.

2.2 Teknologi Zaman Modern

2.2.1 Tantangan Infrastruktur IT di Masa Kini

Pada tahun 2008 Daryl Plummer dan Homs Bittman menyatakan bahwa “tahun 2012, 80% dari 1000 perusahaan akan menggunakan beberapa layanan *cloud computing*, dan 30% dari mereka akan membayar untuk infrastruktur *cloud computing*” (Jenhani, 2011). Prediksi ahli dari IDC, Forrester, The Yankee Group dan RedMonk semua mendukung pernyataan tersebut (Jenhani, 2011). Perkembangan *cloud computing* ini, tidak terlepas dari perkembangan internet yang menjadi tonggak awal kemunculannya.

Banyaknya kalangan industri yang mengadopsi *cloud computing* untuk infrastruktur mereka, membuat banyak lembaga pendidikan memutuskan untuk pindah sebagian atau seluruh infrastruktur mereka ke *cloud* seperti yang dilakukan Amerika Serikat (Jenhani, 2011; Makoza, 2015). Hal ini kemudian diikuti oleh lembaga perguruan tinggi di Jerman pada awal tahun 2011 (Jenhani, 2011). Pengadopsian infrastruktur *cloud computing* ini juga dilakukan oleh perguruan tinggi di negara berkembang, seperti National University of Rwanda dan Kigali Institute of Education di Rwanda, Universitas Nairobi dan Kenya Methodist University di Kenya dan Universitas Mauritius (Kihara & Gichoya, 2014; Sultan, 2010).

Cloud computing adalah sebuah abstraksi yang didasarkan pada gagasan penyatuan sumber daya fisik dan menjadikan sumber daya virtual (Jenhani, 2011). Sedangkan menurut Ercan (2010) *cloud computing* adalah gaya komputasi dalam sekala besar yang mampu menyediakan layanan untuk struktur eksternal menggunakan teknologi internet. *Cloud computing* didasarkan pada konvergensi virtualisasi, komputasi *utilitas* dan layanan perangkat lunak yang dapat diakses melalui internet (Makoza, 2015). National Institute of Standards and Technology (NIST) membagi tiga model layanan pada teknologi cloud computing, yaitu: *Sofware as a Service* (SaaS), *Platform as a Service* (PaaS) dan *Infrastructure as a Service* (IaaS) (Mell & Grance, 2011).

Penerapan *cloud computing* tidak selamanya berjalan lancar, terdapat beberapa tantangan dan permasalahan yang harus diperhatikan. Tantangan *cloud computing* di perguruan tinggi, antara lain: sulitnya mengontrol layanan IT untuk organisasi, isu keamanan atas data dan aplikasi, masalah hukum mengenai yurisdiksi kontrak jika layanan berada di luar negeri, masalah perjanjian tingkat layanan, kecepatan dan infrastruktur internet juga dapat mempengaruhi layanan, dukungan organisasi dan beberapa aplikasi mungkin tidak berjalan pada infrastruktur *cloud* dan isu-isu hak kekayaan intelektual (Ercan, 2010; Low, Chen, & Wu, 2011; Okai, Uddin, Arshad, Alsaqour, & Shah, 2014).

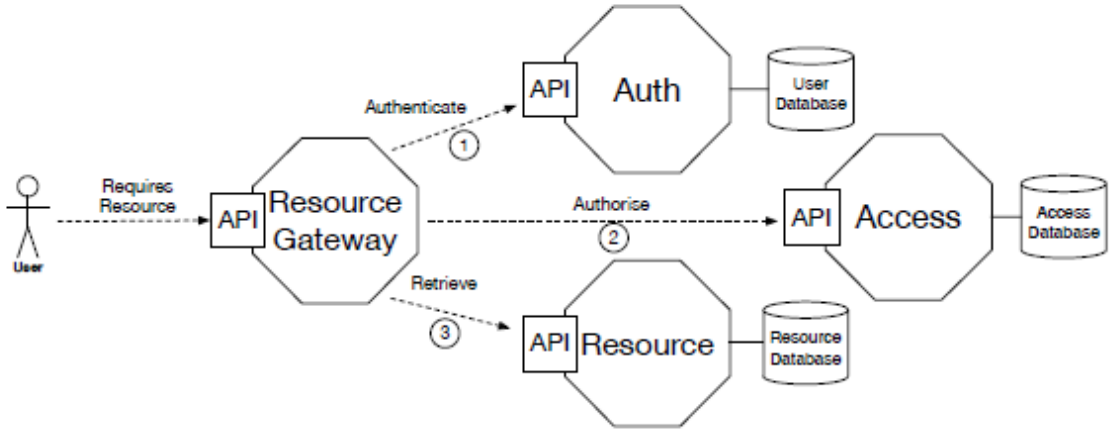
Permasalahan yang harus diperhatikan adalah bagaimana mengontrol infrastruktur *cloud computing*, tidak dapat dipungkiri perkembangan teknologi *cloud* dan virtualisasi semakin cepat dan pengelolaannya semakin sulit dikendalikan. Teknologi *Infrastructure as Code* kemudian muncul untuk mengatasi permasalahan pengelolaan infrastruktur *cloud* ini (Morris, 2016). Kemudian semakin berkembang dan muncul teknologi otomasi yang juga dapat diadopsi dengan menerapkan *Infrastructure as Code*.

2.2.2 Kepunahan Arsitektur Tradisional

Infrastruktur *cloud* semakin populer digunakan di industri maupun perguruan tinggi (Jenhani, 2011; Makoza, 2015). Sebagian besar perusahaan yang menggunakan sistem *cloud* harus mampu berinovasi dengan cepat terhadap produk aplikasi yang mereka kembangkan. Ini adalah alasan mengapa *continuous delivery* menjadi populer, terutama di startup, perusahaan internet besar dan penyedia layanan SaaS. Pengembangan produk aplikasi yang menggunakan *cloud* sangat terbatas dan sulit, jika masih menggunakan arsitektur tradisional atau *monolith* (Villamizar et al., 2015).

Monolith adalah aplikasi perangkat lunak yang terdiri dari modul yang tidak independen (Dragoni et al., 2016). Arsitektur ini sulit untuk mendistribusikan tanpa menggunakan kerangka tertentu atau ad hoc seperti, Network Objects, RMI atau CORBA (Dragoni et al., 2016). Selain itu masih banyak hambatan dan masalah dalam pengembangan arsitektur ini, yang di paparkan dalam beberapa paper, antara lain : Dragoni (2016) mengatakan *monolith* berukuran besar sehingga sulit untuk dikembangkan dan sangat kompleks serta membatasi skalabilitas. Pada paper (Kratzke, 2014) menambahkan *monolith* sering dikaitkan dengan "*dependency hell*", karena sistem aplikasi yang tidak konsisten dan terkompilasi secara penuh ketika ditambahkan atau diperbarui perpustakaannya. Dragoni (2016) juga menyatakan setiap perubahan satu modul dari arsitektur *monolith* membutuhkan *restart* seluruh aplikasi, dan *restart* ini biasanya memerlukan *downtime* yang lama.

Arsitektur *microservices* diusulkan untuk menggantikan arsitektur *monolith* dan mengatasi setiap masalahnya (Lewis & Fowler, 2014). Menurut Dragoni (2016) arsitektur *microservices* adalah aplikasi terdistribusi yang semua modul-modulnya independen minimal dan berinteraksi melalui pesan . Istilah "*microservices*" pertama kali diperkenalkan tahun 2011 pada lokakarya arsitektur, sebagai cara untuk menggambarkan ide-ide umum peserta dalam pola arsitektur perangkat lunak (Lewis & Fowler, 2014). Prinsip arsitektur *microservices* membantu manajer proyek dan pengembang untuk menyediakan pedoman atau desain dan implementasi aplikasi terdistribusi. Gambaran arsitektur *microservices* dapat dilihat di Gambar 2. 1.



Gambar 2. 1Arsitektur *microservices* (Dragoni et al., 2016)

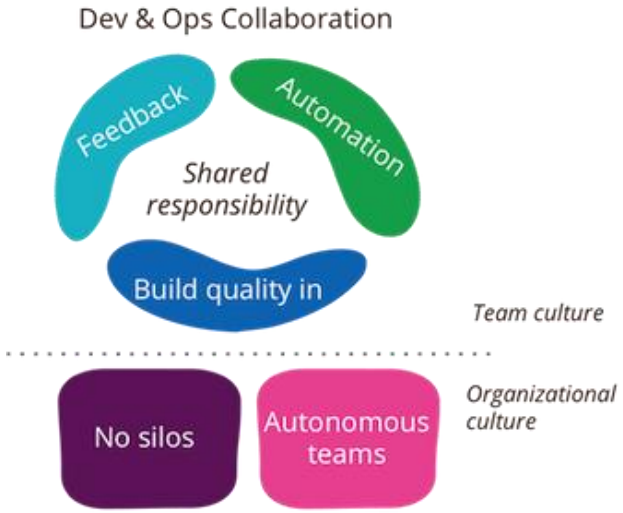
Banyak penelitian yang mendukung arsitektur *microservices* dapat mengatasi masalah pada arsitektur sebelumnya, antara lain: Fowler (2014) menyatakan *microservices* mendorong *continuous integration* dan sangat memudahkan perawatan perangkat lunak. Pernyataan itu didukung juga oleh pendapat Dragoni (2016) yang mengatakan *microservices* menerapkan jumlah fungsionalitas terbatas yang membuat kode dasar menjadi kecil dan inheren sehingga membatasi ruang lingkup *bug*. Dalam penelitian Merkel (2014) menyebutkan *microservices* dapat diterapkan untuk *containerization*. Dragoni (2016) juga menambahkan *microservices* tidak memerlukan *restart* lengkap dari seluruh sistem, tetapi hanya sebagian objek *microservices*, sehingga meminimalkan waktu *downtime*.

Microservices sekarang sudah menjadi tren baru (Dragoni et al., 2016), banyak perusahaan yang sudah mengadopsi arsitektur ini, seperti Amazon (Kramer, 2011), Netflix (Mauro, 2015), Gilt (Goldberg, 2014), Linkedln (Ihde & Parikh, 2015) dan SoundCloud (Calcado, 2014) untuk mendukung skalabilitas aplikasi dan produk mereka. Berdasarkan studi kasus yang dilakukan dalam penelitian Villamizar (2015) ini, menyebutkan *microservices* lebih cocok digunakan pada aplikasi berskala besar dengan pengguna antara ratusan ribu atau jutaan. Adopsi *microservices* harus dilaksanakan sebagai strategi bisnis jangka panjang dan tidak dipandang hanya sebuah proyek saja, karena adopsi mereka memerlukan upaya dan kemampuan yang harus dikembangkan secara bertahap. Penggunaan *microservices* dapat mengurangi biaya infrastruktur sebesar 70% atau lebih sehingga sangat menghemat anggaran (Villamizar et al., 2016).

2.3 Budaya Baru yang Membawa Kebaikan

Organisasi tradisional sering melakukan pengembangan aplikasi yang dilakukan oleh tim *development* dan tim *operations*, yang masing-masing memiliki tujuan tersendiri (de Feijter, 2017). Dalam situasi seperti itu, *development* bertugas menciptakan fungsi baru dan memperbaiki *bug*, sementara *operations* membuat infrastruktur yang handal yang memungkinkan perangkat lunak dapat berjalan dengan stabil. Dapat disimpulkan, peran *development* dan *operations* sangat bertentangan, *development* berusaha untuk perubahan sedangkan *operations* berusaha untuk kesetabilan. Hal ini menjadi masalah ketika merilis perangkat lunak untuk waktu yang cepat dan tepat, karena keduanya sangat bergantung satu sama lain. Oleh karena itu, budaya tradisional dinilai kurang efektif untuk pengembangan aplikasi di masa sekarang.

Berkembangnya pengadopsian arsitektur *microservices* di banyak perusahaan, juga mendorong perubahan budaya pengembangan perangkat lunak yang diterapkan. Arsitektur *monolith* membatasi frekuensi rilis (Balalaie, Heydarnoori, & Jamshidi, 2016) tetapi arsitektur *microservices* tidak membatasi dan dapat terus dikembangkan (Ellen, Riungu-kalliosaari, Mäkinen, & Lwakatare, 2016). Menurut penelitian Ca Technology (2013) yang "*Deliver Perangkat lunak Faster*", kerjasama antara tim-tim IT untuk pengembangan dan pemeliharaan perangkat lunak sangat dibutuhkan, contohnya dengan mengadopsi DevOps. Budaya DevOps mengaburkan batas antara peran *development* dan *operations* dan akhirnya dapat menghilangkan perbedaan (Wilsenach, 2015). Gambaran budaya kolaborasi DevOps dapat dilihat di Gambar 2. 2.



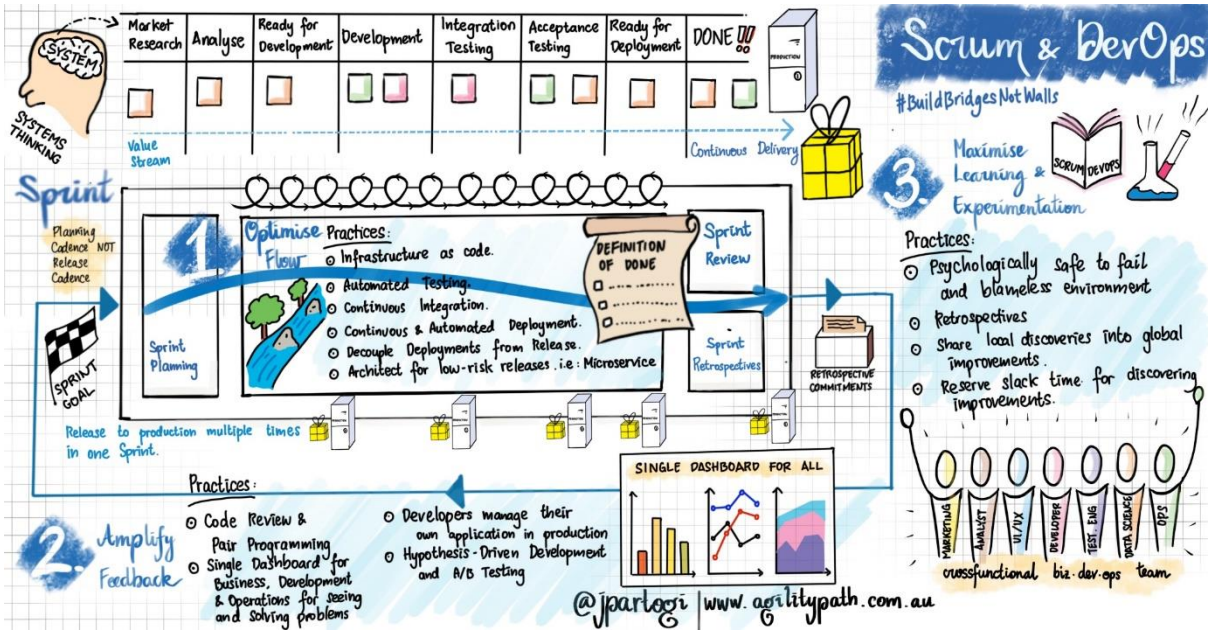
Gambar 2. 2 Kolaborasi DevOps (Wilsenach, 2015)

Sikap tanggung jawab bersama adalah aspek budaya DevOps yang mendorong kolaborasi yang lebih erat. Beberapa pergeseran organisasi diperlukan untuk mendukung budaya tanggung jawab bersama, Sehingga tidak ada silo atau batas antara *development* dan *operations*. Pergeseran organisasi dilakukan untuk mendukung tim otonomi, dimana dalam mengambil keputusan tidak berbelit-belit. Untuk memastikan perubahan dalam produksi tim perlu menilai *building quality* dalam proses pengembangan. Penting juga bagi tim untuk menilai umpan balik, landasan gerakan DevOps adalah otomasi.

Pengadopsian DevOps di beberapa perusahaan maupun organisasi menunjukkan manfaat bagi pengembangan produk aplikasi mereka, seperti yang ditunjukkan pada penelitian Ellen (2016). Devops mendukung kecepatan produksi perangkat lunak dan otomatisasi mengurangi usaha yang dibutuhkan ketika menyiapkan perilisan, sehingga memungkinkan organisasi untuk rilis lebih sering sesuai keperluan. Pada penelitian yang dilakukan Ellen (2016) penerapan DevOps dinilai menghemat penggunaan sumber daya untuk pengembangan dan pemeliharaan, karena adanya otomatisasi, salah satunya *continuous delivery*. DevOps juga berdampak pada karyawan, dimana pada penelitian Ellen (2016) mengungkapkan sering rilis membantu mengurangi stress karena kecemasan sehingga meningkatkan kesejahteraan. Hal itu membuktikan DevOps tidak hanya membawa manfaat untuk perusahaan, namun juga bagi pekerja.

Pengadopsian DevOps tidak mudah dan memerlukan perubahan yang kompleks sehingga harus memiliki strategi yang mendukung. Menurut Hamunen (2016) tantangan ketika mengadopsi DevOps secara garis besar dikelompokkan menjadi empat yaitu: kurangnya kesadaran, kurangnya dukungan, masalah dengan implementasi teknologi DevOps dan masalah dengan proses pengadaptasian DevOps untuk organisasi. Sedangkan, pada penelitian Ellen (2016) menambahkan tantangan pengadopsian DevOps adalah budaya organisasi yang tidak mudah dibentuk, kurangnya komunikasi dan rasa tanggung jawab. Sehingga sangat perlu diperhatikan tantangan-tantangan tersebut untuk penerapan DevOps yang baik dan tepat.

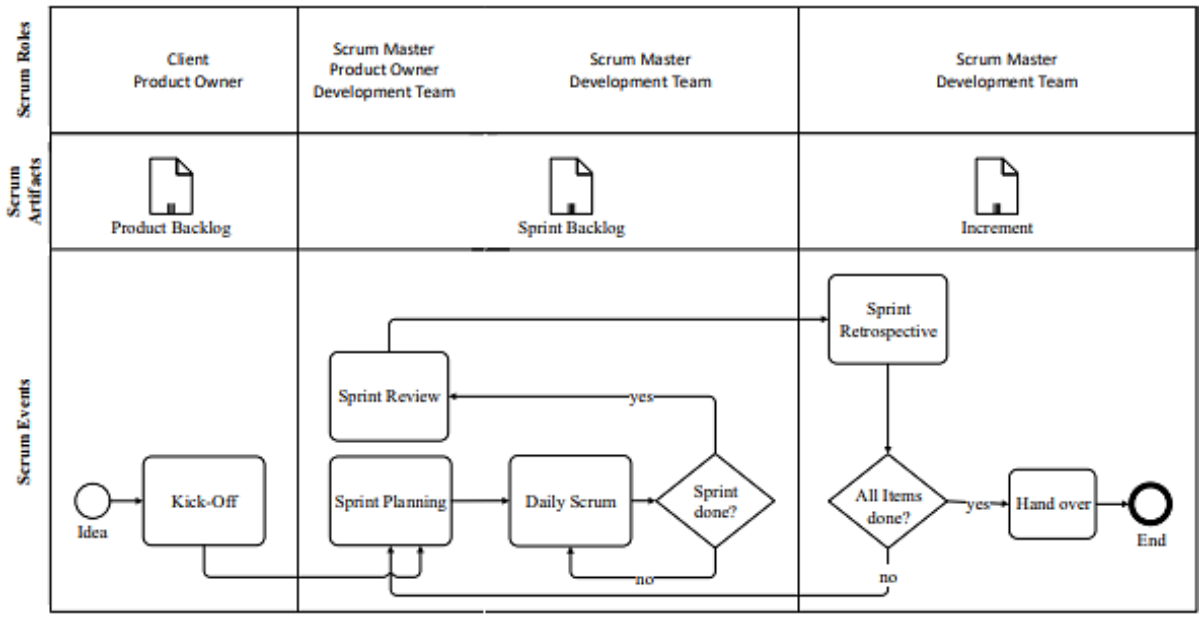
Dalam pengimplementasian DevOps, tidak memiliki kerangka kerja spesifik yang digunakan dalam penerapannya. Sehingga perlu ada jadwal yang mengatur otomasi rilis pada pengembangan perangkat lunak, kerangka kerja yang sesuai digunakan dalam penerapan budaya DevOps adalah Scrum. Gambar 2. 3 menjelaskan bagaimana irisan antara DevOps dan Scrum, sehingga keduanya menjadi sebuah kolaborasi yang tangguh. Aplikasi sistem akan dibangun menggunakan paradigma DevOps, dengan timing yang akan diatur pada Scrum.



Gambar 2. 3 Irisan Scrum & DevOps (Partogi, 2018)

2.4 Scrum

Scrum adalah salah satu dari banyak metode manajemen proyek *agile*. Menurut James dalam artikel (James, n.d.) *Agile* adalah tanggapan terhadap kegagalan paradigma manajemen proyek pengembangan perangkat lunak yang dominan (termasuk *waterfall*) dan mengadaptasi banyak prinsip dari *lean manufacturing*. Pada tahun 2001, 17 pionir metode serupa bertemu di Snowbird Ski Resort di Utah dan menulis *Agile Manifesto*, yang menempatkan penekanan baru pada komunikasi dan kolaborasi, perangkat lunak yang berfungsi, organisasi tim, dan fleksibilitas untuk beradaptasi dengan realitas bisnis (Sachdeva, 2016). Sebuah kerangka Scrum ditunjukkan pada Gambar 2. 4. Kerangka kerja Scrum terdiri dari *Scrum Roles*, *Scrum Artifacts* dan *Scrum Events* (Streule, Miserini, Bartlomé, Klippel, & De Soto, 2016), yang semuanya dijelaskan di bagiannya. Ekspresi Scrum diadaptasi dari Rugby, dimana posisi ditentukan secara rinci, dengan tujuan yang jelas untuk mencapai tujuan bersama (Streule et al., 2016). Selama penelitian ini iterasi scrum dilakukan sebanyak 10 kali, dengan jangka waktu dua minggu satu kali.



Gambar 2. 4 Kerangka Kerja Scrum (Streule et al., 2016)

a. Scrum Roles

Tim Scrum terdiri dari *Product Owner*, Tim Pengembangan dan *Scrum Master*, tim ini sendiri terorganisir dan lintas fungsional. *Product Owner* bertanggung jawab untuk memaksimalkan nilai proyek dan berhubungan langsung dengan klien. Individu yang melakukan pekerjaan yang sebenarnya adalah *development*. Para anggota tim ini semua sama (tidak ada manajer proyek) dan meskipun setiap orang memiliki bidang keahlian mereka, tim ini memiliki pertanggungjawaban secara keseluruhan. *Scrum master* memastikan bahwa setiap orang dalam tim, mengerti apa yang dimaksud dengan Scrum. Salah satu tugas utama dari *Scrum Master* adalah untuk menghilangkan hambatan yang dibawa ke Tim Scrum sehingga *development* dapat fokus pada pekerjaan mereka dan tidak diperlambat oleh hal-hal yang tidak penting. Sedangkan tujuan manajemen adalah untuk membantu dan mendukung Tim Scrum untuk mengeluarkan kemampuan terbaik mereka sehingga Tim Scrum mencapai tujuan mereka

b. Scrum Artifact

Scrum Artifacts dapat digambarkan sebagai elemen dengan definisi tertentu dalam kerangka Scrum. *Product Backlog* adalah daftar prioritas produk yang berbeda (misalnya membuat rencana lantai, mendefinisikan konsep proteksi kebakaran, merancang elemen beban). Setiap item dibagi menjadi tugas dan merupakan deskripsi sederhana rincian tentang apa yang perlu dilakukan oleh *development*. *Sprint Backlog* berisi sejumlah item yang dipilih oleh *Product Owner* dan Tim Pengembang dari *Product Backlog*. *Increment* adalah jumlah dari semua produk yang telah dilakukan.

c. Scrum Events

Bagian ini menjelaskan berbagai aktivitas, di mana Tim Scrum dapat menerapkan faktor-faktor kunci dari Scrum yaitu transparansi, inspeksi, dan adaptasi. Seperti di proyek lain pertemuan *Kick-Off* diselenggarakan - didasarkan pada tuntutan klien - dan *Product Owner* menciptakan *Product Backlog* untuk memenuhi permintaan ini. *Sprint Planning* kemudian dilakukan, *development* menentukan jumlah pekerjaan untuk suatu produk, yang paling penting dari *Product Backlog* dengan perencanaan poker. Setelah itu, *development* memilih Items yang digunakan untuk *Sprint*, dimulai dengan yang paling penting. Selama *Sprint*, *development* dan *Scrum Master* bertemu setiap hari untuk *Daily Scrum*, selama 15 menit yang dijadwalkan pada waktu yang sama dan lokasi yang sama setiap hari selama *Sprint*. Perkembangan dari setiap masalah yang dapat dilakukan akan dipresentasikan dalam *Sprint Review*. Setelah *Sprint Review*, pertemuan *Sprint Retrospective* biasanya diadakan. Tujuan dari pertemuan ini adalah untuk secara langsung mengevaluasi pihak yang terlibat, proses dan teknik yang digunakan, serta hubungan dan interaksi mereka.

2.5 Meningkatkan Performa Kinerja dengan ChatOps

Dalam prakteknya DevOps adalah tentang CAMS: "*a culture of automation, measurement and sharing"*, budaya otomasi, pengukuran dan berbagi. ChatOps dapat membawa konsep CAMS ini keseharian tim, dengan cara menempatkat alat langsung (bot) di tengah percakapan (Zyane, 2017). Pada penelitian Mak (2017) ChatOps dibangun untuk membuat prosedur penyebaran menjadi kurang rumit, mengurangi *black-boxed* dan kurang menakutkan. Mak menempatkan Chatbot untuk melakukan otomatisas penyebaran, grafik, pemantauan. ChatOps menyederhanakan penyebaran dengan otomatisasi, menghilangkan kesalahan koordinasi manual, memungkinkan semua orang untuk berkontribusi ke dalam proyek, mendorong komunikasi terbuka dan dapat mengakses informasi kapan saja ketika dibutuhkan (Mak, 2017). ChatOps juga diterapkan oleh HPE, mereka menempatkan Chatbot untuk mendeteksi masalah dalam lingkungan produksi mereka, yang kemudian membawa anggota tim untuk berkolaborasi tentang apa yang perlu dilakukan. Dalam skenario lain HPE memungkinkan untuk rollback, membangun baru atau melakukan penyebaran baru untuk memperbaiki masalah produksi (Fell, 2017).

Tiga komponen utama dalam ChatOps adalah *Collaboration tools*, Bot dan Integrasi sistem. Sistem ChatOps yang akan dibangun dalam penelitian ini menggunakan Slack sebagai

Collaboration tools, Hubot sebagai bot dan Gitlab sebagai integrasi sistem serta menggunakan Webhook sebagai *plug-in* notifikasi, berikut ini penjelasan lengkapnya:

Slack

Slack adalah program yang memungkinkan komunikasi instan antar rekan kerja. Pengguna dapat berbagi file dan mengirim pembaruan cepat yang diselenggarakan oleh grup atau proyek (Morgan, 2018). Slack memiliki channels yang membantu untk fokus dengan memisahkan pesan diskusi dan pemberitahuan berdasarkan tujuan, departemen atau topik. Slack juga menyediakan saluran pribadi untuk pelanggan yang membutuhkan komunikasi privasi. Slack dapat diakses seperti aplikasi lainnya dari iOS, Android dan Windows maupun linux baik dalam bentuk desktop maupun seluler. Integrasi di Slack juga memungkinkan pengguna untuk memusatkan semua pemberitahuan dari beberapa *tools* ke dalam Slack dan berkolaborasi secara instan.

Hubot

Hubot adalah robot milik infrastruktur obrolan, sebagai satu pengguna lagi yang dapat berkomunikasi dengan pengguna lain. Hubot dapat membantu mengotomasikan banyak tugas, seperti mengelola sistem pelacakan masalah. Hubot ditulis dalam bahasa coffescript di Node.js, sehingga dapat dikembangkan dengan mudah dengan menambahkan skrip sendiri (Soto, 2012). Hubot dikembangkan oleh Github oleh Ryan Tomay, sebagai cara lebih mudah bagi tim untuk mengelola dan memodifikasi perangkat keras dan perangkat lunak yang mendukung GitHub.com. Cukup dengan mengirim pesan ke Hubot, sama dengan mengirim pesan ke siapapun dari dalam klien obrolan. Setelah sumber kode Hubot dibuka secara umum oleh Github banyak yang memodifikasi dan menggunakan hubot tidak hanya untuk menangani berbagai tugas yang luas tetapi menyediakan konteks percakapan tugas (Metz, 2015).

Gitlab

Gitlab adalah Git berbasis web – manajer repository dengan wiki, pelacak masalah dan pipeline CI/CD. Gitlab menggunakan lisensi open source yang dikembangkan oleh Gitlab Inc. Pada rilis Gitlab 10.0, Gitlab mengambil langkah besar untuk tidak hanya sebagai manajemen kode, namun merambah ke *deployment* dan *monitoring*. Gitlab mengatur dan memodifikasi izin orang sesuai dengan perannya dan dapat memberikan akses pelacakan isu tanpa memberikan izin ke kode besar, hal ini bagus untuk tim dan perusahaan besar dengan kontribusi berbasis peran. Di Gitlab mendukung CI dengan gratis dan sangat bermanfaat untuk tim, selain itu Gitlab juga mendukung CI/CD secara otomatis tanpa campur tangan manusia (Peham, 2017).

Webhook

Webhooks adalah callback HTTP yang ditentukan oleh pengguna (cuplikan kode kecil yang ditautkan ke aplikasi web) yang dipicu oleh peristiwa tertentu. Kapan pun peristiwa pemicu terjadi di situs sumber, Webhook akan melihat, mengumpulkan data dan mengirimkannya ke URL yang ditentukan, dalam bentuk permintaan HTTP. Perbedaan mendasar antara API dan Webhook adalah jika API bekerja pada mekanisme keluaran berbasis permintaan, sedangkan Webhook bekerja pada mekanisme keluaran berbasis kejadian (Balaji, 2018). Secara singkat manfaatnya adalah memberikan notifikasi secara instan dan real-time.

BAB III
METODOLOGI

3.1 Analisis Kebutuhan Sistem

3.1.1 Metode Pengumpulan Data

Untuk memperoleh data yang dibutuhkan dalam penelitian ini, maka penulis menggunakan tiga tahapan antara lain observasi, diskusi dan studi pustaka.

Observasi

Observasi ini dilakukan di BSI Universitas Islam Indonesia, selama masa pengerjaan sistem. Kegiatan yang dilakukan adalah menganalisis sistem CI/CD yang sedang digunakan serta meneliti interaksi tim dalam menggunakan Gitlab. Hal itu dilakukan untuk mendapatkan data yang dibutuhkan selama proses perancangan sistem ChatOps. Observasi juga dilakukan saat melakukan pengujian, karena diperlukan pengamatan terhadap beberapa aspek yang dibutuhkan selama pengujian.

Diskusi

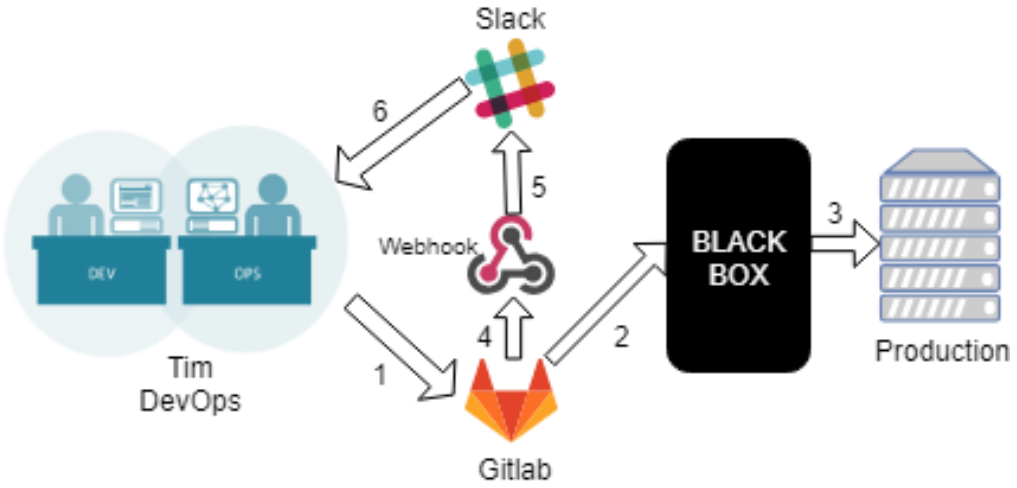
Diskusi dilakukan bersama dengan tim DevOps untuk menentukan bagaimana sistem ChatOps yang baik untuk diterapakan di BSI Universitas Islam Indonesia. Waktu pelaksanaan diskusi tidak terikat, selama mengerjakan sistem diskusi dilakukan secara bertahap. Mulai dari perancangan, pembangunan sistem ChatOps, mengidentifikasi rule otomasi untuk Gitlab yang diperlukan serta pengujian sistem.

Studi Pustaka

Pada tahap ini yang dilakukan adalah mempelajari dan meneliti berbagai sumber tentang pembangunan sistem ChatOps yang telah diterapkan oleh tim-tim DevOps lainnya. Sumber yang dipelajari mulai dari buku-buku, jurnal, skripsi dan referensi dari internet yang berhubungan dengan ChatOps. Studi lebih lanjut juga dilakukan dalam penentuan pengujian yang sesuai untuk penelitian ini. Daftar referensi yang digunakan dapat dilihat di daftar pustaka.

3.1.2 Identifikasi Sistem Sebelumnya

Sistem yang sedang digunakan sudah menerapkan konsep CI/CD dalam perilisan pembangunan *perangkat lunak* yang dilakukannya. Gambaran Arsitektur yang digunakan dapat dilihat pada Gambar 3. 1.

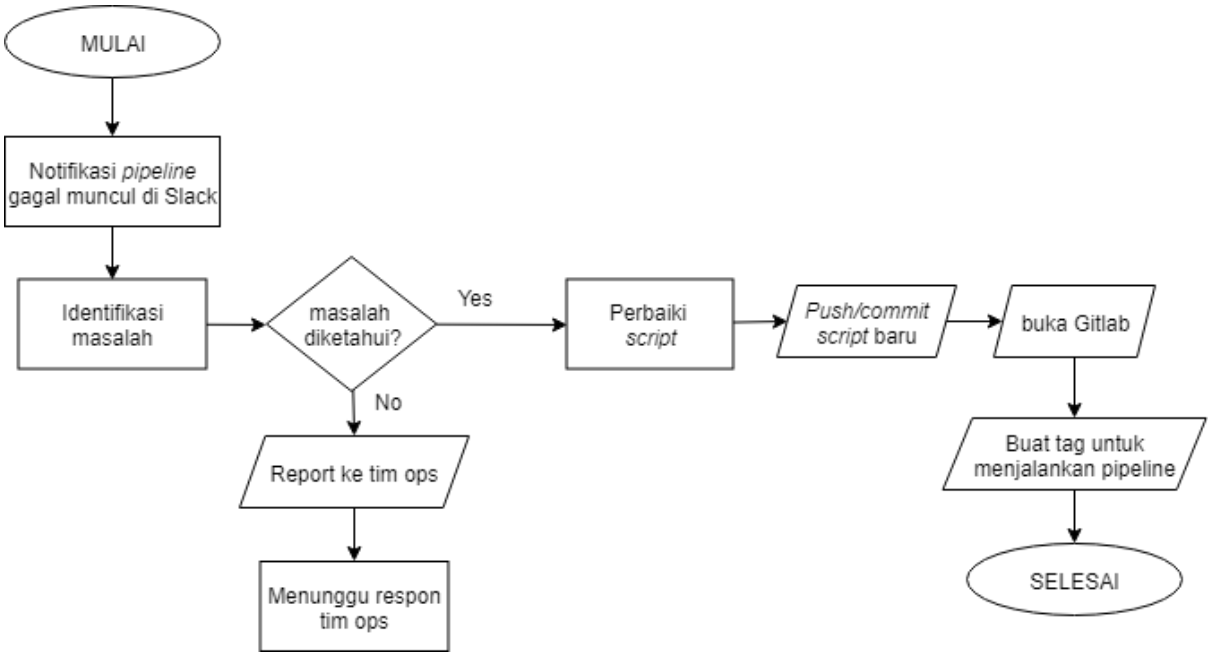


Gambar 3. 1 Gambaran Umum Sistem Sebelumnya

Setiap melakukan perilisan, *development* membuat proyek baru di Gitlab dan menambahkan beberapa berkas yang dibutuhkan untuk proses pembangunan (1). Jika persiapan sudah selesai, dilakukan *commit* dan *push* proyek (1). Gitlab Runners akan menjalankan *script* yang kita tentukan di berkas Gitlab-ci.yml, *pipeline* akan dijalankan sesuai dengan *script* secara pararel (3,4). Jika terjadi kegagalan proses, maka notifikasi akan tampil di *channel* Slack (5). Melalui *incoming* Webhook yang sudah ditambahkan di Slack dan diintegrasikan dengan Gitlab (4). Kemudian kesalahan akan diidentifikasi dan diperbaiki, setelah itu kode diperbarui dan di *push* ke Gitlab. Untuk dapat menjalankan proyek kembali, harus dibuat *tag* baru terlebih dahulu (1). Jika *tag* baru sudah dibuat di Gitlab, *pipeline* baru akan terbentuk dan *job* di proyek sedang dijalankan.

Namun pada alur masalah penanggapan pipeline ini, jika masalah tidak bisa ditemukan oleh seorang *development*, maka akan di *report* ke tim *operation* sehingga proses perbaikan tertunda karena harus menunggu *respon* dari tim *operations*. Pada tim *operations* sendiri sering terjadi penumpukan *request* pengecekan kegagalan *pipeline*, karena dalam suatu perilisan tidak hanya satu *development* yang meminta pengidentifikasian masalah, sedangkan *task* dalam *sprint* tim *operations* harus dikerjakan terlebih dahulu. Penumpukan *request* ini menghambat proses

perilisan perangkat lunak. Contoh permasalahan ini dapat dilihat pada alur yang terdapat pada Gambar 3. 2.



Gambar 3. 2 Alur Skenario Tanggapan Kegagalan *Pipeline* Sistem Lama

3.1.3 Analisis Kebutuhan Perangkat Lunak

Pada tahapan ini ditentukan perangkat lunak yang akan digunakan dalam pembangunan sistem otomasi ChatOps di BSI. Sebagai berikut:

a. Atom

Merupakan aplikasi yang digunakan untuk menuliskan kode program ChatOps dengan menggunakan bahasa *coffescript*, JSON dan Dockerfile.

b. Docker

Sebuah alat yang dirancang untuk mempermudah pembuatan, penerapan dan menjalankan suatu aplikasi atau sistem dengan menggunakan *container*. Docker lokal digunakan sebagai uji coba sebelum di-*build* ke server pusat BSI.

c. Redis

Salah satu database dari dunia NoQSL yang berbasis *key-value store*. Redis berjalan di *memory* sehingga lebih cepat dalam pengambilan data.

d. Hubot

Merupakan salah satu Bot yang dapat digunakan dalam pembangunan ChatOps. Pemasangan Hubot memerlukan node.js, npm serta Redis sebagai otaknya.

e. Slack

Private cloud messaging yang digunakan di BSI yang dapat diakses melalui situs maupun aplikasi di komputer ataupun smartphone.

f. OpenVPN

Perangkat lunak *open source* untuk *Virtual Private Networking* (VPN) yang *cross platform*, dimana aplikasi tersebut bekerja membuat koneksi *point-to-point tunnel* yang telah terenkripsi. Digunakan untuk mengakses *collaboration tools* DevOps BSI UII.

3.1.4 Analisis Kebutuhan Proses

Analisis kebutuhan proses yang diperoleh adalah berupa analisis kebutuhan masukan (*input*), kebutuhan proses, kebutuhan keluaran (*output*). Pengguna sistem otomasi ChatOps hanya satu jenis dan tidak terdapat pengguna lain seperti admin maupun yang lainnya. Pengguna merupakan *user* dari Slack yang telah terhubung di *channel*. Berikut ini daftar proses yang terdapat pada sistem beserta kebutuhan masukan dan keluaran yang ditujukkan pada Tabel 3. 1.

Tabel 3. 1 Analisis Kebutuhan Proses

No	Nama Proses	Deskripsi	Input	Output
1	Membuat proyek baru di Gitlab	Proses dalam membuat proyek baru di Gitlab sesuai dengan perintah masukan yang diberikan melalui *chat.*	Nama proyek dan alamat email atau username akun yang dituju.	Informasi hasil pembuatan proyek baru.
2	*Build* proyek di Gitlab	Proses dalam menjalankan maupun *rebuild* suatu proyek sesuai dengan perintah masukan yang diberikan melalui *chat.*	Alamat proyek dan nama *branch* yang digunakan proyek.	Informasi hasil *build* proyek.
3	Membagi proyek Gitlab ke grup tertentu	Proses dalam membagi proyek Gitlab ke grup tertentu, sesuai dengan perintah masukan yang diberikan melalui *chat*	Alamat proyek dan alamat grup Gitlab yang dituju.	Informasi hasil pembagian proyek.
4	Bergabung ke grup tertentu di Gitlab.	Proses untuk bergabungnya pengguna ke grup tertentu sesuai dengan perintah masukan yang diberikan melalui *chat*	*Username* akun dan alamat grup Gitlab.	Informasi hasil bergabung ke grup.
5	Bergabung ke proyek tertentu di Gitlab	Proses untuk bergabungnya pengguna ke proyek tertentu sesuai dengan perintah masukan yang diberikan melalui *chat*	*Username* akun dan alamat proyek Gitlab.	Informasi hasil bergabung ke suatu proyek.
6	Melihat daftar pengguna di proyek Gitlab	Proses dalam melihat daftar pengguna di suatu proyek di Gitlab sesuai dengan perintah	Alamat proyek di Gitlab yang dimaksud.	Menampilkan daftar informasi pengguna di

		masukan yang diberikan melalui *chat*		proyek yang dimaksud
7	Melihat daftar anggota di grup Gitlab	Proses dalam melihat daftar anggota, suatu grup di Gitlab sesuai dengan perintah masukan yang diberikan melalui *chat*	Alamat grup di Gitlab yang dimaksud.	Menampilkan daftar informasi anggota di grup yang dimaksud
8	Melihat daftar proyek di grup Gitlab	Proses dalam melihat daftar proyek, suatu grup di Gitlab sesuai dengan perintah masukan yang diberikan melalui *chat*	Alamat grup di Gitlab yang dimaksud.	Menampilkan daftar informasi proyek di grup yang dimaksud.
9	Melakukan pencarian di Gitlab	Proses melakukan pencarian proyek, pengguna, grup dan isu di Gitlab sesuai dengan kata kunci yang diberikan melalui chat.	Variabel yang akan dicari (proyek, pengguna, grup atau isu) serta kata kunci yang ingin dicari.	Menampilkan hasil dari pencarian yang dibutuhkan.
10	Membuat penjadwalan pipeline di proyek Gitlab	Proses membuat penjadwalan *pipeline* untuk suatu proyek tertentu, untuk menjalankan *job* pada waktu yang ditentukan.	Nama untuk penjadwalan dan alamat proyek	Menampilkan informasi hasil pembuatan penjadwalan *pipeline*
11	Membuat isu di Gitlab	Proses membuat isu untuk suatu proyek di Gitlab sesuai dengan alamat proyek dan deskripsi dari isu yang diberikan.	Nama isu, alamat proyek dan deskripsi isu	Menampilkan informasi hasil pembuatan isu pada proyek Gitlab melalui notifikasi
12	Mengomentari isu di Gitlab	Proses memberi komentar terhadap isu suatu proyek yang ada, sebagai bentuk tanggapan yang diberi sesuai dengan deskripsi yang diberi.	Id isu, alamat proyek dan deskripsi untuk mengomentari isu	Menampilkan informasi hasil mengomentari isu Gitlab melalui notifikasi
13	Membuat tag baru di proyek Gitlab	Proses membuat tag untuk suatu proyek digunakan sebagai parameter untuk *build* proyek di Gitlab	Nama untuk tag, alamat proyek dan pesan	Menampilkan informasi hasil pembuatan tag baru
14	Menutup isu di Gitlab	Proses menutup isu untuk suatu proyek di Gitlab sesuai dengan alamat proyek yang diberikan.	Id isu dan alamat proyek	Menampilkan informasi hasil penutupan isu pada proyek Gitlab
15	Mengirim tugas isu ke pengguna di Gitlab	Proses mengirim tugas isu tertentu ke suatu pengguna di Gitlab	Id isu, alamat proyek dan username pengguna	Menampilkan informasi hasil pengiriman tugas isu Gitlab.

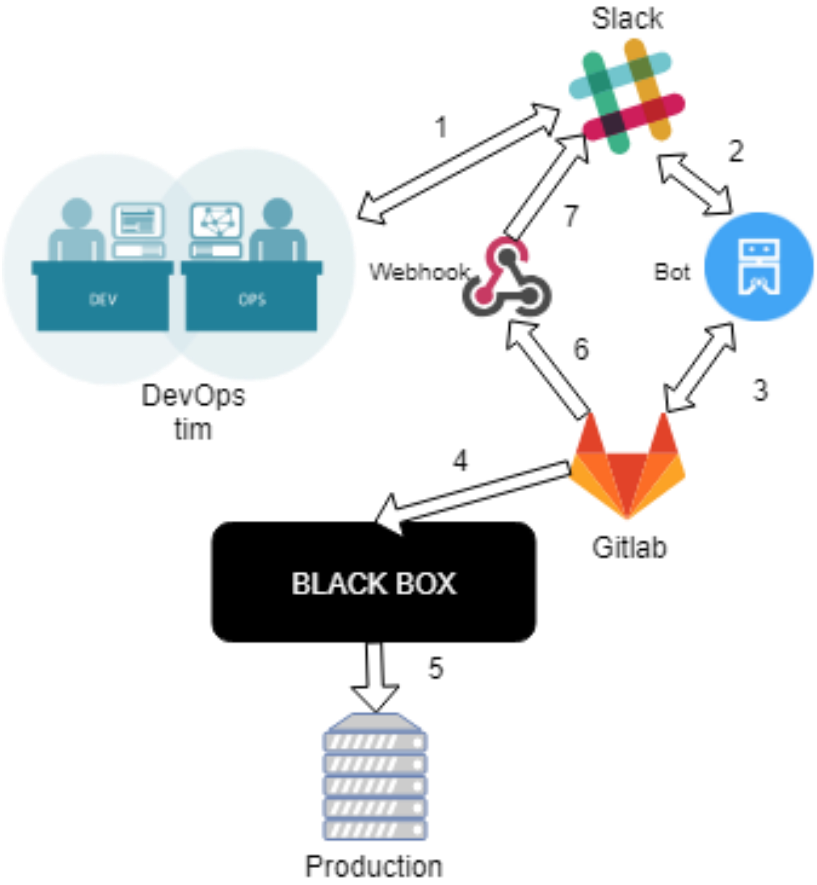
3.2 Perancangan

Perancangan dilakukan ketika data observasi yang dibutuhkan sudah terpenuhi. Merancang sistem ChatOps untuk diterapkan di tim, memerlukan identifikasi dan perancangan yang tepat. Diskusi juga dilakukan untuk mendapatkan perancangan sistem yang sesuai. Sebelum

merancang sistem yang baru, penulis perlu mengidentifikasi sistem yang sebelumnya untuk dapat dibandingkan nantinya dalam pengujian dan mendapatkan hasil yang diinginkan.

3.2.1 Gambaran Sistem Baru

Berdasarkan pengamatan dan diskusi yang dilakukan, diketahui bahwa sistem otomasi yang digunakan di BSI Universitas Islam Indonesia masih belum optimal. Maka diperlukan suatu sistem otomasi yang dapat membantu *workflow* tim DevOps menjadi lebih efisien. Dimulai dari pemanfaatan media Chatbot untuk melakukan otomasi perintah yang dilakukan sehari-hari. Berdasarkan penjelasan yang telah dijabarkan, gambaran sistem yang akan diimplementasikan dapat dilihat di Gambar 3. 3, sebagai berikut:



Gambar 3. 3 Gambaran Perancangan Sistem ChatOps

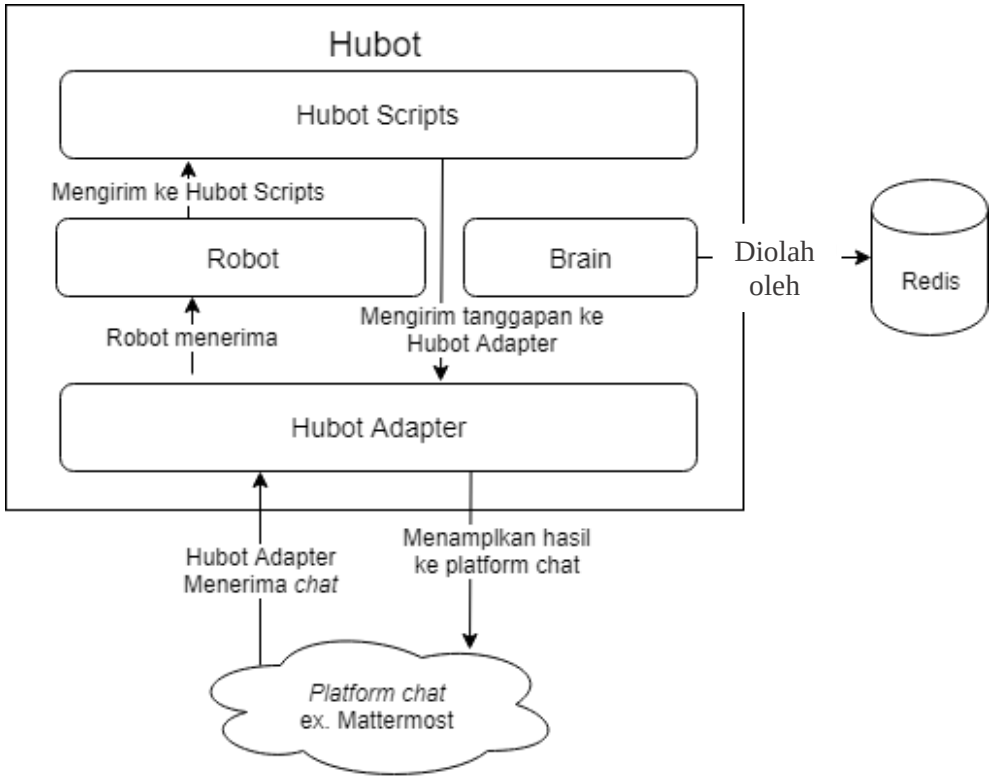
Gambar 3. 3 menjelaskan bagaimana alur kerja sistem otomasi dengan menggunakan Chatbot, yang akan diterapkan di BSI. Secara garis besar sistem yang digunakan akan sama dengan sebelumnya (Gambar 3. 1) namun bot ditambahkan untuk dapat meminimalisir penggunaan Gitlab secara langsung. Bot akan mengeksekusi setiap perintah yang diberikan

melalui Slack oleh tim DevOps (2). Gitlab akan merespon permintaan dari bot sesuai dengan *rule* yang sudah ditetapkan dalam bot (3). Chatbot yang melibatkan *collaboration tools* DevOps disebut ChatOps. Terdapat empat komponen utama yang akan saling terhubung dalam kerja sistem otomasi ini. Pertama, tim DevOps itu sendiri, yang memberikan perintah untuk dieksekusi melalui sebuah *chat*. Kedua, *Platform chat* sebagai antarmuka yang lebih baik bagi tim dibandingkan command line dan efisien karna semua terekam ke sebuah percakapan dalam melakukan tugas sehari-hari. Ketiga, bot yang disisipkan dengan berbagai *rule* yang akan dibuat untuk memproses tugas. Keempat, *collaboration tools* DevOps yang akan dihubungkan dan dikendalikan oleh bot dengan berbagai *rule*.

Platform chat yang digunakan oleh tim DevOps BSI adalah Slack. Penggunaan *Platform chat* ini dinilai masih kurang, karena jarang digunakan oleh tim dalam berkomunikasi sehari-hari. Pembangunan Chatbot ini diharapkan dapat meningkatkan aktifitas penggunaan Slack, sehingga lebih terpusat. *Collaboration tools* DevOps yang digunakan dalam pembuatan sistem ini selain Slack adalah Gitlab. Gitlab merupakan manajer repositori dalam pengembangan *perangkat lunak,* selain itu juga berfungsi untuk CI/CD.

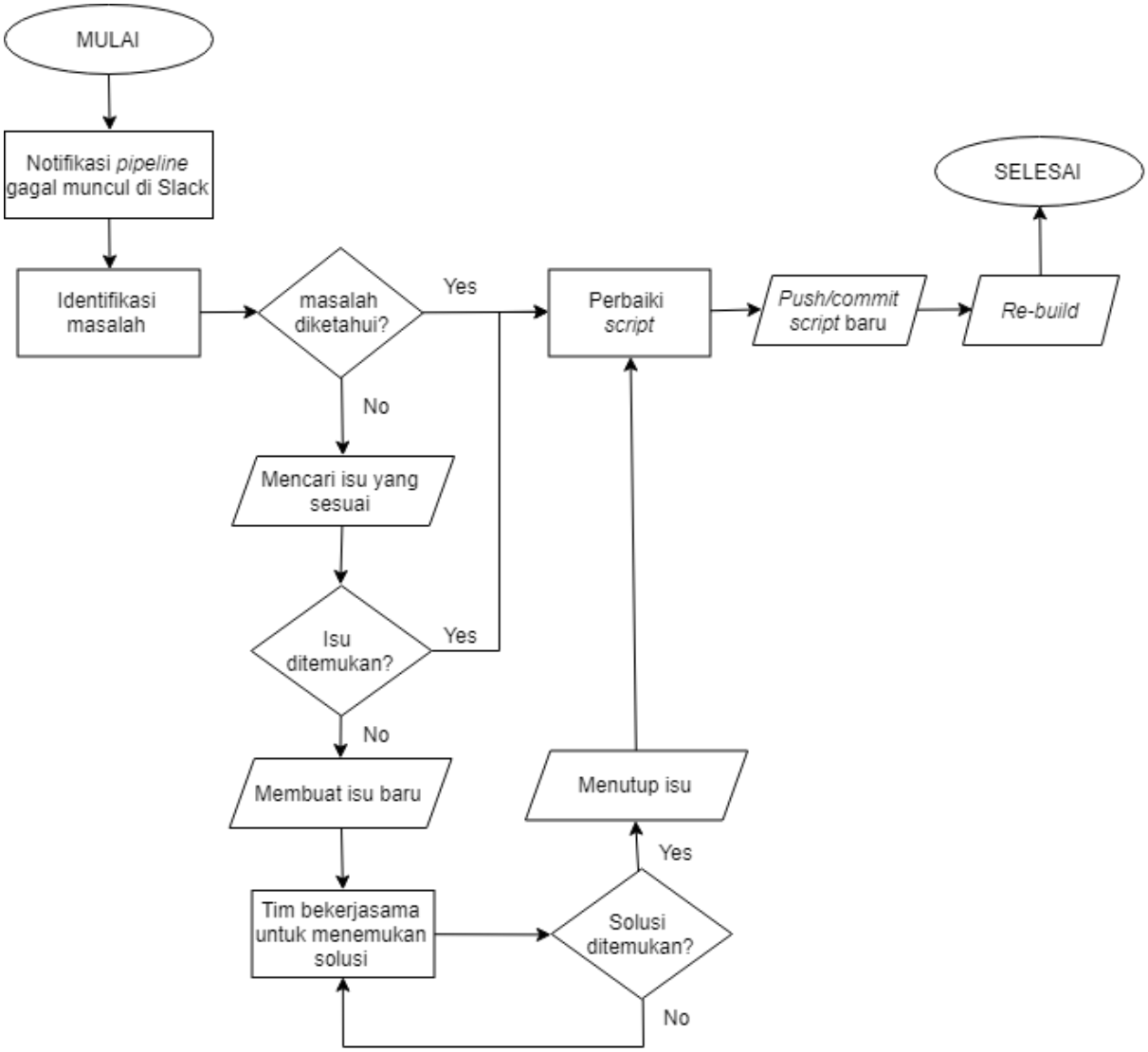
Hubot akan digunakan dalam implemenasi sistem ini, bot ini dikembangkan oleh Github dan bersifat *open source*. Hubot membutuhkan *tools* lain untuk dapat digunakan, yaitu Redis sebagai otaknya. Secara garis besar sistem ini akan meneruskan perintah tim DevOps melalui *chat* sesuai dengan dialog *rule* yang sudah diatur dalam Hubot ke Gitlab dengan berbagai aksi melalui *http request*. Dengan demikian sistem ini menerapkan *Continous Delivery* (CD) dan *Continous Integration* (CI).

Dalam Gambar 3. 4 dijelaskan secara detail bagaimana komponen Hubot, yaitu Hubot *script*s yang menyimpan kode *rule* untuk otomasi. Robot adalah bagian inti dari Hubot itu sendiri, Redis digunakan sebagai penyimpanan otak dari Hubot. Hubot adapter yang menjadi penghubung dengan *Platform chat*. Gambar 3. 4 juga menjelaskan bagaimana alur kerja Hubot, pertama *Platform chat* akan memberikan perintah ke *bot*, dan terhubung melalui Hubot *adapter*, disini *adapter* Slack yang akan digunakan untuk menghubungkan Slack dengan Hubot. Perintah akan diterima *bot* kemudian akan diteruskan ke Hubot *script*s. Bagian ini merupakan kumpulan *rule* dalam bentuk kode yang ditulis dalam bahasa *coffescript*. Perintah yang diberikan ke Hubot akan diidentifikasi oleh Hubot *script*s kemudian tanggapan akan dikirim ke Hubot Adapter. Program akan diproses menggunakan otak Hubot dalam Redis, sebelum dikirim ke *Platform chat*.



Gambar 3. 4 Komponen dan Alur Kerja Hubot (Atalay, 2017)

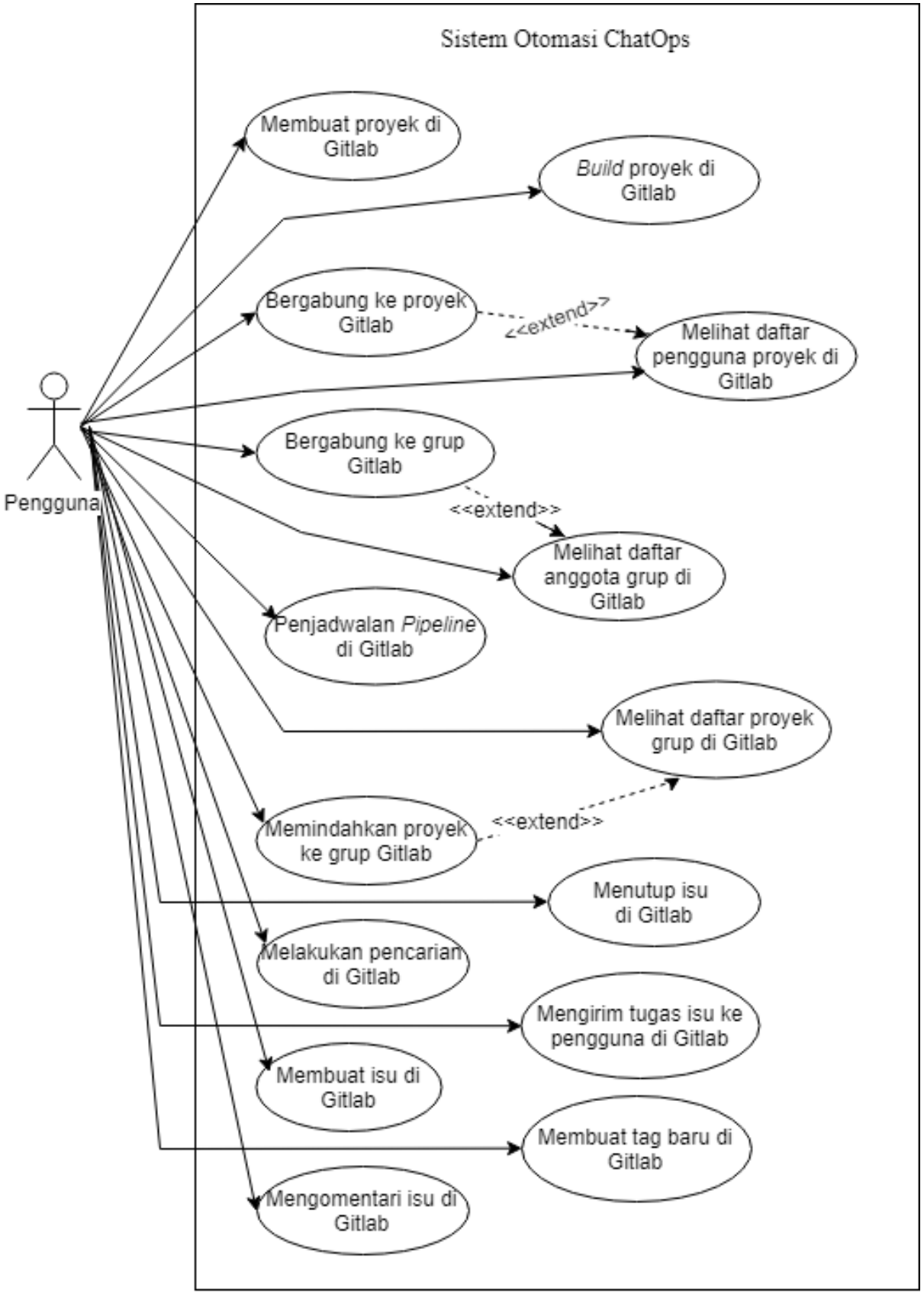
Pada sistem lama, jika *pipeline* yang gagal terdeteksi maka tim *development* akan mengidentifikasi masalah secara individu. Pada sistem ChatOps sebagian proses dialihkan ke Slack dengan memanfaatkan bot untuk mengirim berbagai perintah yang sesuai. Pada skenario sistem baru tim *development* dikenalkan dengan sistem kolaborasi baru untuk saling membantu dalam penyelesaian masalah kegagalan *pipeline*. Pemanfaatan manajemen pengetahuan melalui pengelolaan isu juga digunakan, selain untuk dokumentasi juga digunakan untuk rujukan jika terjadi masalah yang sama. Pada sistem ChatOps juga terdapat kebijakan untuk memberikan isu ke pengguna yang bertanggung jawab karena kesalahan *pipeline*. Pada skenario pengujian ini alur yang digunakan berfokus pada masalah kegagalan *pipeline* pada tim *development*. Alur dari penanganan kegagalan pipeline yang menggunakan sistem ChatOps dapat dilihat pada Gambar 3. 5.



Gambar 3. 5 Alur Skenario Tanggapan Kegagalan *Pipeline* pada Sistem ChatOps

3.2.2 Perancangan *Use Case* Diagram

Use case diagram berfungsi untuk memberikan gambaran secara ringkas bagaimana sistem dapat digunakan. Melalui *use case* dapat diketahui fungsi apa saja yang dapat dilakukan di dalam sistem. Pada Gambar 3. 6 menunjukkan bagaimana gambaran cara kerja Hubot dalam melakukan otomasi ke berbagai pekerjaan yang menyangkut Gitlab. Hubot akan mengidentifikasi setiap perintah yang dimasukan dan mengeksekusi sesuai dengan *rule* yang dibuat. *Use case* mempresentasikan sebuah interaksi antara aktor dengan sistem. Terdapat satu aktor yaitu pengguna, yang dapat melakukan beberapa aktivitas yang terdapat pada sistem.



Gambar 3. 6 Use case diagram sistem otomasi ChatOps

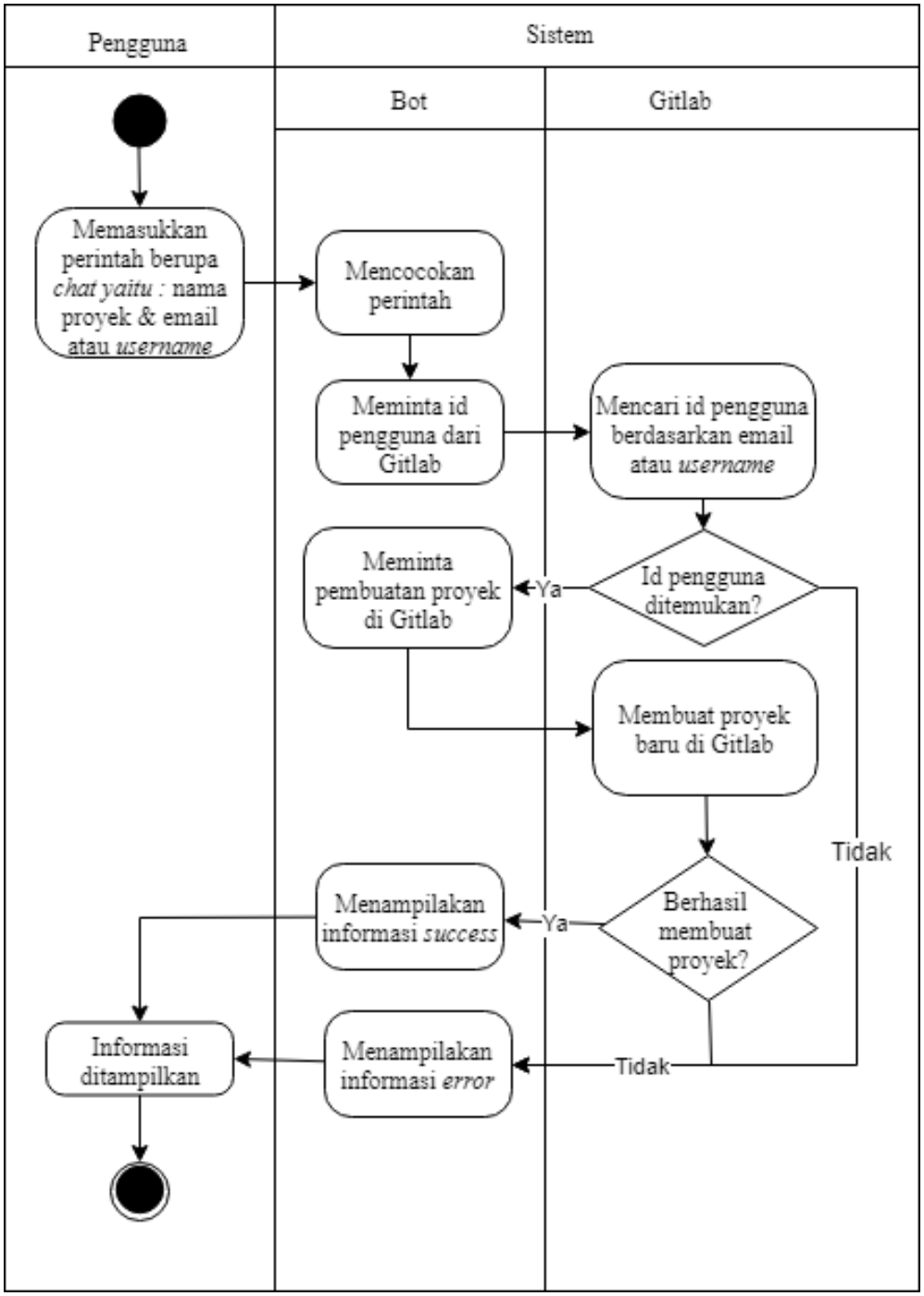
Aktivitas sistem dibagi menjadi empat bagian berdasarkan pada studi kasus yang disampaikan oleh (Jones, 2017) dalam "*Improving your IT service delivery and operations with ChatOps*". Manajemen kejadian dan notifikasi, meliputi *build* proyek di Gitlab dan membuat tag baru untuk proyek. Manajemen masalah, diantaranya membuat isu, memberi komentar terhadap isu, menutup isu serta memberikan tugas isu ke pengguna yang bersangkutan. Mengubah dan manajemen perilisan dengan *Continuous Delivery* membuat proyek baru di Gitlab, berbagi proyek ke grup Gitlab, bergabung ke grup atau proyek tertentu di Gitlab, memindahkan proyek di grup, dan membuat penjadwalan *pipeline.* Manajemen pengetahuan, antara lain melihat daftar pengguna proyek, melihat anggota grup, melihat daftar proyek di grup Gitlab dan melakukan pencarian di Gitlab melalui perintah di-*chat*.

3.2.3 Diagram Aktivitas

Diagram aktivitas memodelkan alur kerja sebuah urutan aktivitas yang ada pada sistem. Diagram ini dibuat untuk menggambarkan aktivitas aktor. Berikut diagram aktivitas dari sistem otomasi ChatOps di BSI. Pengguna merupakan tim DevOps yang memiliki akun di Slack. Sistem memiliki dua bagian tersendiri, yaitu bot yang merupakan robot yang memiliki aturan untuk mengakses *tools*. Gitlab adalah *tools* yang digunakan untuk membantu proses perilisan *software.* Pengguna, bot dan Gitlab berperan aktif dalam sistem otomasi ChatOps di BSI untuk mendukung CI/CD.

a. Diagram aktivitas membuat proyek baru di Gitlab

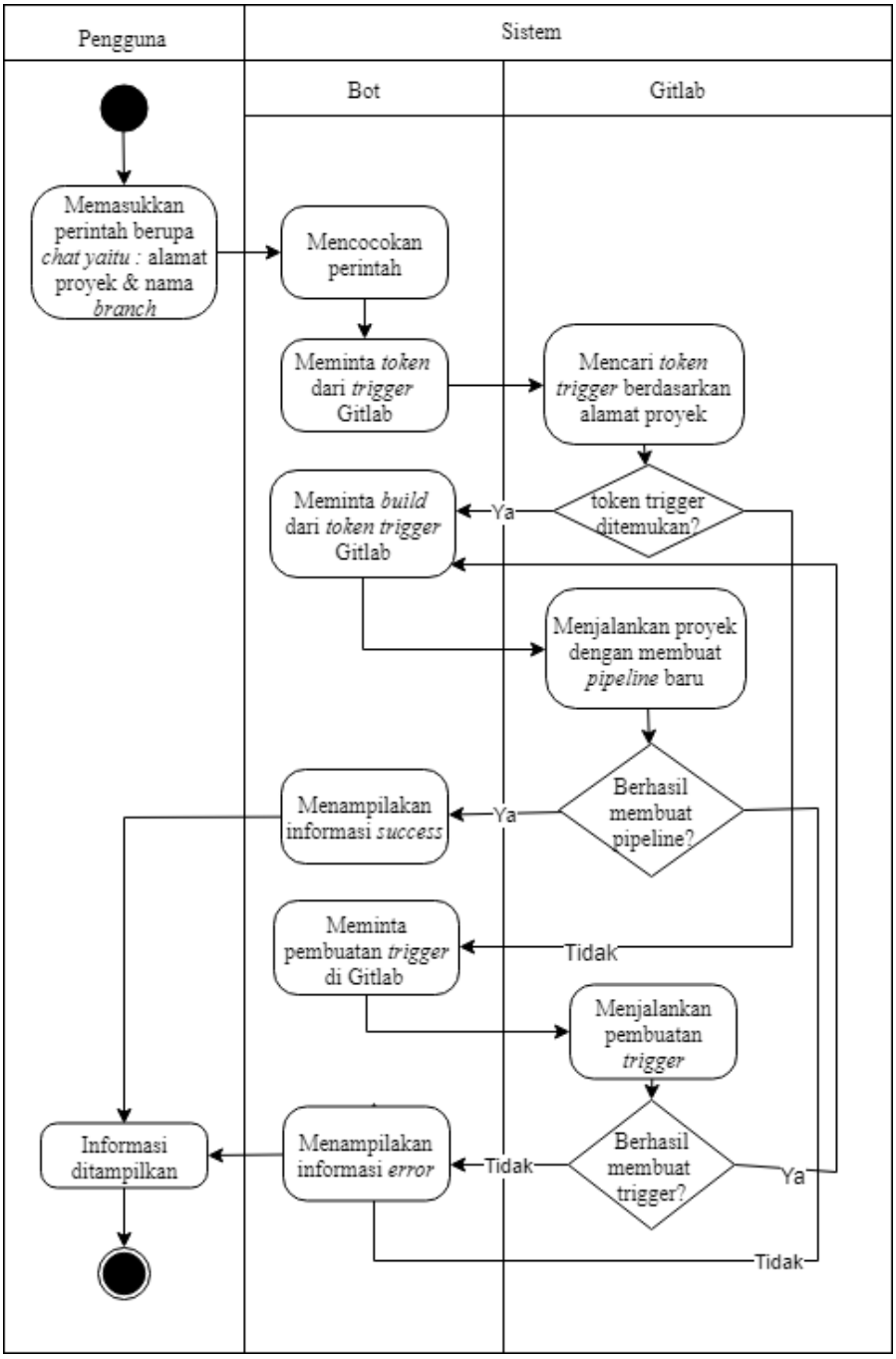
Gambar 3. 7 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat membuat proyek baru di Gitlab melalui perintah di *chat*. Pengguna dapat memasukan perintah, berupa nama proyek yang akan dibuat dan email atau *username* akun Gitlab. Kemudian sistem akan mencocokkan perintah dan mengirim permintaan ke Gitlab untuk mendapatkan id pengguna berdasarkan email atau *username* yang dimasukan. Gitlab akan memproses permintaan dan mengirim data yang dibutuhkan untuk membuat proyek di Gitlab. Setelah itu Sistem akan mengirim permintaan untuk membuat proyek, jika data yang diperlukan terpenuhi.Gitlab akan memproses permintaan pembuatan proyek baru berdasarkan nama masukan sebelumnya. Data hasil proses pembuatan proyek baru akan dikirim ke sistem dan kemudian akan ditampilkan ke Slack oleh sistem.



Gambar 3. 7 Diagram aktivitas membuat proyek baru

b. Diagram aktivitas *build* proyek di Gitlab

Pada Gambar 3. 8 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat membuat proyek baru di Gitlab melalui perintah di *chat*.



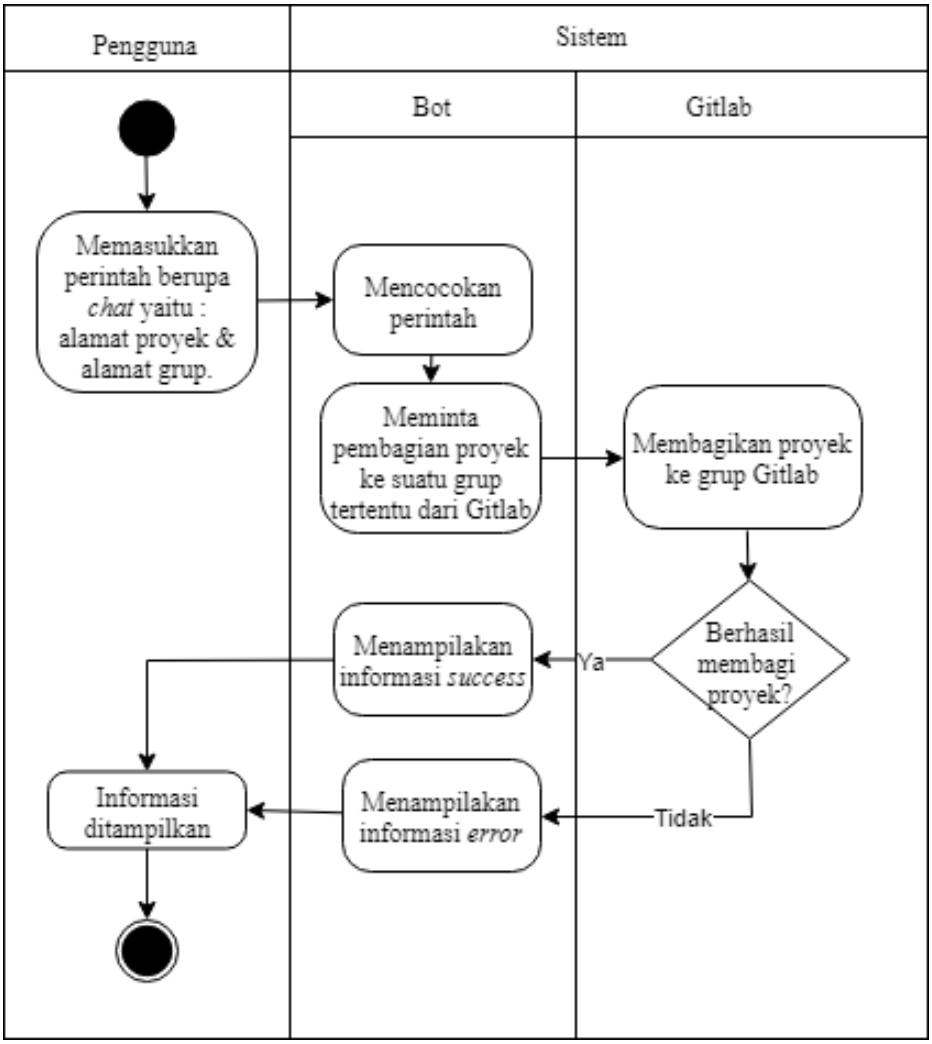
Gambar 3. 8 Diagram aktivitas *build* proyek di Gitlab

Pengguna dapat memasukan perintah, berupa alamat proyek yang akan dijalankan dan nama *branch*. Kemudian sistem akan mencocokkan perintah dan mengirim permintaan ke Gitlab untuk meminta token trigger dari Gitlab. *Token trigger* akan dicari berdasarkan alamat proyek yang dimasukan sebelumnya. Kemudian sistem akan mengirim permintaan *build* proyek ke Gitlab, jika data *token trigger* didapatkan di proses sebelumnya. Gitlab akan menjalankan proyek berdasarkan nama branch yang dimasukan, sistem akan menampilkan informasi hasil

dari data proses *build* proyek. Jika *token trigger* sebelumnya tidak diperoleh maka sistem akan mengirim permintaan pembuatan *trigger* baru untuk proyek tersebut. Setelah *trigger* dibuat oleh Gitlab, maka proses permintaan *build* proyek diulang kembali. Data hasil proses *build* proyek akan dikirim ke sistem dan kemudian akan ditampilkan ke Slack oleh sistem.

c. Diagram aktivitas membagi proyek ke grup Gitlab.

Gambar 3. 9 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat membagi proyek ke grup Gitlab melalui perintah di *chat*.



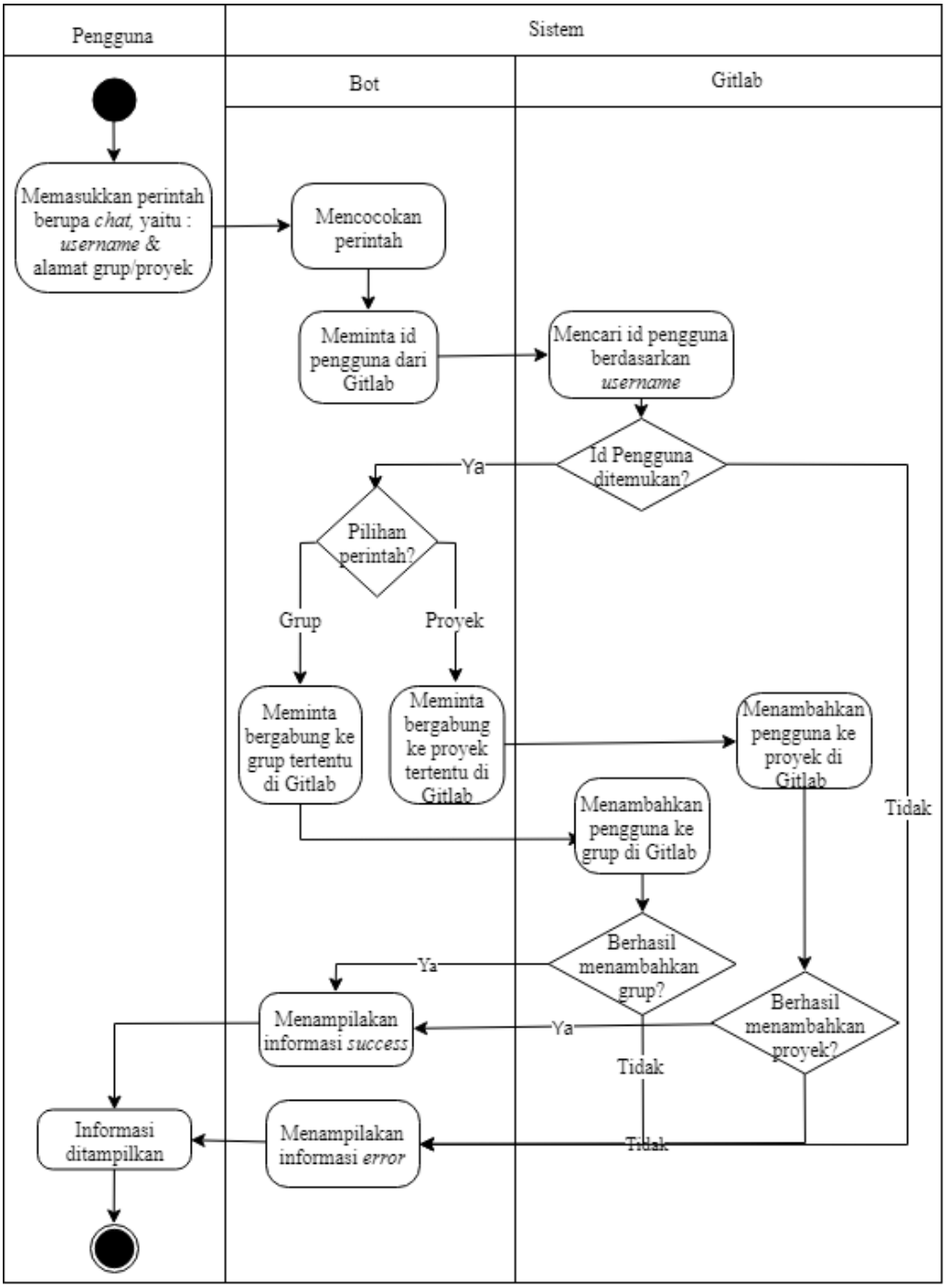
Gambar 3. 9 Diagram aktivitas membagi proyek ke grup Gitlab

Pengguna dapat memasukan perintah, berupa alamat proyek dan alamat grup yang dituju. Kemudian sistem akan mencocokkan perintah dan mengirim permintaan ke Gitlab untuk memindahkan proyek ke grup yang dimasukan. Gitlab akan memproses permintaan dan

mengirim data hasil proses pembagian. Data akan dikirim ke sistem dan kemudian akan ditampilkan ke Slack oleh sistem.

d. Diagram aktivitas bergabung ke grup atau proyek tertentu di Gitlab

Gambar 3. 10 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat bergabung ke proyek atau grup Gitlab melalui perintah di *chat*.

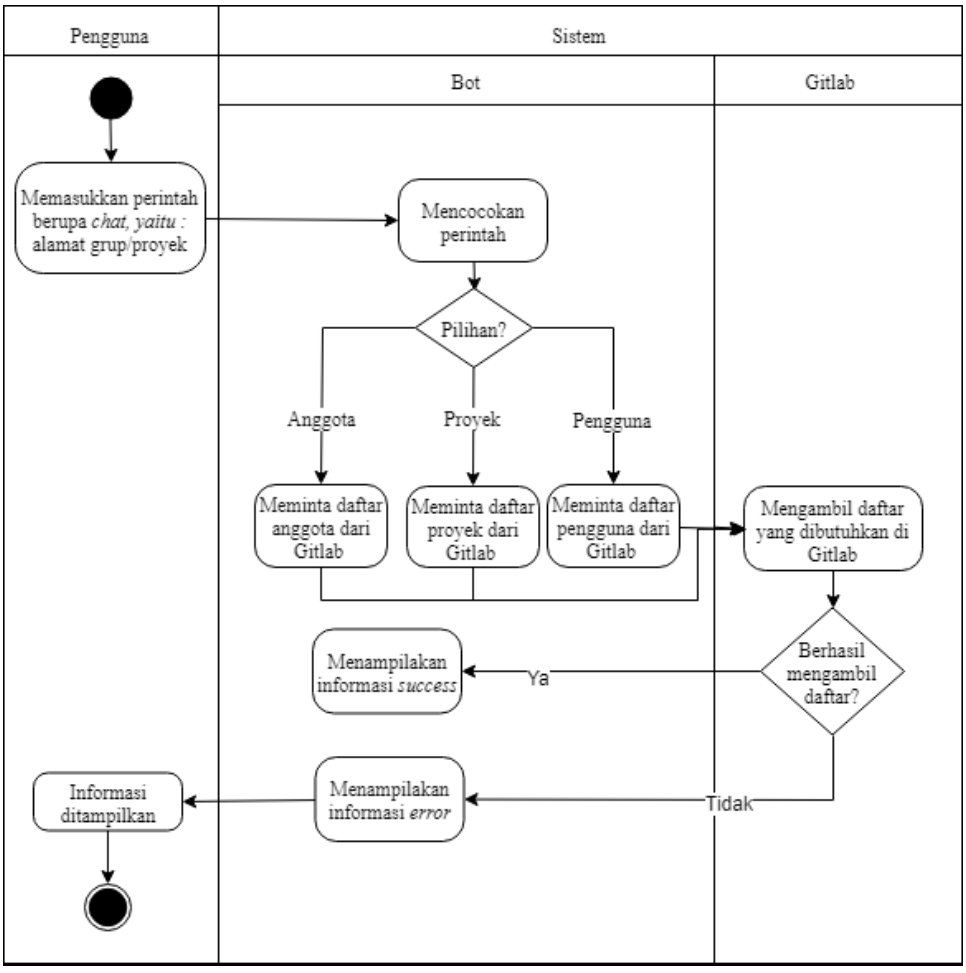


Gambar 3. 10 Diagram aktivitas bergabung ke grup atau proyek tertentu di Gitlab

Pengguna dapat memasukan perintah, berupa *username* akun, alamat proyek atau alamat grup yang dituju. Kemudian sistem akan mencocokkan perintah dan mengirim permintaan ke Gitlab untuk mencari id pengguna yang sesuai berdasarkan *username* yang dimasukan.Gitlab akan memproses permintaan dan mengirim data id yang dibutuhkan. Setelah itu sistem akan mengirim permintaan untuk dapat bergabung ke grup atau proyek yang dimasukan sebelumnya. Gitlab akan menanggapi proses dari permintaan sistem dan mengirim hasil proses bergabungnya pengguna ke grup atau proyek yang diinginkan. Data akan diterima sistem dan kemudian akan ditampilkan Slack oleh sistem.

e. Diagram aktivitas melihat daftar pengguna proyek, anggota grup dan daftar proyek di grup Gitlab

Gambar 3. 11 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat melihat daftar pengguna proyek, anggota grup dan daftar proyek di grup Gitlab melalui perintah di *chat*.

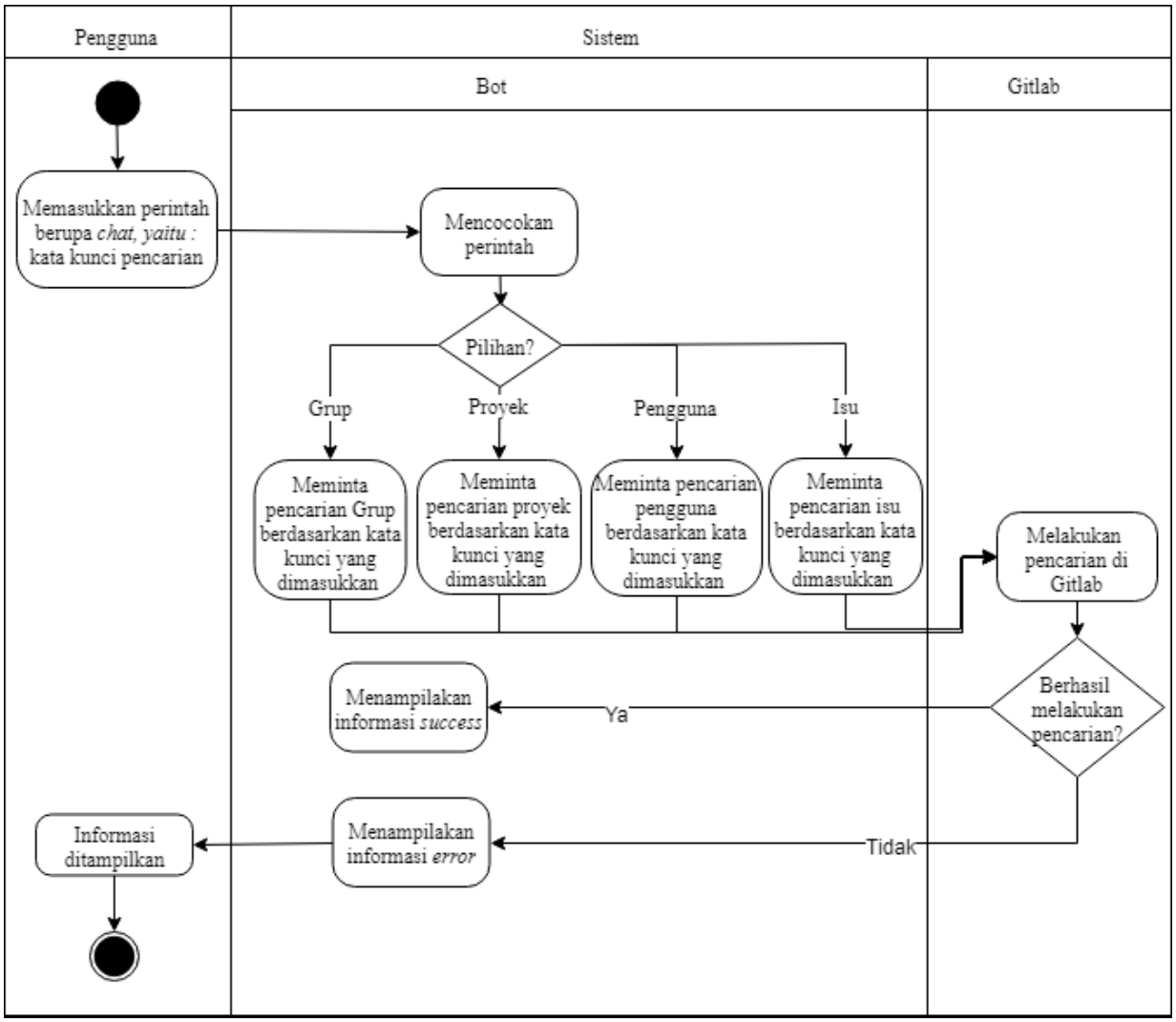


Gambar 3. 11 Diagram aktivitas melihat daftar tertentu di Gitlab

Pengguna dapat memasukan perintah, berupa alamat proyek atau alamat grup sesuai kebutuhan. Kemudian sistem akan mencocokkan perintah dan mengirim permintaan ke Gitlab untuk meminta daftar anggota proyek atau pengguna yang dibutuhkan sesuai dengan perintah yang dimasukan sebelumnya. Gitlab akan memproses permintaan dan mengirim data yang dibutuhkan. Data akan diterima sistem dan kemudian akan ditampilkan ke Slack oleh sistem.

f. Diagram aktivitas melakukan pencarian di Gitlab

Gambar 3. 12 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat melakukan pencarian di Gitlab melalui perintah di *chat*.



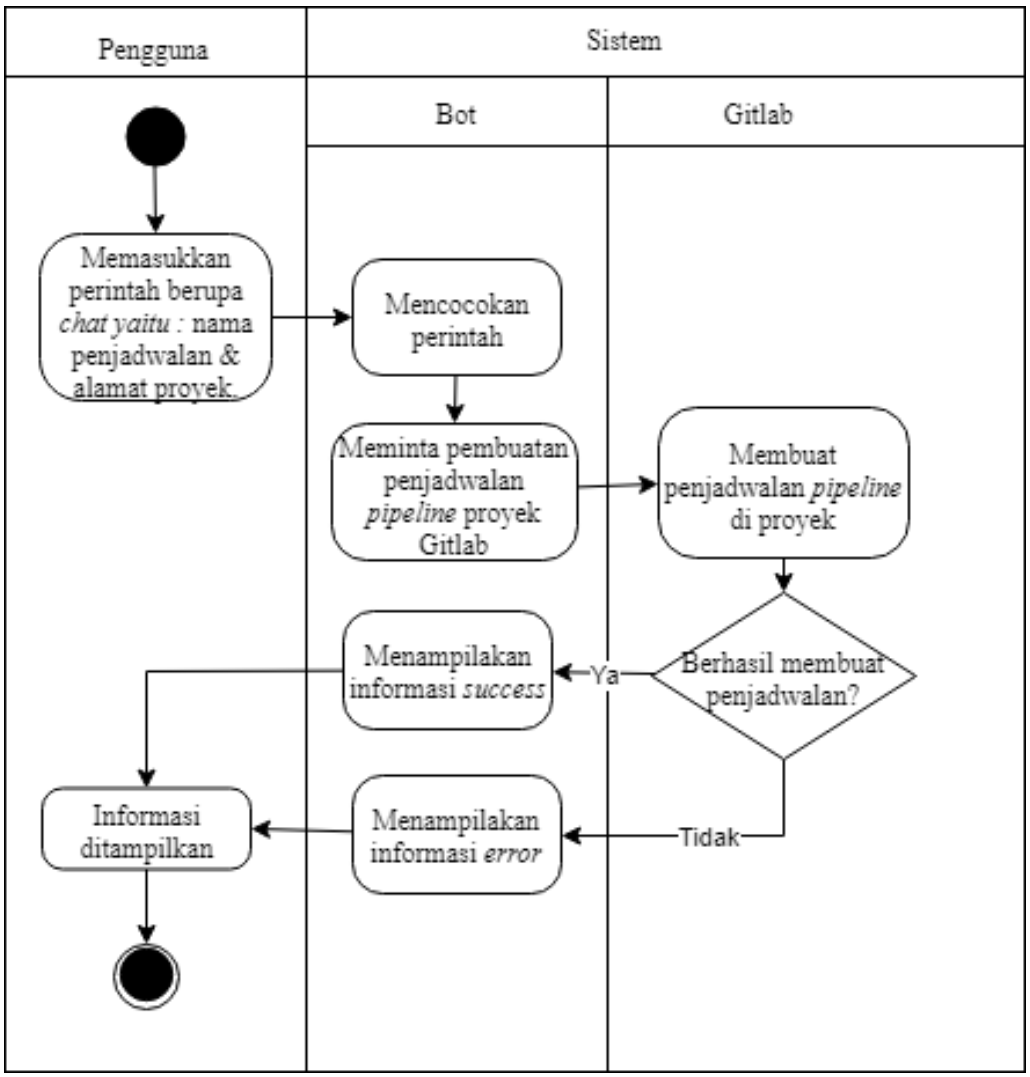
Gambar 3. 12 Diagram aktivitas melakukan pencarian di Gitlab

Pengguna dapat memasukan perintah, berupa kata kunci yang dicari. Kemudian sistem mencocokkan perintah dan mengirim permintaan ke Gitlab untuk melakukan pencarian

berdasarkan kata kunci yang dimasukan. Gitlab memproses permintaan dan mengirim data yang dibutuhkan.

g. Diagram aktivitas membuat penjadwalan *pipeline* di Gitlab

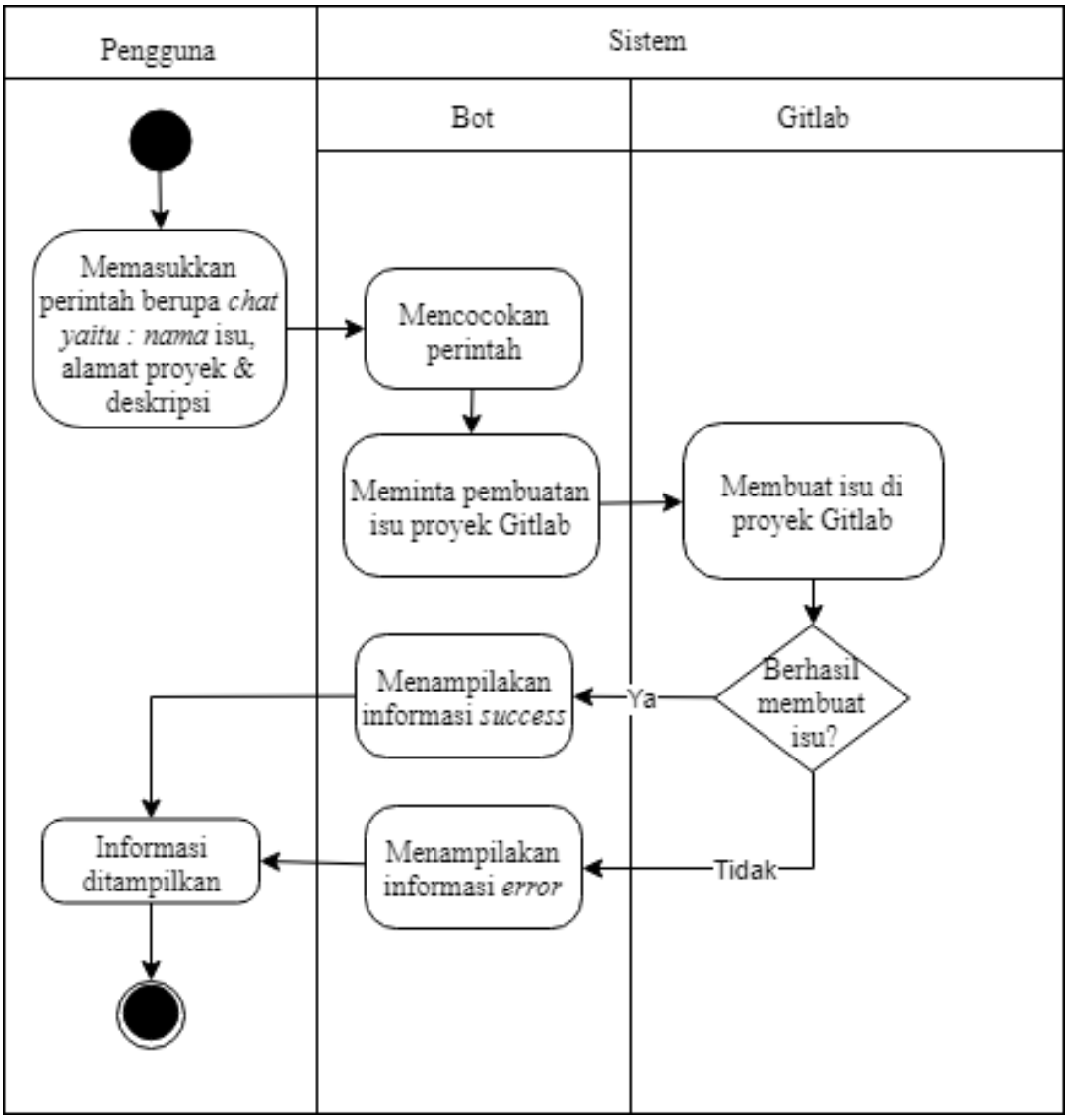
Gambar 3. 13 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat membuat penjadwalan *pipeline* ke suatu proyek di Gitlab melalui perintah di chat. Pengguna dapat memasukan perintah, berupa nama untuk penjadwalan serta alamat proyek. Kemudian sistem mencocokkan perintah dan mengirim permintaan ke Gitlab untuk melakukan pembuatan jadwal berdasarkan data yang dimasukan. Gitlab memproses permintaan dan mengirim data yang dibutuhkan.



Gambar 3. 13 Diagram aktivitas membuat penjadwalan *pipeline* di Gitlab

h. Diagram aktivitas membuat isu di Gitlab

Gambar 3. 14 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat membuat isu baru ke suatu proyek yang dituju melalui perintah di *chat*. Pengguna dapat memasukan perintah, berupa nama isu, alamat proyek serta deskripsi dari isu yang akan dibuat. Kemudian sistem mencocokkan perintah dan mengirim permintaan ke Gitlab untuk melakukan pembuatan isu berdasarkan data yang dimasukan. Gitlab memproses permintaan dan mengirim data yang dibutuhkan.

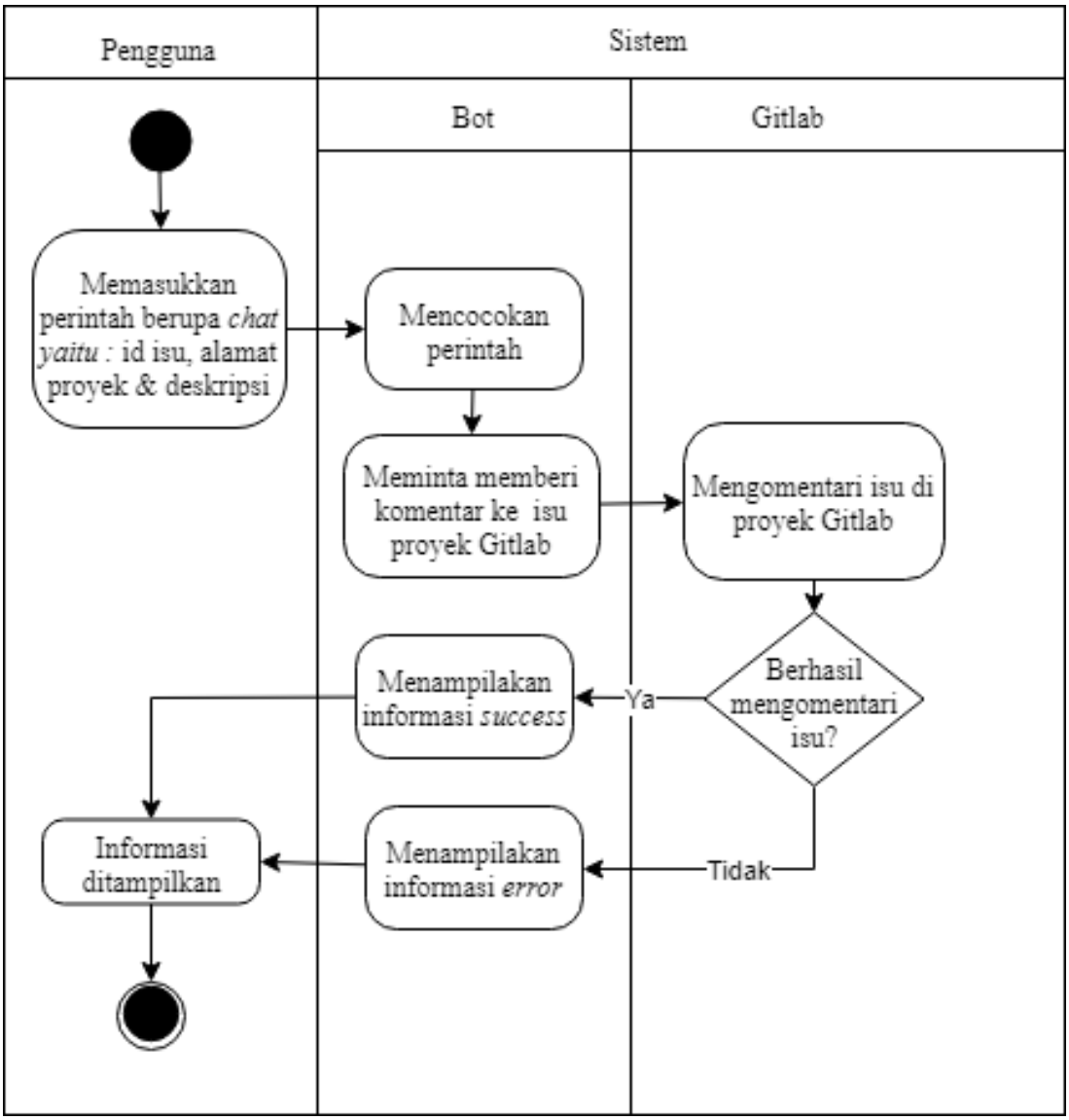


Gambar 3. 14 Diagram aktivitas membuat isu di Gitlab

i. Diagram aktivitas mengomentari isu di Gitlab

Gambar 3. 15 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat memberi komentar terhadap suatu isu di proyek tertentu dalam Gitlab melalui perintah di *chat*. Pengguna

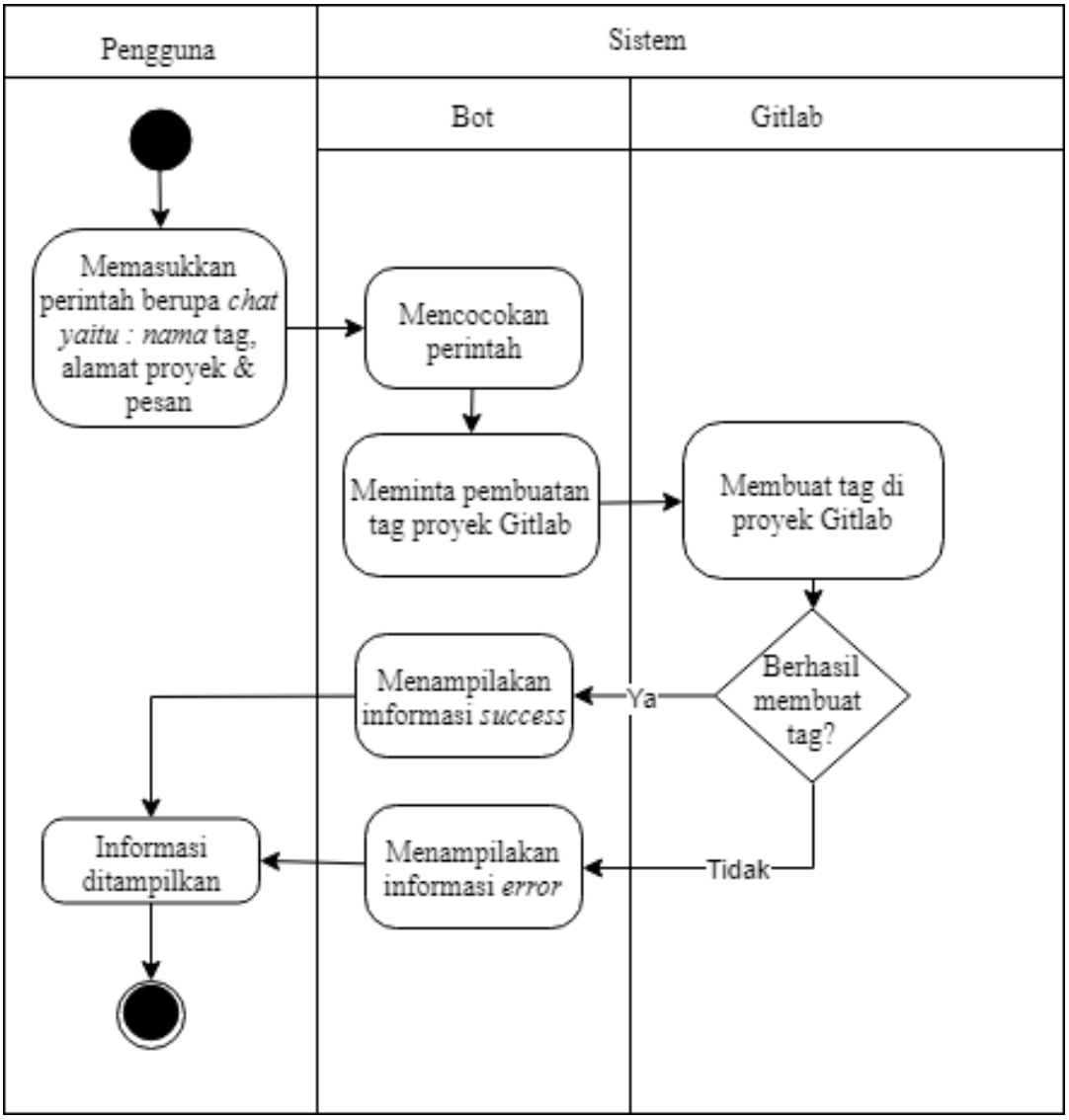
dapat memasukan perintah, berupa id dari isu yang ingin dikomentari, alamat proyek dari isu serta deskripsi dari isi komentar. Kemudian sistem mencocokkan perintah dan mengirim permintaan ke Gitlab untuk mengomentar isu berdasarkan data yang dimasukan. Gitlab memproses permintaan dan mengirim data yang dibutuhkan.



Gambar 3. 15 Diagram aktivitas mengomentari isu di Gitlab

j. Diagram aktivitas membuat tag di Gitlab

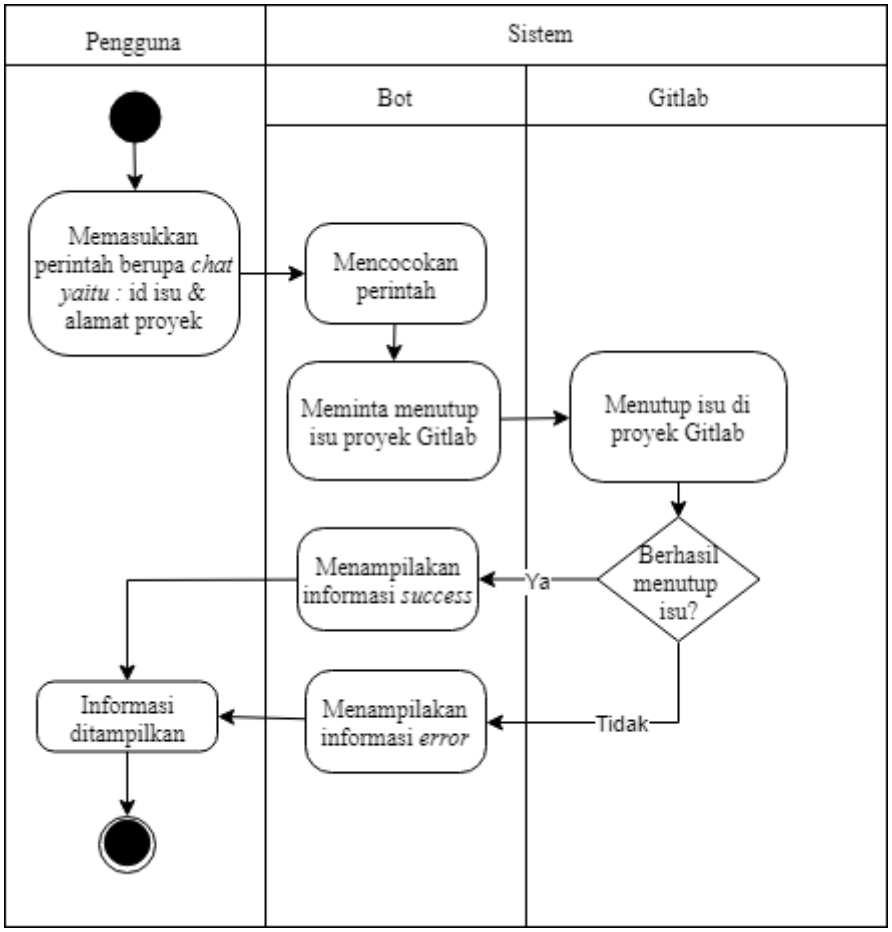
Gambar 3. 16 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat membuat tag baru di proyek tertentu dalam Gitlab melalui perintah di *chat*. Pengguna dapat memasukan perintah, berupa nama tag, alamat proyek serta pesan. Kemudian sistem mencocokkan perintah dan mengirim permintaan ke Gitlab untuk membuat tag berdasarkan data yang dimasukan. Gitlab memproses permintaan dan mengirim data yang dibutuhkan.



Gambar 3. 16 Diagram aktivitas membuat tag di Gitlab

k. Diagram aktivitas menutup isu di Gitlab

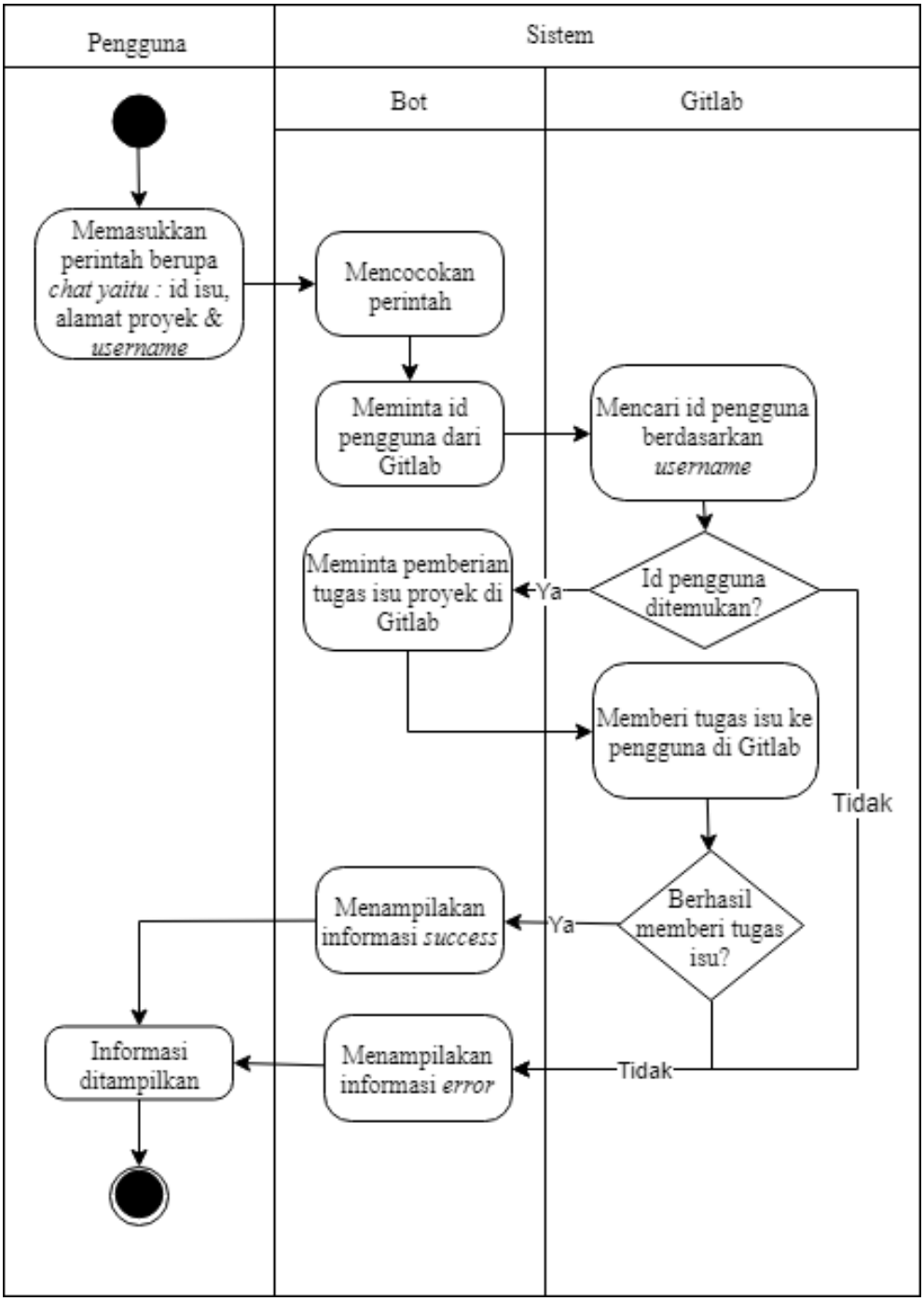
Gambar 3. 17 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat menutup isu di proyek tertentu dalam Gitlab melalui perintah di *chat*. Pengguna dapat memasukan perintah, berupa isu id dan alamat proyek. Kemudian sistem mencocokkan perintah dan mengirim permintaan ke Gitlab untuk menutup isu berdasarkan data yang dimasukan. Gitlab memproses permintaan dan mengirim data yang dibutuhkan.



Gambar 3. 17 Diagram aktivitas menutup isu di Gitlab

l. Diagram aktivitas memberi tugas isu ke pengguna di Gitlab

Gambar 3. 18 menjelaskan aktivitas yang dilakukan pengguna Slack untuk dapat memberikan tugas isu ke pengguna di Gitlab melalui perintah di *chat*. Pengguna dapat memasukan perintah, berupa id isu, alamat proyek *username* akun Gitlab. Kemudian sistem akan mencocokkan perintah dan mengirim permintaan ke Gitlab untuk mendapatkan id pengguna berdasarkan *username* yang dimasukan. Gitlab akan memproses permintaan dan mengirim data yang dibutuhkan untuk mengirim tugas isu proyek di Gitlab. Setelah itu Sistem akan mengirim permintaan untuk memberi tugas isu proyek ke pengguna, jika data yang diperlukan terpenuhi. Gitlab akan memproses permintaan pemberian tugas isu berdasarkan data masukan sebelumnya. Data hasil proses pemberian tugas isu akan dikirim ke sistem dan kemudian akan ditampilkan ke Slack oleh sistem.



Gambar 3. 18 Diagram aktivitas memberi tugas isu ke pengguna

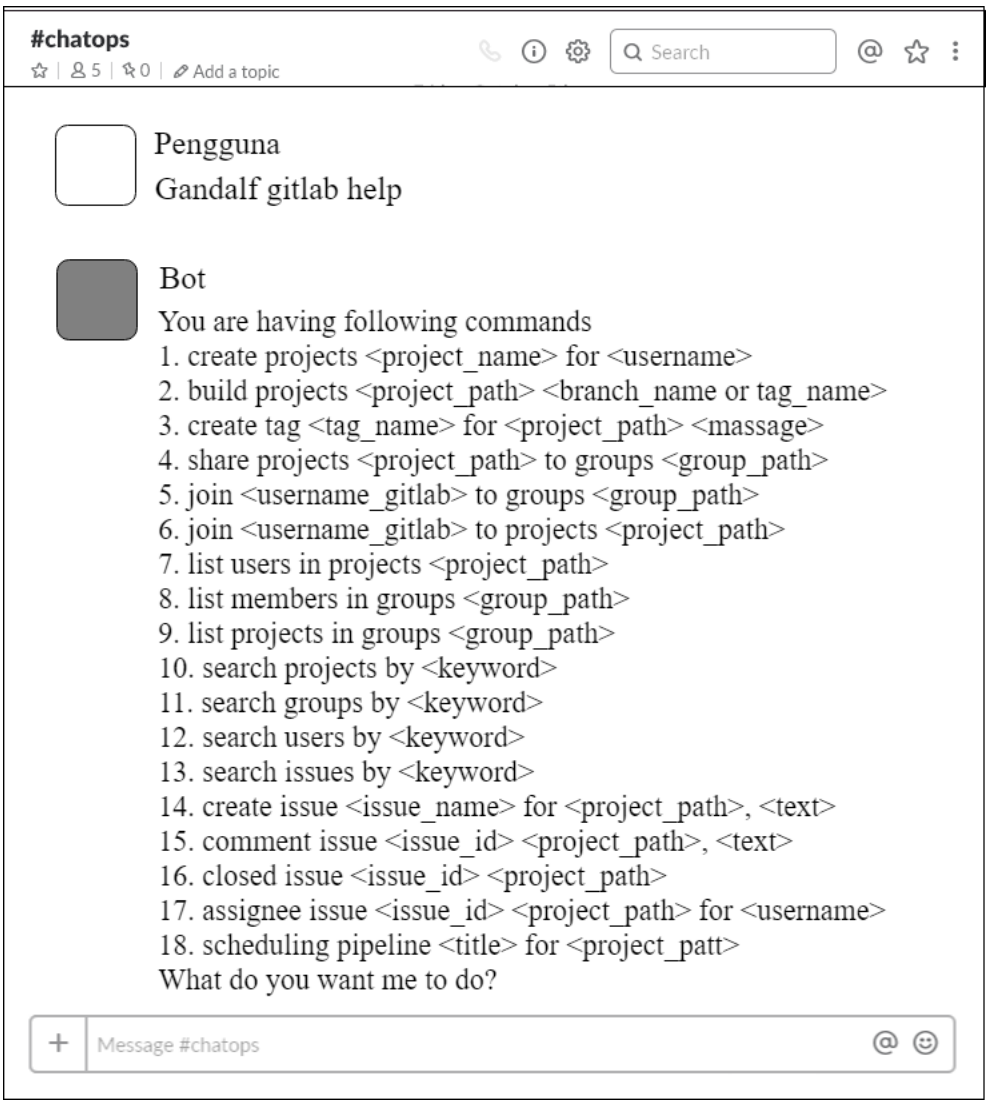
3.2.4 Rancangan Antarmuka

Antarmuka merupakan sesuatu yang penting untuk mempermudah pengguna dalam berinteraksi dengan sistem. *Conversation user interface* (CUI) adalah antarmuka yang diterapkan Chatbot kepada pengguna di dalam *platform* pesan Messenger. Penulis tidak membuat aplikasi, melainkan menggunakan aplikasi Messenger yang sudah tersedia di Slack. Penulis membuat CUI ini bertujuan agar komunikasi antara Chatbot dan pengguna terjadi

dengan interaktif. Selain dapat berintaksi dengan cara memasukkan perintah berupa oesan, pengguna juga dapat memanfaatkan fungsionalitas yang ada di dalamnya. Ada beberapa rancangan yang penulis buat dalam perancangan antarmuka Chatbot ini. Diantaranya adalah rancangan *manual book* dan rancangan perintah *Chatbot*.

a. Rancangan *manual book*

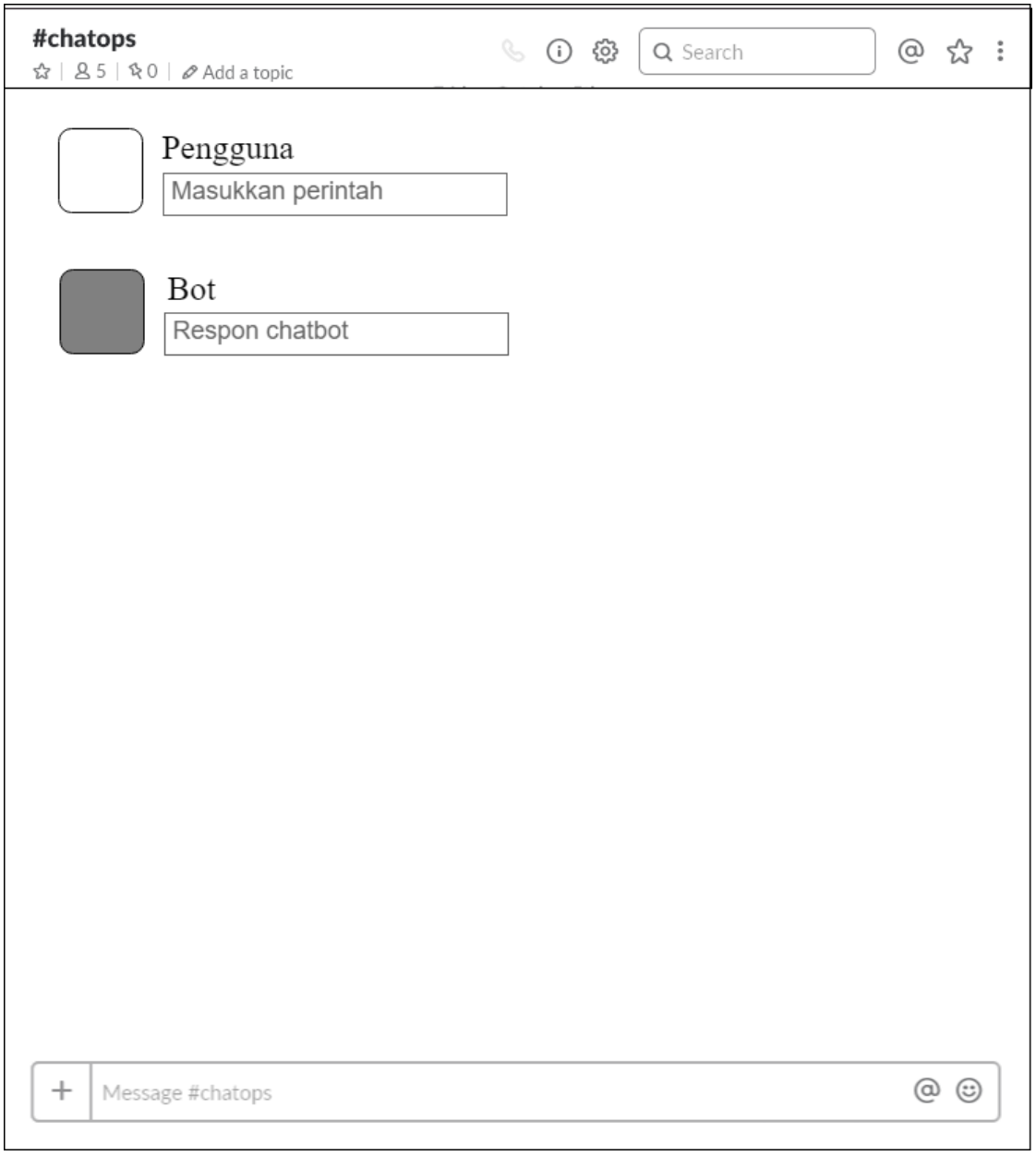
Pada rancangan antarmuka ini, Chatbot akan menampilkan balasan pesan berupa informasi yang diperoleh dari Gitlab. Pada rancangan ini merupakan representasi dari antarmuka Slack. Berikut merupakan rancangan *manual book* yang berisi daftar perintah yang dapat digunakan untuk berinteraksi dengan Chatbot. Daftar perintah dapat diakses dengan menggunakan perintah “Gandalf gitlab help”, yang kemudian akan tampil seperti pada Gambar 3. 19.



Gambar 3. 19 Rancangan *manual book*

b. Rancangan perintah *Chatbot*

Pada rancangan *manual book* pada Gambar 3. 19 menunjukkan perintah yang dapat digunakan untuk mengakses Chatbot, daftar perintah tersebut akan terhubung ke Gitlab. Rancangan antarmuka penggunaan setiap perintah yang terdapat pada daftar di rancang sama. Sehingga rancangan dapat dilihat pada Gambar 3. 20.



Gambar 3. 20 Rancangan perintah *Chatbot*

Berikut ini daftar perintah yang dapat digunakan pada sistem, yang ditunjukkan pada Tabel 3. 2.

Tabel 3. 2 Daftar Perintah Sistem ChatOps

No	*Use Case*	Aktivitas Sistem	Perintah
1	Membuat proyek di Gitlab	Membuat proyek	create projects <project_name> for <username>
2	*Build* proyek di Gitlab	*Build* proyek	build projects <project_path> <branch_name or tag_name>
3	Bergabung ke proyek Gitlab	Bergabung ke proyek	join <username_gitlab> to projects <project_path>
4	Melihat daftar pengguna proyek di Gitlab	Melihat daftar pengguna proyek	list users in projects <project_path>
5	Bergabung ke grup Gitlab	Bergabung ke grup	join <username_gitlab> to groups <group_path>
6	Melihat daftar anggota grup di Gitlab	Melihat daftar anggota grup	list members in groups <group_path>
7	Penjadwalan *pipeline* di Gitlab	Penjadwalan *pipeline*	scheduling pipeline <title> for <project_path>
8	Melihat daftar proyek grup di Gitlab	Melihat daftar proyek grup	list projects in groups <group_path>
9	Memindahkan proyek ke grup di Gitlab	Memindahkan proyek ke grup	share projects <project_path> to groups <group_path>
10	Melakukan pencarian di Gitab	Melakukan pencarian isu	search issues by <keyword>
		Melakukan pencarian pengguna	search users by <keyword>
		Melakukan pencarian grup	search groups by <keyword>
		Melakukan pencarian proyek	search projects by <keyword>
11	Membuat isu di Gitlab	Membuat isu	create issue <issue_name> for <project_path>, <text>
12	Mengomentari isu di Gitlab	Mengomentari isu	comment issue <issue_id> <project_path>, <text>
13	Mengirim tugas isu ke pengguna di Gitlab	Mengirim tugas isu ke pengguna	assignee issue <issue_id> <project_path> for <username>
14	Menutup isu di Gitlab	Menutup isu	closed issue <issue_id> <project_path>
15	Membuat tag baru di Gitlab	Membuat tag baru	create tag <tag_name> for <project_path> <massage>

3.3 Rencana Pengujian

Pengujian sistem dilakukan untuk menemukan kesalahan yang mungkin terjadi di sistem, kemudian dapat diperbaiki menjadi lebih baik. Sistem yang dikembangkan harus mampu beroperasi sesuai dengan kebutuhan pengguna. Rencana pengujian aplikasi dibagi menjadi dua bagian yaitu pengujian fungsionalitas dan pengujian usabilitas.

3.3.1 Pengujian Fungsionalitas

Pengujian fungsionalitas ini menggunakan metode *blackbox testing* yang dilakukan oleh penulis. Pengujian dilakukan dengan melakukan serangkaian proses operasi yang sesuai dengan skenario yang telah dibuat. *Blackbox* berfokus pada bagaimana cara pengoperasian sistem, apakah masukan data akan menghasilkan output yang diharapkan. Pengujian ini berusaha untuk menemukan kesalahan dalam fungsi-fungsi yang telah dirancang sudah sesuai atau belum.

3.3.2 Pengujian Usabilitas

Pengujian ini mengukur tingkat efektif, efisien dan kepuasan pengguna dalam menggunakan fungsionalitas dari sistem. Terdapat lima unsur pokok dari pengujian usabilitas yaitu (Handiwidjojo & Ernawati, 2016):

a. *Learnability*

   Berkaitan dengan kemudahan suatu sistem dalam penggunaannya, diukur dari pemakaian fungsi-fungsi dan fitur dalam sistem yang diuji.

b. *Efficiency*

   Berkaitan dengan kecepatan dalam pengerjaan tugas tertentu, dalam beberapa kasus yang dirancang dalam skenario.

c. *Memorability*

   Berkaitan dengan kemampuan pengguna dalam mempertahankan pengetahuannya, untuk mengulang beberapa perintah yang sama dalam skenario.

d. *Error*s

   Berkaitan dengan kesalahan-kesalahan yang dibuat oleh pengguna, dan bagaimana sistem menanggapi kesalahan tersebut.

e. *Satisfaction*

   Berkaitan dengan kepuasan pengguna dalam menggunakan sistem, pengukuran dilakukan dengan melihat dampak dari adanya sistem seperti manfaat yang didapatkan.

Metode Pengumpulan Data

Dalam penelitian ini metode pengumpulan data yang akan digunakan adalah kuesioner, dalam bentuk pertanyaan pilihan ganda dan bersifat tertutup. Skala *Likert* digunakan untuk pengukuran variabel penelitian, dibuat menggunakan skala 1 sampai 5, dengan rincian:

a. Jawaban SS (Sangat Setuju) diberi nilai 5
b. Jawaban S (Setuju) diberi nilai 4

c. Jawaban N (Netral) diberi nilai 3
d. Jawaban TS (Tidak Setuju) diberi nilai 2
e. Jawaban STS (Sangat Tidak Setuju) diberi nilai 1

Hal-hal yang ingin ditanyakan dalam kuisoner penelitian ini meliputi fungsionalitas dari sistem ChatOps yang bangun. Berikut ini adalah rancangan kuesioner yang akan dibagikan:

a. Apakah sistem baru lebih efektif?
b. Apakah sistem mudah digunakan?
c. Dapat belajar dengan cepat ketika menggunakannya?
d. Apakah sistem baru membantu lebih produktif?
e. Apakah sistem baru menghemat waktu penyelesaian kasus?
f. Apakah sistem meningkatkan kolaborasi tim?
g. Apakah mudah mengingat cara menggunakannya?
h. Apakah bisa menggunakannya tanpa instruksi tertulis?
i. Apakah notifikasi error membantu memperbaiki kesalahan?
j. Apakah puas dengan sistem ini?

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi ChatOps

Pada tahap ini penulis akan menjelaskan mengenai pembangunan Chatbot yang di integrasikan dengan Gitlab dan Slack, sesuai dengan skenario perancangan yang sudah dibuat. Ada beberapa persiapan dan tahapan yang dilakukan penulis dalam membangun ChatOps antara lain, pengaturan *collaboration tools*, membangun ChatOps dengan mengintegrasikan Gitlab dan Slack serta membangun kode integrasi.

4.1.1 Pengaturan *Collaboration Tools*

DevOps memiliki banyak *collaboration tools* yang digunakan untuk mendukung perilisan *perangkat lunak* yang dibangun, Gitlab dan Slack adalah salah satu *tools* yang digunakan. Gitlab digunakan untuk manajer repositori yang mendukung CI/CD, sedangkan Slack merupakan pusat kolaborasi yang menghubungkan setiap tim untuk dapat menyelesaikan berbagai hal dengan media *Platfrom chat*. Sebelum mengintegrasikan Chatbot dengan Gitlab maupun Slack, perlu persiapan terlebih dahulu yaitu:

a. Akun admin Gitlab: Akun ini merupakan akun yang memiliki otoritas tinggi dalam mengelola Gitlab, sehingga dapat mengintegrasikan beberapa perintah penting untuk kolaborasi ChatOps.
b. *Channel* untuk kolaborasi: Membuat *channel* baru di dalam Slack yang akan digunakan untuk seluruh tim untuk dapat berkolaborasi dengan ChatOps.
c. Aplikasi Hubot di Slack: Menambahkan aplikasi Hubot di dalam Slack, sebagai bot yang akan diintegrasikan dengan Gitlab dan ditambahkan dalam *channel* kolaborasi.
d. Slack *token*: untuk dapat berkomunikasi dengan Hubot yang akan dibangun diperlukan *token* Slack untuk berintegrasi, *token* ini didapatkan saat menambahkan aplikasi Hubot ke Slack.
e. Akses *token*: sedangkan untuk dapat terhubung dengan Gitlab, diperlukan akses *token* pengguna yang dapat diregenerate oleh setiap pengguna yang menginginkannya.

4.1.2 Membangun ChatOps dengan mengintegrasikan Gitlab dan Slack

ChatOps dikonfigurasi menggunakan Hubot yang dibangun menggunakan *container* di Docker. Hubot menggunakan NodeJs untuk dapat menginstall berbagai paket npm yang diperlukan. Seperti yang ditunjukkan dalam konfigurasi Dockerfile pada Gambar 4. 1.

```
FROM node:carbon
RUN npm install -g hubot coffee-script  yo generator-hubot

RUN yo hubot --owner="BSI" --name="Gandalf" --description="Delightfully aware robutt" --adapter=slack --defaults --allow-root

RUN npm install hubot-slack
RUN yarn add hubot-command-log
RUN npm i hubot-command-log
```

Gambar 4. 1 Konfigurasi Docker

Di dalam Gambar 4. 1 dapat dilihat versi NodeJs yang digunakan adalah carbon, kemudian beberapa paket npm diinstall untuk mengkonfigurasi Hubot. Paket-paket ini diantaranya, hubot, coffe-*script*, yo, generator-hubot, hubot-slack serta hubot-command-log. Hubot-slack dipasang sebagai adapter yang akan diintegrasikan dengan Hubot, dapat dilihat juga dalam Gambar 4. 1 kode pengaturan adapter Hubot yang digunakan. Sedangkan Hubot-command-log dipasang sebagai monitoring Hubot saat dijalankan, untuk mempermudah identifikasi masalah saat integrasi dan kolaborasi. Pemberian nama Hubot harus sesuai dengan pengaturan nama Hubot saat menambahkan aplikasi di Slack, sehingga dapat terintegrasi dengan baik.

Untuk dapat terhubung dengan workspace *channel* Slack yang dibuat, maka diperlukan Slack *token* yang dikonfigurasi dalam pembangunan Hubot, seperti dalam Gambar 4. 2.

```
ENV HUBOT_SLACK_TOKEN=xoxb-xxxxxxxxxxxxxxx-xxxxxxxxxxx-xxxxxxxxxx
```

Gambar 4. 2 Konfigurasi Slack Token

Variabel “ENV HUBOT_SLACK_TOKEN” dimasukan dalam konfigurasi Dockerfile. Juga terdapat konfigurasi berkas docker-compose.yml yang digunakan untuk membangun konfigurasi Hubot di Dockerfile menjadi lebih mudah. Konfigurasi docker-compose.yaml ditunjukkan pada Gambar 4. 3.

```
version: "2"
services:
  hubot-slack:
    build:
      context: .
    restart: always
    user: hubot-slack
    environment:
      - HUBOT_GITLAB_URL=http://gitlab.[redacted].id/api/v4
      - HUBOT_GITLAB_TOKEN=[redacted]
```

Gambar 4. 3 Konfigurasi Docker Compose

Konfigurasi yang terdapat dalam berkas docker-compose.yml pada Gambar 4. 3 diantaranya, penambahan *environment* Gitlab *url* dan *token* yang diperlukan untuk integrasi saat melakukan kolaborasi menggunakan Hubot melalui *http request*. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File Dockerfile dan File docker-compose.yml).

4.1.3 ChatOps untuk Manajemen Kejadian dan Notifikasi

Notifikasi disalurkan ke Slack seperti pada skenario sistem sebelumnya, yang ditunjukkan pada Gambar 3. 3. Notifikasi digunakan sebagai peringatan jika terjadi perubahan dalam Gitlab, seperti halnya jika ada proses *pipeline* gagal dijalankan. Bot dapat dirancang untuk dapat menjalankan kembali *job* yang gagal dijalankan setelah selesai diperbaiki, sebagai tanggapan dari peringatan yang diberikan oleh notifikasi. Seperti pada Gambar 4. 4 yang menunjukkan kode untuk *build* ulang suatu proyek jika terjadi suatu masalah.

```
url = process.env.HUBOT_GITLAB_URL
git_token = process.env.HUBOT_GITLAB_TOKEN

module.exports = (robot) ->
robot.respond /build projects (.*) branch (.*)/i, (msg) ->
    projectname = msg.match[1]
    branch = msg.match[2]
    project = projectname.split('/').join('%2F');
```

Gambar 4. 4 *Script* kode *build* ulang

Pada tahap awal dilakukan inisialisasi pada variabel url dan git_token yang mengarah pada variabel *envirotment* di pengaturan berkas konfigurasi pembangunan sistem ChatOps pada Gambar 4. 5. Bot akan merespon ketika masukan sudah sesuai dengan *regex* yang dikonfigurasi atau belum. Masukan akan diidentifikasi sebagai alamat proyek dan nama branch.

```
path                                                                    =
"#{url}/projects/#{project}/triggers?private_token=#{git_token}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
```

Gambar 4. 5 *Script http request* menggunakan API Gitlab

Komunikasi ChatOps dengan Gitlab menggunakan *http request response*, ditunjukkan pada variabel path yang berisi alamat lengkap pemanggilan API Gitlab untuk mendapatkan data *trigger* dari proyek. Tanggapan dari data yang berbentuk JSON, diambil dan digunakan untuk inisialisasi variabel yang diperlukan seperti *token*. Konfigurasi path pada Gambar 4. 6 meminta proses *http request* dari Gitlab API untuk *build* proyek menggunakan *trigger* dengan *token* yang sudah diidentifikasi.

```
if 200 <= res.statusCode < 400
              content = JSON.parse body
              token = content.token
              path =
"#{url}/projects/#{project}/trigger/pipeline?token=#{token}"
              req = msg.http(path)
              req.query(_render: "json", ref: branch)
              req.header('Content-Length', 0)
              req.post() (err, res, body) ->
```

Gambar 4. 6 *Script build* proyek menggunakan *token trigger*

Pada beberapa kasus belum terdapat *trigger* pada proyeknya, maka proses pembuatan *trigger* perlu dilakukan terlebih dahulu, seperti pada Gambar 4. 7.

```
path =
"#{url}/projects/#{project}/triggers?private_token=#{git_toke
n}"
          req = msg.http(path)
          req.query(_render: "json", descript ion: branch)
          req.header('Content-Length', 0)
          req.post() (err, res, body) ->
```

Gambar 4. 7 *Script* permintaan pembuatan *trigger*

Pada beberapa kasus build proyek, terdapat kegagalan dikarenakan proyek yang sudah ada terdapat lebih dari satu branch yang digunakan, untuk itu proses build dilakukan dengan membuat tag baru di proyek yang akan dijalankan. *Script* pembuatan tag baru dapat dilihat di

Gambar 4. 8. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File re-build_gitlab.coffee).

```
robot.respond /create tag (.*) for (.*) (.*)/i, (msg) ->
    tagname = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F');
    massage = msg.match[3]

    path =
"#{url}/projects/#{project}/repository/tags?private_token=#{git_to
ken}&tag_name=#{tagname}&ref=master&message=#{massage}"
    req = msg.http(path)
    req.query(_render: "json")
    req.header('Content-Length', 0)
    req.post()
```

Gambar 4. 8 *Script* pembuatan tag baru

4.1.4 ChatOps untuk Manajemen Masalah

Perekaman suatu insiden baru atau *error* perlu dilakukan dan disimpan untuk mengatasi masalah yang sama jika terjadi kembali. Penanganan terhadap kesalahan yang terjadi juga dapat di bawa dalam forum untuk dapat saling menganalisis dan pro aktif terhadap insiden yang terjadi. Seperti pada Gambar 4. 9 pembuatan isu dan pengelolaannya dalam Gitlab, yang dapat dikonfigurasi melalui Hubot.

```
robot.respond /create issue (.*) for (.*) desc (.*)/i, (msg) ->
    issuename = msg.match[1]
    projectname = msg.match[2]
    de script ion = msg.match[3]
    project = projectname.split('/').join('%2F')
    issue = issuename.split(' ').join('%20');
    path =
"#{url}/projects/#{project}/issues?private_token=#{git_token}&titl
e=#{isu}&de scription=#{de scription}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.post() (err,res,body) ->
```

Gambar 4. 9 *Script* Pembuatan isu Baru

Regex akan menseleksi masukan yang sesuai dengan kode yang telah dikonfigurasi. Dimana kode akan mengidentifikasi nama isu yang diberikan, alamat proyek serta isi dari isu yang dibuat. Path menunjukkan proses *request http* untuk dapat membuat isu baru dari proyek

yang menjadi tujuan. Bot juga dapat memberikan komentar pada isu tertentu dengan meminta Gitlab API untuk membuat note pada isu yang dituju, seperti pada Gambar 4. 10.

```
robot.respond /comment issue (.*) for (.*) desc (.*)/i, (msg) ->
    issueid = msg.match[1]
    projectname = msg.match[2]
    description = msg.match[3]
    project = projectname.split('/').join('%2F');
    path = "#{url}/projects/#{project}/ issue/#{
issueid}/notes?private_token=#{git_token}&body=#{description}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.post() (err,res,body) ->
```

Gambar 4. 10 *Script* memberi komentar isu

Setelah melakukan diskusi pada forum isu yang dibuat, ketika menemukan solusi untuk penyelesaian masalah, maka isu dapat ditutup oleh pengguna yang ditunjukkan pada Gambar 4. 11. Ada beberapa isu yang kesalahan terjadi pada bagian tertentu yang merupakan tanggung jawab dari tim atau pengguna lain, maka tuga penyelesaian isu dapat diberikan ke pengguna tersebut.

```
robot.respond /closed issue (.*) (.*)/i, (msg) ->
    id = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F')
    path =
"#{url}/projects/#{project}/issues/#{id}?private_token=#{git_token
}&state_event=close"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.put()
```

Gambar 4. 11 *Script* menutup sebuah isu

Sedangkan *script* untuk memberikan tugas ke pengguna dapat dilihat di Gambar 4. 12. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File issue-scheduling.coffee).

```
robot.respond /assignee issue (.*) (.*) for (.*)/i, (msg) ->
    id = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F')
    username = msg.match[3]
    user_id = user.id
    path =
"#{url}/projects/#{project}/issues/#{id}?private_token=#{git_
token}&assignee_ids=#{user_id}"
            #msg.send path
            req = msg.http(path)
            req.header('Content-Length', 0)
            req.put()
```

Gambar 4. 12 *Script* memberi tugas isu ke pengguna

4.1.5 ChatOps untuk Manajemen perilisan dengan *Continuous Delivery*

Penggunaan bot untuk mendukung CD sangat sering digunakan dalam pembangunan ChatOps, karena banyaknya *collaboration tools* dalam DevOps yang mendukung teknologi CD ini. Seperti halnya pada Gitlab yang dapat di otomasikan beberapa proses terkait *pipeline* proyek dan grup.

a. Penjadwalan *pipeline*

Dalam skenario yang sudah dirancang, terdapat beberapa job proyek tertentu yang harus dijalankan dalam kurun waktu yang ditentukan atau disebut penjadwalan *pipeline*. Proses ini dapat dikonfigurasi di Hubot, dalam bentuk kode seperti Gambar 4. 13. Path menginisialisasi alamat *http request* yang digunakan untuk meminta API Gitlab membuat penjadwalan *pipeline* sesuai dengan masukan yang telah diidentifikasi oleh *regex*. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File issue-scheduling.coffee).

```
robot.respond /scheduling pipeline (.*) for (.*)/i, (msg) ->
    pipeline = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F');
    path =
"#{url}/projects/#{project}/pipeline_schedules?private_token=#{git
_token}&description=#{pipeline}&ref=master&active=true&cron_timezo
ne=Asia/Jakarta"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.query(_render: "json", cron: "0 1 * * 5")
    req.post() (err,res,body) ->
```

Gambar 4. 13 *Script* Penjadwalan *Pipeline*

b. Membuat proyek baru dan membagi ke grup

Pembuatan proyek baru sangat sering dilakukan oleh DevOps. Proses ini juga dapat diotomasi menggunakan bot, seperti yang dapat dilihat pada Gambar 4. 14 dan Gambar 4. 15. Pembuatan proyek ini dapat ditujukan ke pengguna lain ataupun pengguna itu sendiri. Cara, untuk dapat mengidentifikasi pengguna yang dituju adalah dengan, dilakukan proses pencarian id *user* berdasarkan masukan *username* yang diidentifikasi *regex*, dapat dilihat pada Gambar 4. 14. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File project_gitlab.coffee dan File group_gitlab.coffee).

```
module.exports = (robot) ->
  robot.respond /create projects (.*) for (.*)/i, (msg) ->
    project = msg.match[1]
    username = msg.match[2]
    path =
"#{url}/users?private_token=#{git_token}&search=#{username}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
```

Gambar 4. 14 *Script* permintaan pencarian id pengguna

Jika proses pencarian id pengguna berhasil, maka akan diambil data JSON dari tanggapan proses pada Gambar 4. 15 kemudian “user.id” akan diinisialisasi menjadi user_id, untuk dapat melengkapi alamat pada path yang akan dijalankan untuk membuat proyek baru. Setelah proses pembuatan proyek berhasil, juga akan diregenerate *trigger* pada proyek tersebut untuk dapat digunakan menjalankan *pipeline* jika terjadi kegagalan.

```
            user_id = user.id
            path =
"#{url}/projects/user/#{user_id}?private_token=#{git_token}"
            req = msg.http(path)
            req.query(_render: "json", name: project)
            req.header('Content-Length', 0)
            req.post() (err, res, body) ->
```

Gambar 4. 15 *Script* pembuatan proyek baru

Ketika proyek baru sudah dibuat, perlu dibagikan ke grup agar dapat saling berkolaborasi membangun kode yang dibutuhkan. Proses ini juga dapat diotomasi menggunakan bot, seperti yang dapat dilihat pada Gambar 4. 16. Pembagian proyek di Gitlab ini memerlukan variabel yang diidentifikasi sebagai alamat proyek dan alamat grup yang dituju berdasarkan masukan

perintah yang diseleksi oleh *regex*. Path akan mengirim *http request* menggunakan API Gitlab untuk dapat membagi proyek ke grup, dapat dilihat pada Gambar 4. 16.

```
robot.respond /share projects (.*) to groups (.*)/i, (msg) ->
    projectname = msg.match[1]
    groupname = msg.match[2]
    project = projectname.split('/').join('%2F');
    group = groupname.split('/').join('%2F');

    path =
"#{url}/groups/#{group}/projects/#{project}?private_token=#{git_to
ken}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.post() (err,res,body) ->
```

Gambar 4. 16 *Script* pembagian proyek ke grup Gitlab

c. Bergabung ke grup atau proyek

Pengguna dapat bergabung ke grup atau proyek tertentu tanpa harus mengakses Gitlab secara langsung. Hal ini sangat memudahkan dan mengefesiensi waktu, karena sudah diotomasikan oleh sistem ChatOps. Masukan akan diidentifikasi oleh *regex* ke beberapa variabel yang diperlukan, seperti *username* pengguna dan alamat proyek atau grup yang dituju. Permintaan *http request* akan diberikan ke Gitlab lewat API Gitlab untuk menseleksi id pengguna berdasarkan *username* yang dimasukan, seprti Gambar 4. 17. Kemudian data JSON akan diseleksi dan diambil data “id user” yang berhasil didapatkan. Kemudian proses permintaan penggabungan pengguna ke grup melalui *http request* dilakukan, lihat Gambar 4. 18. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File group_gitlab.coffee).

```
robot.respond /join (.*) to (groups|projects) (.*)/i, (msg) ->
    username = msg.match[1]
    pilih = msg.match[2]
    address = msg.match[3]
    alamat = address.split('/').join('%2F');
    path                                                              =
"#{url}/users?private_token=#{git_token}&username=#{username}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
```

Gambar 4. 17 *Script* seleksi id Pengguna

```
for user in JSON.parse body
            user_id = user.id
            path =
"#{url}/#{pilih}/#{alamat}/members?private_token=#{git_token}&user
_id=#{user_id}&access_level=30"
            req = msg.http(path)
            req.query(_render: "json")
            req.header('Accept','application/json')
            req.post() (err, res, body) ->
```

Gambar 4. 18 *Script* kode bergabung ke grup atau proyek

4.1.6 ChatOps untuk Manajemen pengetahuan

Penggunaan ChatOps menjadikan proses dokumentasi menjadi mudah, karena semua sudah terekam dalam *channel* ChatOps yang digunakan. Terdapat beberapa perintah yang digunakan untuk menampilkan informasi yang dibutuhkan bagi pengguna, yaitu melihat daftar proyek, pengguna, anggota dan isu yang dapat dilihat pada Gambar 4. 19 dan melakukan pencarian terhadap pengguna, proyek, grup dan isu sesuai kata kunci seperti pada Gambar 4. 20. Keseluruhan *script* dapat dilihat pada bagian Lampiran (File group_gitlab.coffee).

```
robot.respond /list (projects|users|members|issues) in
(projects|groups) (.*)/i, (msg) ->
    pilih1 = msg.match[1]
    pilih2 = msg.match[2]
    msgname = msg.match[3]
    name = msgname.split('/').join('%2F');

    path =
"#{url}/#{pilih2}/#{name}/#{pilih1}?private_token=#{git_token}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
```

Gambar 4. 19 *Script* kode melihat daftar di Gitlab

```
robot.respond /search (projects|groups|users|issues) by (.*)/i,
(msg) ->
    pilih = msg.match[1]
    key = msg.match[2]
    path =
"#{url}/#{pilih}?private_token=#{git_token}&search=#{key}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
```

Gambar 4. 20 *Script* kode pencarian di Gitlab

Gambar 4. 19 akan menyeleksi daftar yang dibutuhkan sesuai dengan masukan yang diidentifikasi *regex*, kemudian permintaan *http request* untuk melihat daftar dikirim ke Gitlab melalui API. Sedangkan pada Gambar 4. 20 masukan juga akan diseleksi oleh *regex* dan mengidentifikasi kata kunci yang diberikan. Proses pencarian akan diminta melalui path sesuai dengan variabel kata kunci sebelumnya.

4.2 Hasil Pengujian Sistem

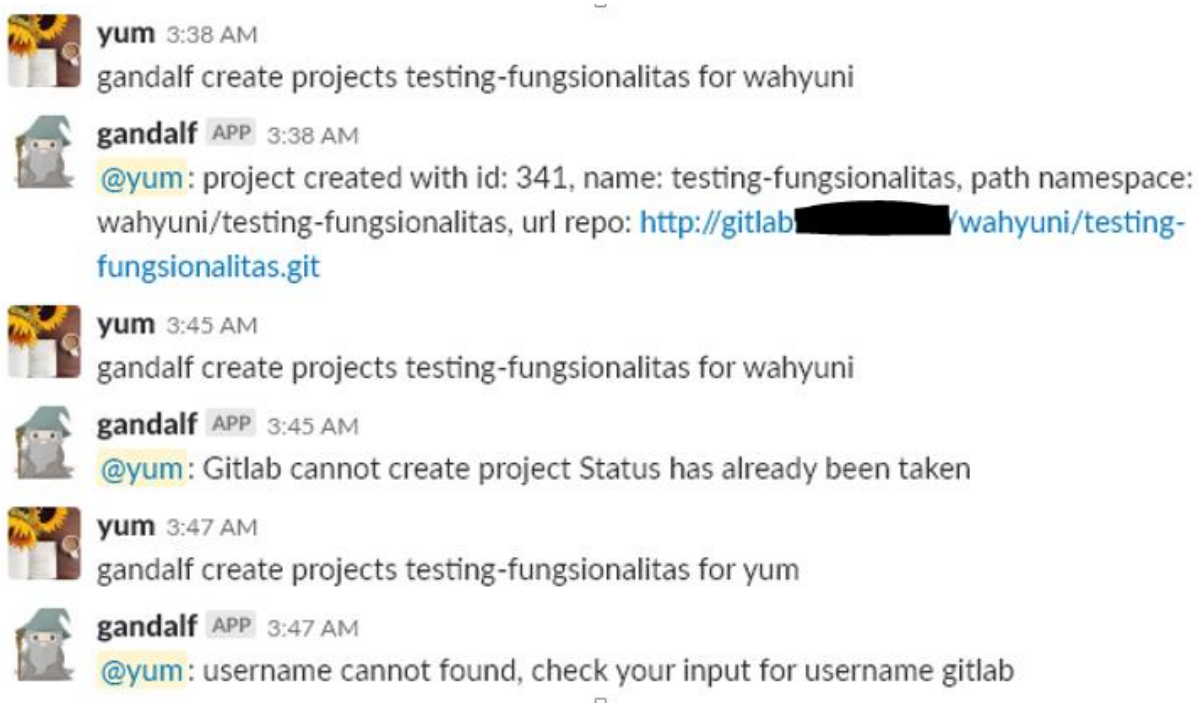
Sistem ChatOps menggunakan dua metode pengujian yaitu pengujian fungsionalitas dan pengujian usabilitas. Kedua pengujian tersebut memiliki tujuan tersendiri, pengujian fungsionalitas digunakan untuk memeriksa kebutuhan sistem yang dibuat sudah sesuai atau belum. Sedangkan, pengujian usabilitas digunakan mengukur tingkat efektif, efisien dan kepuasan pengguna dalam menggunakan fungsionalitas dari sistem. Bagian ini berisi ulasan terhadap hasil analisis pengujian fungsionalitas dan pengujian usabilitas.

4.2.1 Hasil Evaluasi Pengujian Fungsionalitas

Pengujian dilakukan di dalam *channel* "ChatOps" *workspace* Slack BSI UII, dengan memulai percakapan dengan Gandalf (bot). Skenario pengujian dibuat dan diterapkan untuk menganalisis sistem ChatOps sudah memenuhi kriteria yang dirancang. Pengujian ini dilakukan oleh penulis dengan serangkaian skenario yang sudah dirancang.

a. Membuat proyek baru

Percakapan dimulai dengan memanggil bot dan memberikan perintah untuk pembuatan proyek, seperti "Gandalf create projects testing-fungsionalitas for wahyuni" kemudian bot akan menanggapi permintaan sesuai dengan masukan yang diberikan. Perintah tersebut akan mengidentifikasi "testing-fungsionalitas" sebagai nama proyek baru yang akan dibuat, sedangkan "wahyuni" sebagai username yang dituju. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 1 ditunjukkan pada Gambar 4. 21. Pada Tabel 4. 1 tiga pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.



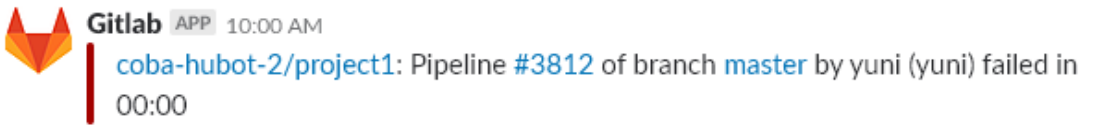
Gambar 4. 21 Hasil Pengujian Membuat Proyek Baru

Tabel 4. 1 Pengujian Terhadap Aksi Pembuatan Proyek

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Membuat proyek baru di Gitlab dengan benar	Perintah dengan nama proyek dan *username* (benar)	Proyek terbuat, menampilkan nama, alamat proyek dan *url repo*.	Informasi nama, alamat dan *url repo tampil* (valid)
Membuat proyek baru di Gitlab dengan benar, namun proyek sudah ada sebelumnya	Perintah dengan nama proyek dan username (salah)	Proyek tidak terbuat, menampilkan pesan *error*	Pesan *error* ditampilkan (valid)
Membuat proyek baru di Gitlab dengan username salah	Perintah dengan nama proyek dan username (salah)	Proyek tidak terbuat, menampilkan pesan *error*	Pesan *error* ditampilkan (valid)

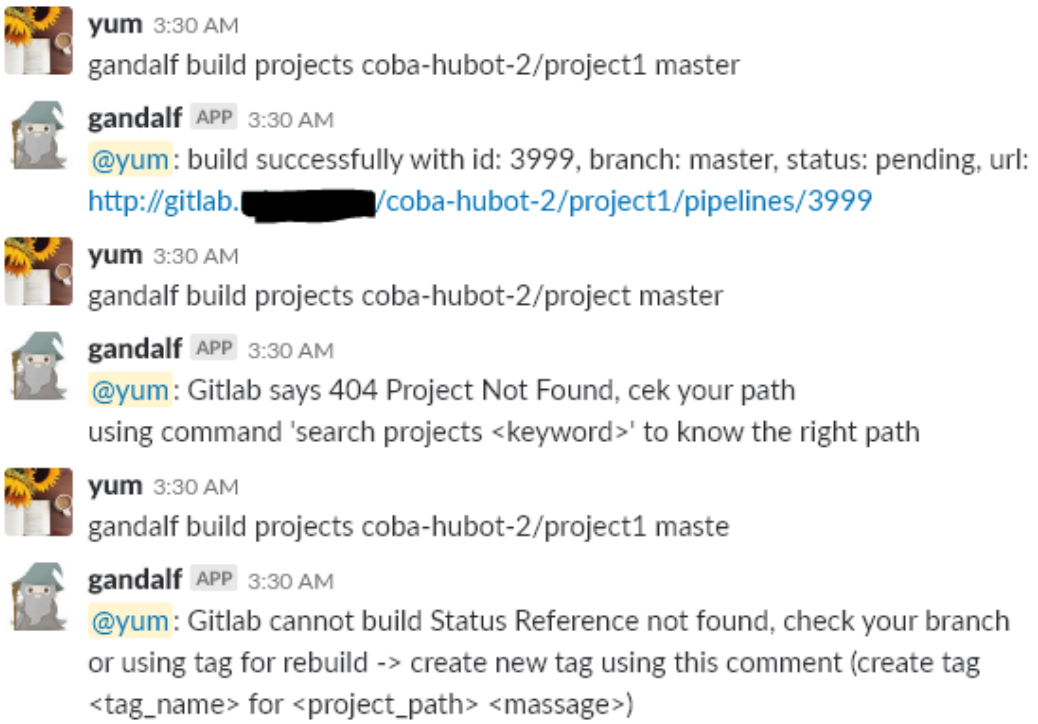
b. *Build* Proyek

Notifikasi *pipeline* yang gagal akan tampil seperti pada Gambar 4. 22, kemudian dapat diidentifikasi alamat proyek dan nama *branch* dari notifikasi, untuk dapat dijalankan ulang. Perintah *build* diberikan kepada bot dalam bentuk percakapan di dalam *channel*.



Gambar 4. 22 Notifikasi Proses *Pipeline* Gagal

Pemberian perintah dapat dilakukan, seperti "Gandalf *build* projects coba-hubot-2/project1 master" kemudian bot akan menanggapi permintaan sesuai dengan masukan yang diberikan. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 2 ditunjukkan pada Gambar 4. 23. Pada Pengujian Terhadap Aksi *Build Pipeline* Proyektiga pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.



Gambar 4. 23 Hasil Pengujian *Build Pipeline* Proyek

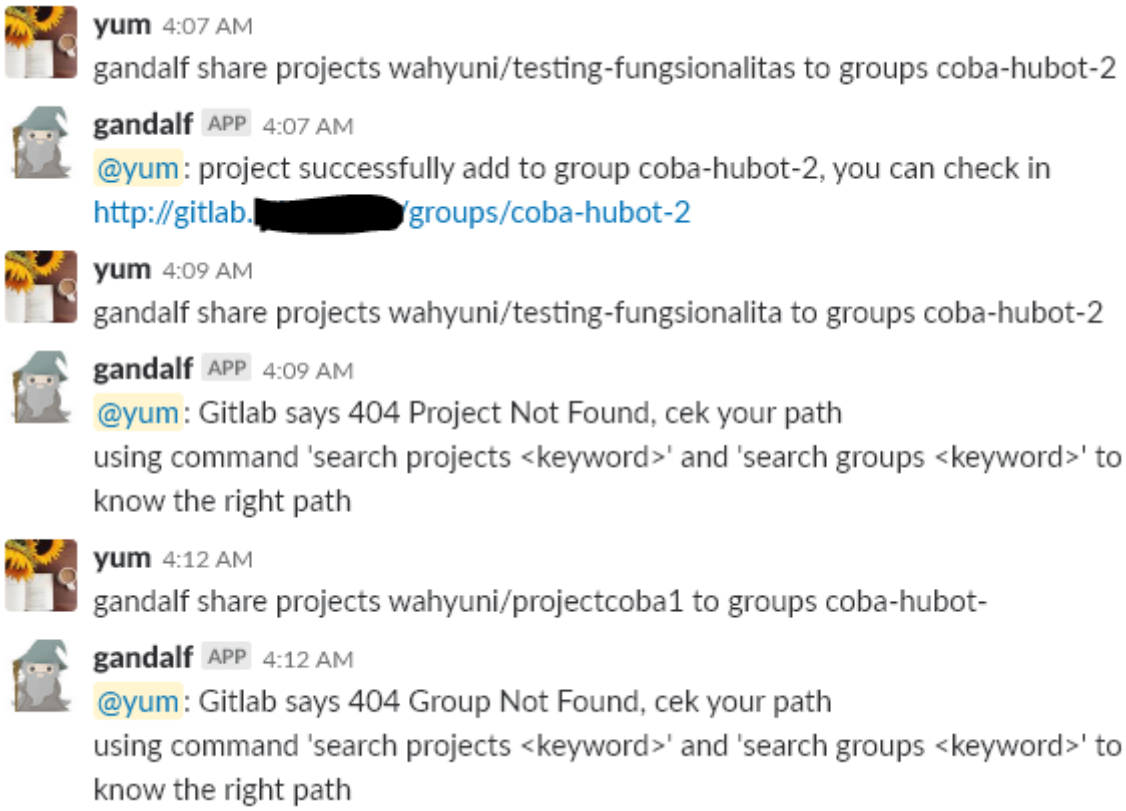
Tabel 4. 2 Pengujian Terhadap Aksi *Build Pipeline* Proyek

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Melakukan *Build* ulang ke sebuah proyek tertentu.	Perintah dengan alamat proyek dan nama branch (benar)	*Build* berhasil, menginformasikan id, nama branch, status dan url.	Informasi nama, alamat dan url repo tampil (valid)

Melakukan *Build* ulang ke sebuah proyek tertentu, dengan alamat proyek yang salah.	Perintah dengan alamat proyek yang salah dan nama branch benar (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Melakukan *Build* ulang ke sebuah proyek tertentu, dengan nama branch yang salah.	Perintah dengan alamat proyek yang benar dan nama branch salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)

c. Membagi Proyek Gitlab ke Grup

Pengguna akun Slack memberikan perintah untuk membagian proyek di grup, seperti "Gandalf share projects wahyuni/testing-fungsionalitas to groups coba-hubot-2" bot akan menanggapi permintaan sesuai dengan masukan yang diberikan. Perintah tersebut akan mengidentifikasi "wahyuni/testing-fungsionalitas" sebagai alamat proyek yang akan dibagi, sedangkan "coba-hubot-2" sebagai alamat grup yang dituju. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 3 ditunjukkan pada Gambar 4. 24. Pada Tabel 4. 3 tiga pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.



Gambar 4. 24 Hasil Pengujian Pembagian Proyek ke Grup Gitlab

Tabel 4. 3 Pengujian Terhadap Aksi Pembagian Proyek ke Grup

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Melakukan Pembagian suatu proyek ke sebuah grup di Gitlab dengan benar	Perintah dengan alamat proyek dan alamat grup (benar)	Pembagian berhasil, menginformasikan url alamat proyek yang baru	Informasi url tampil (valid)
Melakukan Pembagian suatu proyek ke sebuah grup di Gitlab dengan alamat proyek yang salah	Perintah dengan alamat proyek salah dan alamat grup benar (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Melakukan Pembagian suatu proyek ke sebuah grup di Gitlab dengan alamat grup yang salah	Perintah dengan alamat proyek benar dan alamat grup salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)

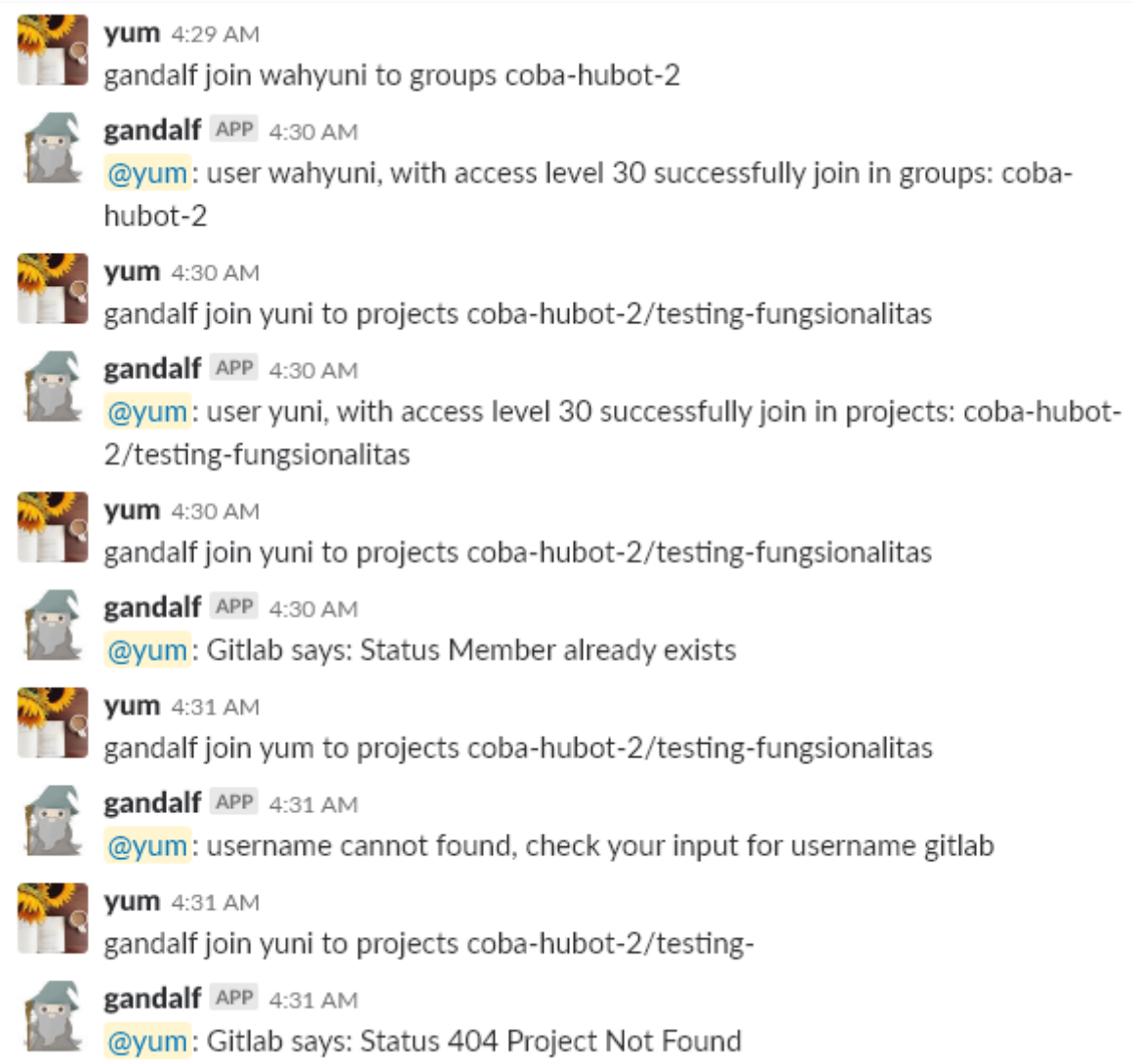
d. Bergabung ke Grup atau Proyek tertentu

Bergabung ke grup atau proyek yang diinginkan, sekarang dapat dilakukan melalui percakapan dengan bot, seperti “Gandalf join zaidatin to groups coba-hubot” untuk bergabung dengan suatu grup, “zaidatin” akan diidentifikasi sebagai *username* dan “coba-hubot” adalah alamat grup yang dituju. Penggunaan perintah untuk dapat bergabung dengan suatu proyek, yaitu “Gandalf join zaidatin to projects coba-hubot-2/testing-fungsionalitas. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 4 ditunjukkan pada Gambar 4. 25. Pada Tabel 4. 4 empat pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.

Tabel 4. 4 Pengujian Terhadap Aksi Bergabung ke Grup atau Proyek Tertentu

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Bergabung ke suatu grup atau proyek tertentu dengan benar	Perintah dengan username dan alamat grup atau proyek (benar)	Berhasil bergabung dengan grup, menginformasikan nama user dan access level.	Informasi nama dan access level tampil (valid)
Bergabung ke suatu proyek atau grup tertentu dengan benar, namun beratabrakan karena sudah bergabung	Perintah dengan username dan alamat grup atau proyek (benar)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Bergabung ke suatu proyek atau grup tertentu, dengan username yang salah	Perintah dengan alamat proyek atau grup yang benar dan username yang salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Bergabung ke suatu proyek atau grup tertentu, dengan	Perintah dengan alamat proyek atau grup yang	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)

alamat grup atau proyek yang salah	salah dan username yang benar (salah)		

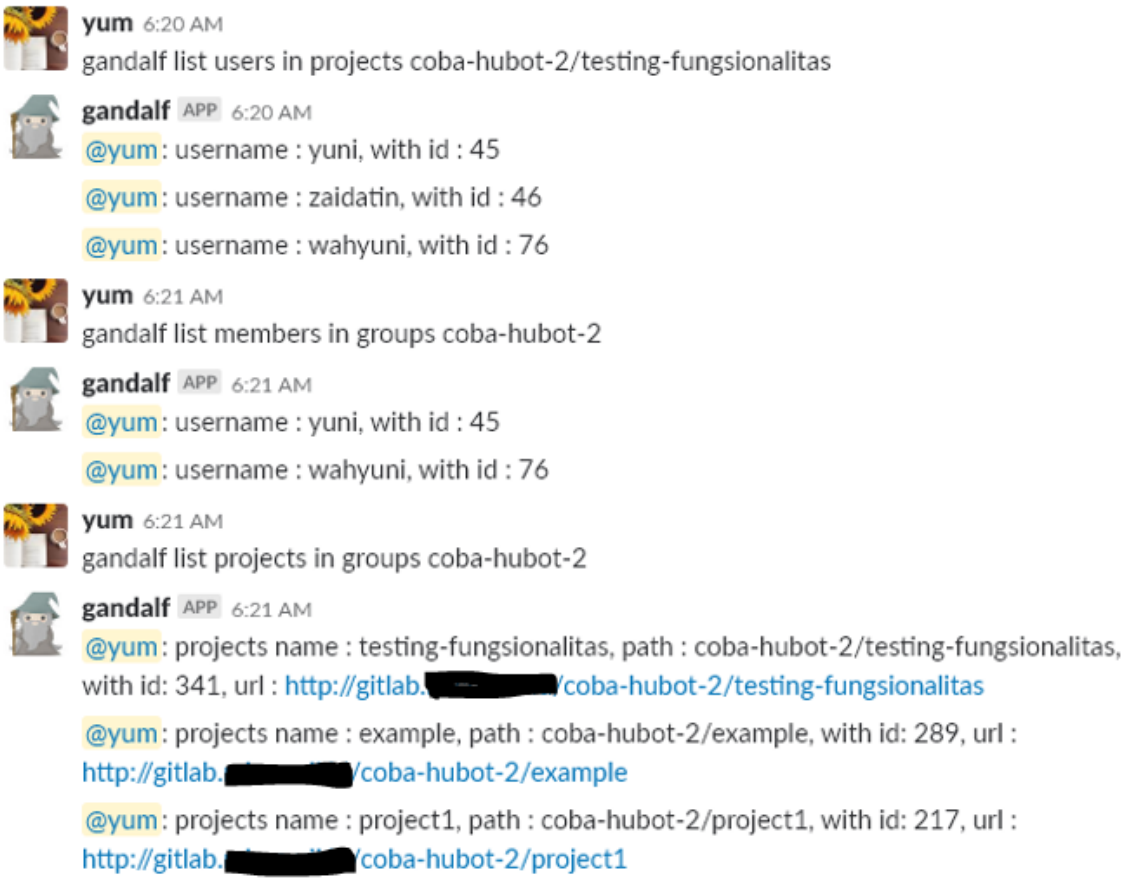


Gambar 4. 25 Hasil Pengujian Bergabung ke Grup atau Proyek Tertentu

e. Melihat Daftar Pengguna, Anggota dan Proyek

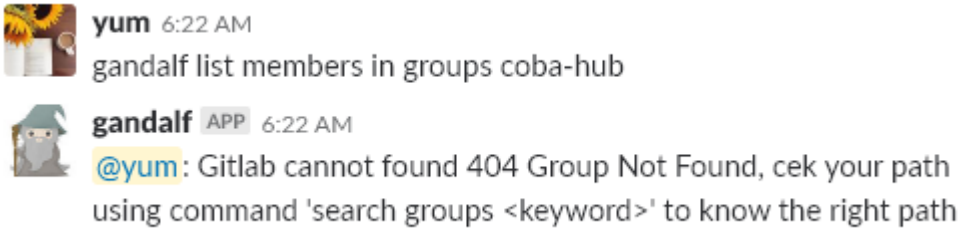
Mendapatkan informasi dari Gitlab juga diotomasikan dalam sistem ChatOps, seperti untuk melihat daftar pengguna dari suatu proyek, menggunakan “Gandalf list users in projects coba-hubot-2/testing-fungsionalitas”. Melihat anggota dari suatu grup, dapat memakai “Gandalf list members in groups coba-hubot-2”.Kemudian untuk melihat daftar proyek dari suatu grup dapat menggunakan “Gandalf list projects in groups coba-hubot-2”. Contoh

penggunaan dan tanggapan dapat dilihat di Gambar 4. 26 yang sesuai skenario pada Tabel 4. 5.



Gambar 4. 26 Hasil Pengujian Melihat Daftar Pengguna, Anggota dan Proyek

Pada Gambar 4. 26 menunjukkan tanggapan dari data yang diberikan oleh Gitlab API, tambahan tanggapan *error* pada Gambar 4. 27 dari bot melengkapi skenario yang dirancang dalam Tabel 4. 5. Pada Tabel 4. 5 juga menyatakan empat pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.



Gambar 4. 27 Hasil Pengujian *Error* saat Melihat Daftar Pengguna, Anggota dan Proyek

Tabel 4. 5 Pengujian Terhadap Aksi Melihat Daftar Pengguna, Anggota dan Proyek

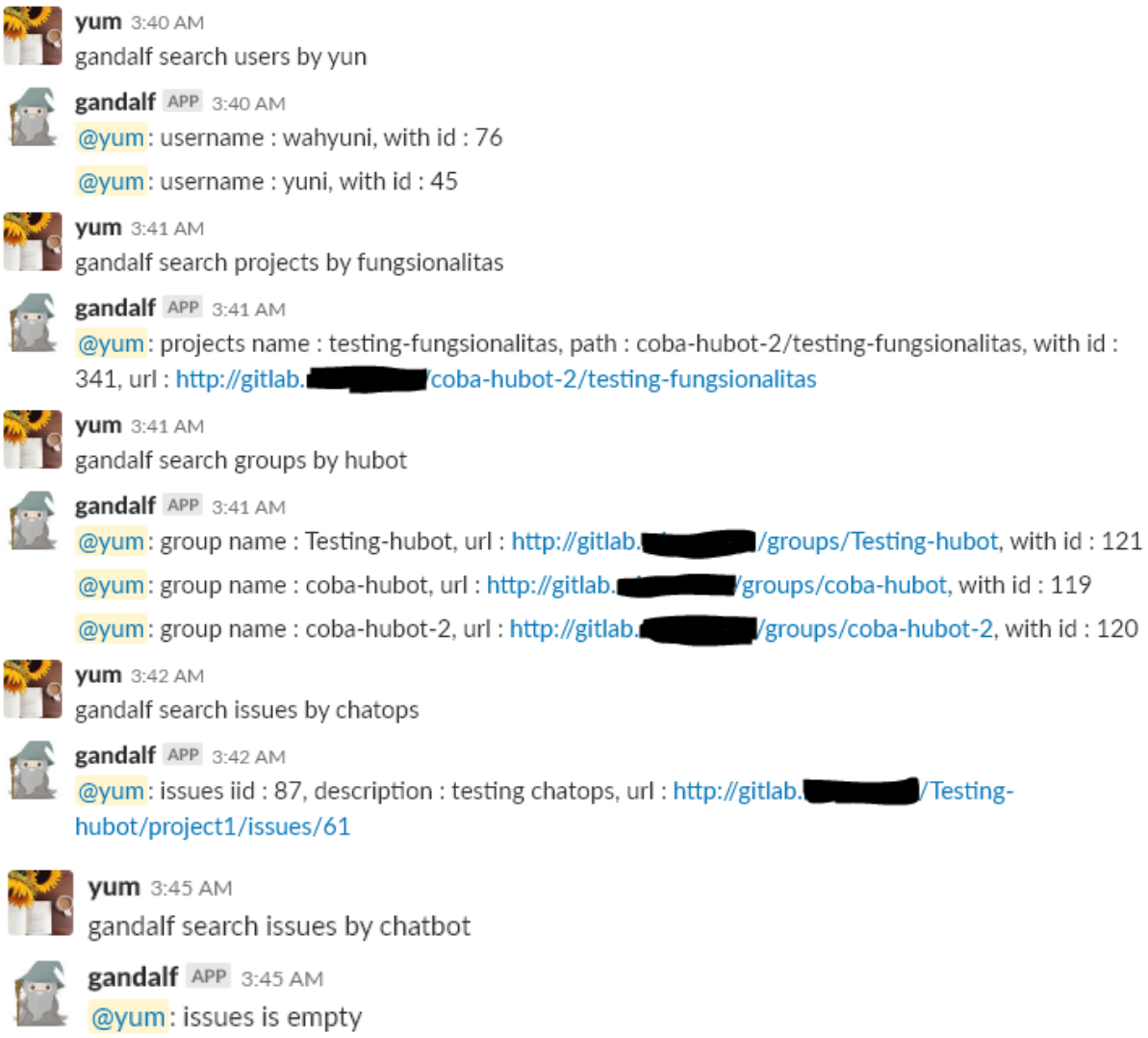
Skenario pengujian	Input	Output yang diharapkan	Output hasil
Melihat daftar pengguna dari suatu proyek dengan benar	Perintah dengan alamat proyek (benar)	Berhasil menampilkan daftar user pengguna dari proyek.	Daftar pengguna tampil (valid)
Melihat daftar anggota dari suatu grup dengan benar	Perintah dengan alamat grup (benar)	Berhasil menampilkan daftar user anggota dari grup.	Daftar anggota tampil (valid)
Melihat daftar proyek dari suatu grup dengan benar	Perintah dengan alamat grup (benar)	Berhasil menampilkan daftar proyek dari grup.	Daftar proyek tampil (valid)
Melihat daftar pengguna, anggota atau proyek jika alamat salah	Perintah dengan alamat proyek atau grup yang salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)

f. Mencari Pengguna, Proyek, Grup dan Isu

Ketika melakukan berbagai perintah yang terdapat pada sistem ChatOps, sering dibutuhkan variabel-variabel tertentu yang harus tepat, seperti *username*, alamat proyek, alamat grup dan id isu. Kesalahan masukan bisa terjadi, untuk itu pencarian sangat diperlukan. Melakukan pencarian terhadap variabel tertentu di Gitlab melalui bot dapat dilakukan, dengan "Gandalf search users by yun" untuk mencari pengguna, "gandalf search projects by fungsionalitas" untuk mencari proyek, "gandalf search groups by hubot" untuk mencari grup, sedangkan "gandalf search issues by hubot" untuk mencari isu. Kata kunci akan dideteksi di setiap perintah memulai pencarian yang sesuai. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 6 ditunjukkan pada Gambar 4. 28. Pada Tabel 4. 6 dua pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.

Tabel 4. 6 Pengujian Terhadap Aksi Pencarian Pengguna, Proyek, Grup dan Isu di Gitlab

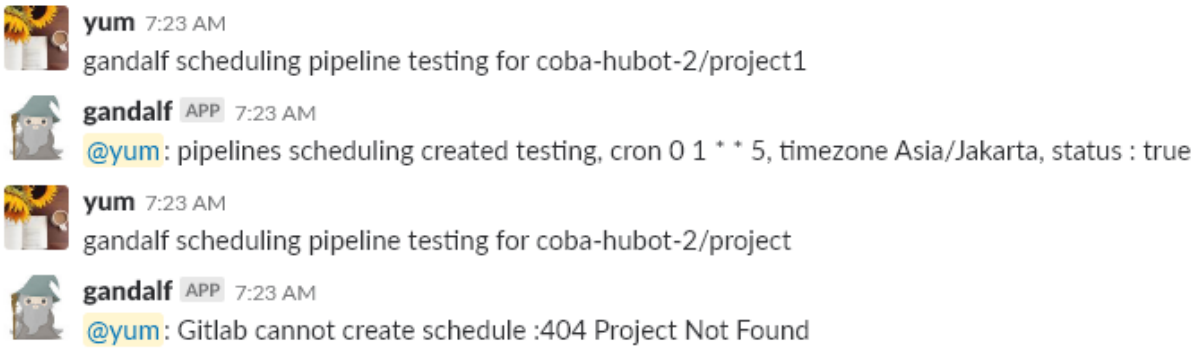
Skenario pengujian	Input	Output yang diharapkan	Output hasil
Mencari pengguna, proyek, grup atau isu dengan kata kunci dengan benar	Perintah dengan kata kunci untuk pencarian (benar)	Berhasil menampilkan daftar sesuai kata kunci yang dicari	Daftar pengguna tampil (valid)
Mencari pengguna, proyek, grup atau isu dengan kata kunci dengan benar, namun hasil yang dicari tidak ada	Perintah dengan kata kunci untuk pencarian (benar)	Berhasil menampilkan pesan peringatan jika yang dicari tidak ada	Pesan peringatan tampil (valid)



Gambar 4. 28 Hasil Pengujian Mencari Pengguna, Proyek, Grup dan Isu

g. Penjadwalan *Pipeline*

Penjadwalan *pipeline* diperlukan untuk otomasi *build* proyek pada waktu-waktu tertentu. Hal ini juga dapat dilakukan secara otomatis di sistem ChatOps dengan membuat penjadwalan melalui bot, sepeti "Gandalf scheduling *pipeline* testing for coba-hubot-2/project1". Variabel "coba-hubot-2/project1" akan diidentifikasi sebagai alamat proyek yang dilakukan penjadwalan *pipeline*. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 7 ditunjukkan pada Gambar 4. 29. Pada Tabel 4. 7 dua pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.



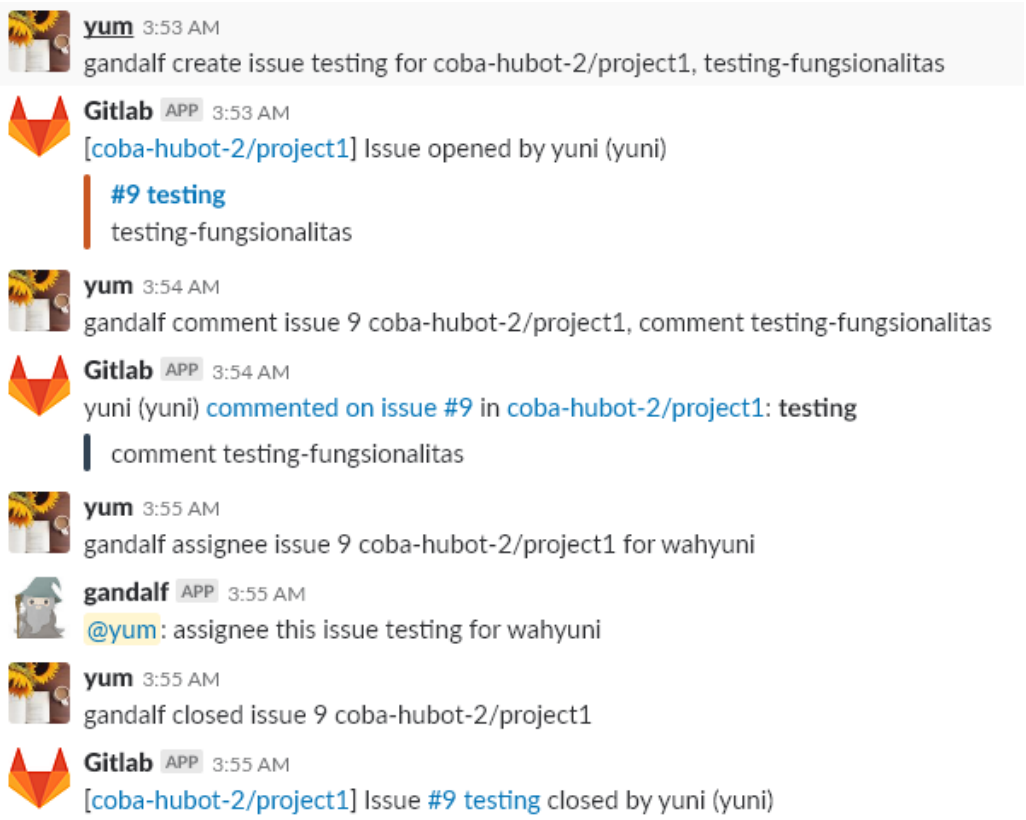
Gambar 4. 29 Hasil Pengujian Penjadwalan *Pipeline* Proyek

Tabel 4. 7 Pengujian Terhadap Aksi Penjadwalan *Pipeline* Proyek

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Melakukan penjadwalan *build* untuk proyek tertentu dengan benar	Perintah dengan nama penjadwalan dan alamat proyek (benar)	Penjadwalan berhasil dibuat, menginformasikan deskripsi, cron, timezone yang telah aktif.	Informasi penjadwalan baru yang telah aktif tampil (valid)
Melakukan penjadwalan *build* untuk proyek tertentu dengan alamat proyek salah	Perintah dengan nama penjadwalan dan alamat proyek yang salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)

h. Pengelolaan Isu

Pengelolaan isu sangat dibutuhkan untuk menjadi dokumentasi dari setiap masalah dan penganalisisan dilakukan secara bersama untuk menghilangkan batas antar tim. Isu juga dapat digunakan sebagai rujukan bila terjadi masalah yang sama. Sistem ChatOps menambahkan pengelolaan isu terkait pembuatan isu, dengan "Gandalf create isu testing for coba-hubot-2/project1, testing-fungsionalitas" dan mengomentari isu dengan "Gandalf comment isu 8 for coba-hubot-2/project1, comment testing-fungsionalitas". Perintah tersebut akan mengidentifikasi "coba-hubot-2/project1" sebagai alamat proyek yang akan dibuat isu, sedangkan "8" merupakan id dari isu yang akan dikomentari. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 8 ditunjukkan pada Gambar 4. 30 dan Gambar 4. 31. Pada Tabel 4. 8 sembilan pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.

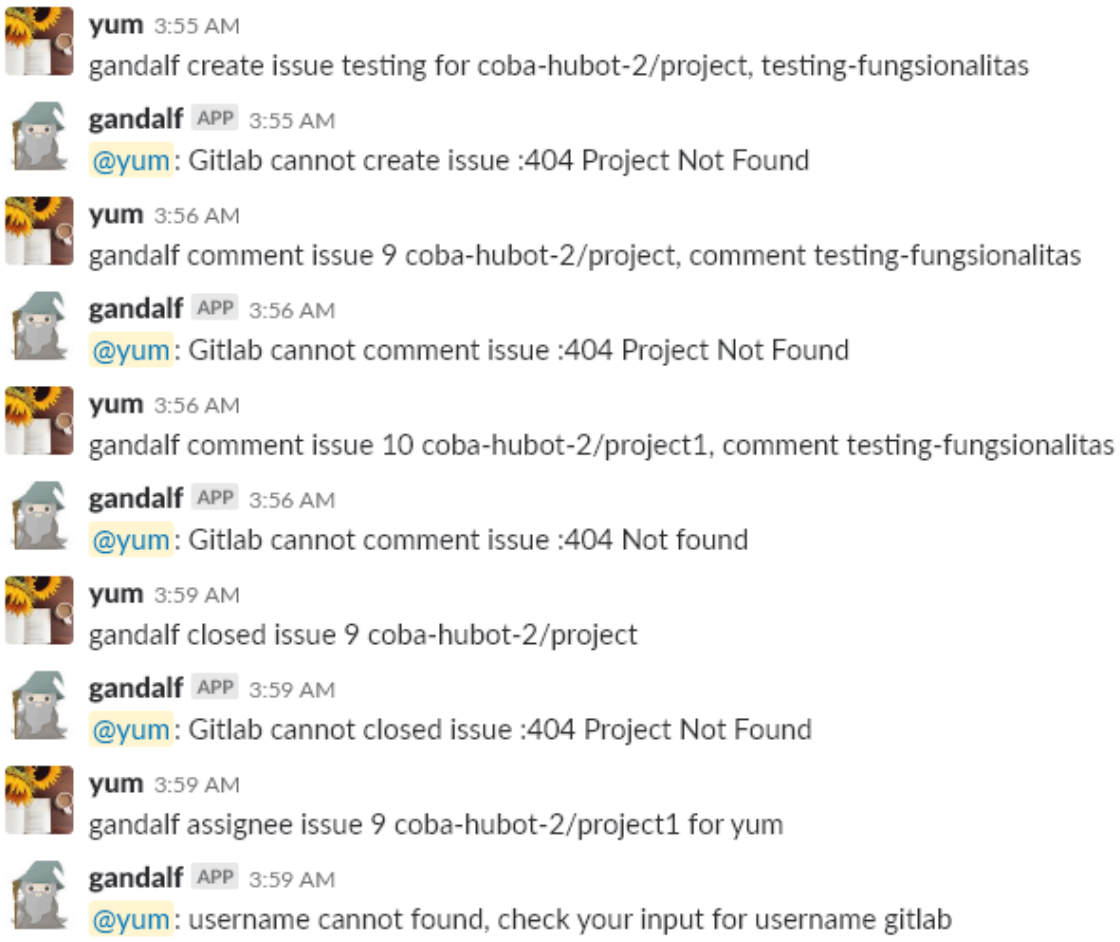


Gambar 4. 30 Hasil Pengujian Pengelolaan Isu Berhasil

Tabel 4. 8 Pengujian Terhadap Aksi Pengelolaan Isu

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Membuat isu untuk suatu proyek dengan benar	Perintah dengan nama isu, alamat proyek dan deskripsi isu (benar)	Notifikasi pembuatan isu baru dari Gitlab akan tampil	Notifikasi tampil (valid)
Membuat isu untuk suatu proyek dengan alamat proyek yang salah	Perintah dengan nama isu, alamat proyek yang salah dan deskripsi isu (salahr)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Mengomentari isu untuk suatu proyek dengan benar	Perintah dengan id isu, alamat proyek dan deskripsi komentar (benar)	Notifikasi komentar dari Gitlab akan tampil	Notifikasi tampil (valid)
Mengomentari isu untuk suatu proyek dengan id isu yang salah	Perintah dengan id isu yang salah, alamat proyek dan deskripsi komentar benar (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Mengomentari isu untuk suatu proyek	Perintah dengan id isu benar,	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)

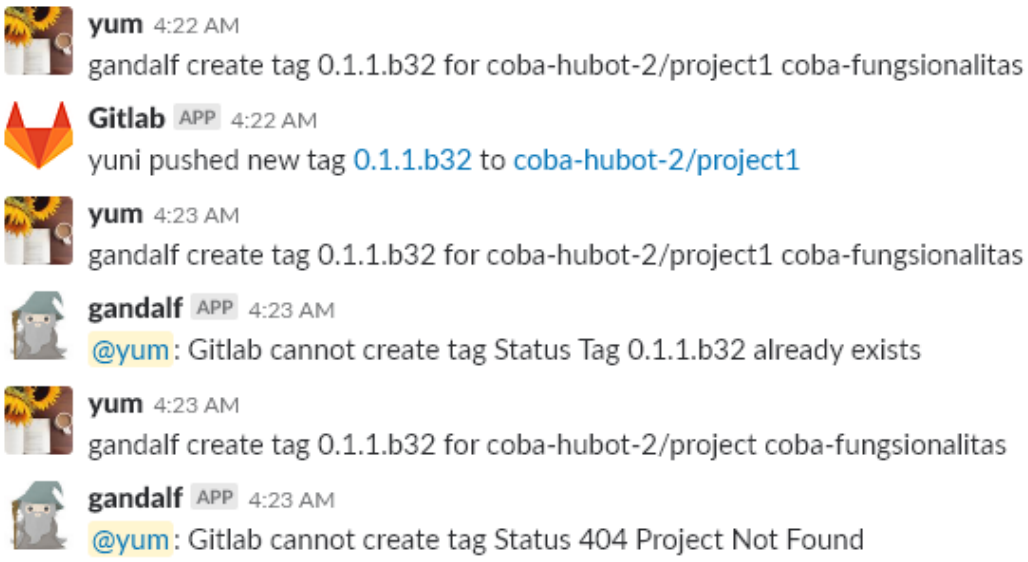
dengan alamat proyek yang salah	alamat proyek salah dan deskripsi komentar (salah)		
Menutup isu untuk suatu proyek dengan benar	Perintah dengan id isu, alamat proyek (benar)	Notifikasi penutupan isu dari Gitlab akan tampil	Notifikasi tampil (valid)
Menutup isu untuk suatu proyek dengan masukan yang salah	Perintah dengan id isu, alamat proyek salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)
Memberikan tugas isu ke pengguna dengan benar	Perintah dengan id isu, alamat proyek dan username (benar)	Notifikasi pemberian tugas ke pengguna dari Gitlab akan tampil	Pesan pemberitahuan pemberian tugas ditampilkan (valid)
Memberikan tugas isu ke pengguna dengan masukan yang salah	Perintah dengan id isu, alamat proyek, dan username salah (salah)	Peringatan serta pesan *error*	Pesan *error* ditampilkan (valid)



Gambar 4. 31 Hasil Pengujian Pengelolaan Isu Gagal

i. Membuat tag baru

Percakapan dimulai dengan memanggil bot dan memberikan perintah untuk pembuatan tag proyek, seperti “Gandalf create tag 0.1.1.b32 for coba-hubot-2/project1 coba-fungsionalitas” kemudian bot akan menanggapi permintaan sesuai dengan masukan yang diberikan. Perintah tersebut akan mengidentifikasi “0.1.1.b32” sebagai nama tag baru yang akan dibuat, sedangkan “coba-hubot-2/project1” sebagai alamat proyek yang dituju, kemudian “coba-fungsionalitas” akan diidentifikasi sebagai pesan. Tanggapan akan diberikan oleh bot sesuai dengan data yang diberikan oleh Gitlab API, tanggapan lengkap dari bot sesuai dengan skenario yang dirancang dalam Tabel 4. 9 ditunjukkan pada Gambar 4. 32. Pada Tabel 4. 9 tiga pengujian yang dilakukan dikatakan valid sehingga *use case* ini berhasil.



Gambar 4. 32 Hasil Pengujian Membuat Tag Proyek Baru

Tabel 4. 9 Pengujian Terhadap Aksi Pembuatan Tag Proyek

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Membuat tag proyek baru di Gitlab dengan benar	Perintah dengan nama tag proyek, alamat proyek dan pesan (benar)	Tag terbuat, notifikasi informasi pembuatan akan tampil.	Notifikasi pembuatan tag ditampilkan (valid)
Membuat tag proyek baru di Gitlab dengan benar, namun tag proyek sudah ada sebelumnya	Perintah dengan nama tag proyek, alamat proyek dan pesan (salah)	Tag tidak terbuat, menampilkan pesan *error*	Pesan *error* ditampilkan (valid)
Membuat tag proyek baru di Gitlab dengan alamat proyek salah	Perintah dengan nama tag proyek, alamat proyek	Tag tidak terbuat, menampilkan pesan *error*	Pesan *error* ditampilkan (valid)

	salah dan pesan (salah)		

4.2.2 Hasil Pengujian Usabilitas

Pengujian dilakukan dengan melakukan *task-task* berdasarkan alur pada Gambar 3. 5 untuk melihat bagaimana perbedaan dengan menggunakan sistem lama (yang ditunjukkan pada alur Gambar 3. 2). Pada pengujian ini akan dinilai aspek-aspek *learnability*, *efficiency*, *memorability*, *error*s dan *satisfaction* untuk menunjukkan tingkat usabilitas menurut responden, selain itu akan diukur bagaimana dampak penggunaan sistem baru berdasarkan pertanyaan yang diajukan di kuisoner. Responden merupakan Tim 4 yang berjumlah 5 orang bagian development di BSI yang merupakan pengguna dari Gitlab.

Berikut ini plot kelima aspek usabilitas terhadap 10 pertanyaan kuisoner yang dapat dilihat pada Tabel 4. 10.

Tabel 4. 10 Plot Aspek usabilitas terhadap pertanyaan

No	Pertanyaan
Learnability	
1.	Apakah sistem baru lebih efektif?
2.	Apakah sistem mudah digunakan?
3.	Dapat belajar dengan cepat ketika menggunakannya?
Efficiency	
4.	Apakah sistem baru membantu lebih produktif?
5.	Apakah sistem baru menghemat waktu penyelesaian kasus?
6.	Apakah sistem meningkatkan kolaborasi tim?
Memorability	
7.	Apakah mudah mengingat cara menggunakannya?
8.	Apakah bisa menggunakannya tanpa instruksi tertulis?
Error	
9.	Apakah notifikasi *error* membantu memperbaiki kesalahan?
Satisfaction	
10.	Apakah puas dengan sistem ini?

Uji Validitas dan Reliabilitas

Setelah kuisoner dibuat menggunakan penilaian skala likert, sebelum data digunakan diperlukan pengujian terhadap tongkat validitas dan realibilitasnya. Pengujian validitas *Product Moment Pearson Correlation*, peneliti menggunakan Microsoft office excel berdasarkan data dari 5 responden. Berdasarkan hasil keluaran perhitungan validitas, dapat diambil kesimpulan dengan perbandingan rhitung dan rtabel. Nilai rtabel, dengan jumlah N=5

pada signifikan 10% adalah 0,8054. Tabel perbandingan rtabel dan rhitung dapat dilihat pada Tabel 4. 11. Berdasarkan Tabel 4. 11, maka kuisoner yang digunakan valid seluruhnya.

Tabel 4. 11 Perbandingan Rtabel dan Rhitung

Q	Pearson Correlation (rhitung)	Rtabel (signifikansi 10%)	Keterangan
1	0,860268076	0,8054	Valid
2	0,927411313	0,8054	Valid
3	0.860268076	0,8054	Valid
4	0.927411313	0,8054	Valid
5	0.877862291	0,8054	Valid
6	0.951017482	0,8054	Valid
7	0,846079885	0,8054	Valid
8	0,860268076	0,8054	Valid
9	0,874371771	0,8054	Valid
10	0.860268076	0,8054	Valid

Setelah dilakukan pengujian validitas selanjutnya dilakukan uji reliabilitas. *Cronbach's alpha* yang digunakan dalam penelitian ini adalah 0,8054. Setiap item dimasukan ke uji relibilitas semua item yang benar. Berdasarkan keluaran uji relibilias nilai Alpha sebesar 0,958205912. Hal ini menunjukkan nilai Alpha lebih besar dari rtabel, maka item-item kuisoner dinyatakan reliable atau konsisten dan terpercaya sebagai alat pengumpul data.

Analisis Usabilitas testing

Setelah kuisoner berhasil dikumpulkan, maka dilakukan perekapan terhadap hasil kuisoner. Persentase jawaban kuisoner yang dilakukan terhadap 5 responden dapat dilihat pada Tabel 4. 12.

Tabel 4. 12 Hasil Rekapitulasi Kuisoner

Pertanyaan Kuisoner	Sangat Tidak Setuju	Tidak Setuju	Netral	Setuju	Sangat Setuju
1	0	0	0	4	1
2	0	0	1	3	1
3	0	0	0	4	1
4	0	0	1	3	1
5	0	0	1	2	2
6	0	0	2	2	1
7	0	0	3	2	0
8	0	0	4	1	0
9	0	0	2	1	2
10	0	0	0	4	1

Berdasarkan hasil rekapitulasi kuisoner pada Tabel 4. 12, maka diperoleh rekap hasil usabilitas yang dapat dilihat di Tabel 4. 13.Nilai rekap berasal dari perhitungan total nilai jawaban dibagi 5 responden dari masing-masing pertanyaan.

Tabel 4. 13 Nilai Rekap setiap Pertanyaan

No	Pertanyaan	Nilai	Persentase
1.	Apakah sistem baru lebih efektif?	4.2	84%
2.	Apakah sistem mudah digunakan?	4	80%
3.	Dapat belajar dengan cepat ketika menggunakannya?	4.2	84%
4.	Apakah sistem baru membantu lebih produktif?	4	80%
5.	Apakah sistem baru menghemat waktu penyelesaian kasus?	4.2	84%
6.	Apakah sistem meningkatkan kolaborasi tim?	3.8	76%
7.	Apakah mudah mengingat cara menggunakannya?	3.4	68%
8.	Apakah bisa menggunakannya tanpa instruksi tertulis?	3.2	64%
9	Apakah notifikasi *error* membantu memperbaiki kesalahan?	4	80%
10.	Apakah puas dengan sistem ini?	4.2	84%

Tabel 4. 12 menunjukkan nilai kepuasan penerimaan user terhadap item pertanyaan. Apabila disesuaikan dengan Tabel 4. 13, dapat dikatakan bahwa sistem yang diterapkan memiliki nilai usabilitas yang sangat baik.

Berdasarkan hasil pengukuran aspek usabilitas didapatkan bahwa sistem memiliki nilai usabilitas yang baik. Nilai usabilitas dari masing-masing aspek dapat dilihat di Tabel 4. 14 yang didapat dari hasil perbandingan Tabel 4. 13 yang merupakan nilai rekap dari setiap pertanyaan yang diajukan, denganTabel 4. 10 yang merupak plot dari setiap aspek usabilitas di setiap pertanyaan.

Tabel 4. 14 Hasil pengujian usabilitas

No	Aspek Usabilitas	Nilai
1	*Learnability*	4.1
2	*Efficiency*	4
3	*Memorability*	3.3
4	*Errors*	4
5	*Satisfaction*	4.2

Dapat dilihat pada Tabel 4. 14, bahwa sistem ChatOps mudah untuk dipelajari dilihat dari aspek *learnability*, sistem ChatOps juga efisien dilihat dari aspek *efficiency*, memiliki tingkat kesalahan yang rendah dan memuaskan responden namun dalam aspek *memorability* memiliki nilai yang paling rendah yang menunjukkan sistem butuh waktu untuk dapat diingat atau dipelajari secara lancar. Pada Tabel 4. 12, menunjukkan nilai rekap responden terhadap masing-masing pertanyaan, dan dalam pertanyaan tersebut terdapat pertanyaan-pertanyaan

khusus yang ditujukan untuk mengetahui dampak penggunaan sistem ChatOps dibandingkan sistem yang lama. Selain itu dilakukan wawancara kepada responden terkait dengan penilaian mereka terhadap pertanyaan ini, hasilnya dirangkum sebagai berikut:

Sistem ChatOps dinyatakan lebih efektif dengan skor 4.2 atau 84% responden menyetujui. Semua responden membenarkan jika sistem meningkatkan keefektifan kerja dengan beragam alasan. Responden pertama dan kelima berpendapat bahwa sistem sangat membantu karena rekap pekerjaan terlihat dan siapa anggota tim yang melakukan pekerjaan tertentu di Gitlab terekam, sehingga mudah diidentifikasi jika terdapat masalah. Responden kedua, ketiga dan Keempat berpendapat bahwa sistem menggantikan komunikasi langsung ke media *chat*, sehingga jika terdapat masalah tidak harus menemui tim *operations* secara langsung, karena tim *operations* tidak hanya bekerja di satu tempat saja.

Sistem ChatOps membantu lebih produktif dengan skor 4 atau 80% responden menyetujui. Semua responden membenarkan jika sistem membantu lebih produktif dengan berbagai alasan. Responden pertama berpendapat sistem memperpendek waktu dengan *real-time* notifikasi *pipeline*. Responden kedua berpendapat sistem membantu produktif dalam pekerjaan *developer* karena tidak perlu meninggalkan pekerjaan untuk bertemu dengan tim lain sehingga konsentrasi tetap fokus. Responden ketiga berpendapat bahwa setiap anggota dapat melihat pekerjaan anggota lain sehingga dapat mengerjakan pekerjaan lain yang belum ditangani. Responden keempat dan kelima berpendapat dengan sistem yang baru setiap masalah yang dihadapi lebih cepat ditanggapi oleh banyak orang.

Sistem ChatOps menghemat waktu penyelesaian kasus dengan skor 4.2 atau 84% responden menyetujui. Semua responden membenarkan jika sistem membantu menghemat waktu penyelesaian kasus dengan berbagai alasan. Responden pertama berpendapat sistem menghemat waktu karena cukup melakukan setiap kegiatan dengan mengetikkan perintah di Slack. Responden kedua, ketiga dan keempat berpendapat sistem membantu menghemat waktu karena tidak perlu bertatap muka secara langsung ke tim lain untuk menyelesaikan masalah sehingga dapat fokus pada pekerjaan masing-masing. Sedangkan responden kelima berpendapat sistem menghemat waktu penyelesaian tergantung pada kasus yang dikerjakan, namun secara umum sistem ini lebih menghemat waktu dari pada sistem yang lama.

Sistem ChatOps meningkatkan kolaborasi tim dengan skor 3.8 atau 76% responden menyetujui. Semua responden membenarkan jika sistem membantu meningkatkan kolaborasi tim dengan berbagai alasan. Responden pertama berpendapat sistem membantu komunikasi dengan tim *operations* sehingga lebih intensif dan semua tim tahu kegiatan tim lainnya.

Responden kedua, ketiga dan keempat berpendapat jika sistem membantu anggota lain untuk peduli dengan masalah yang dihadapi oleh yang lainnya, sehingga dapat saling membantu menyelesaikan masalah. Responden kelima berpendapat bahwa peningkatan kolaborasi tim bergantung pada personal anggota masing-masing, namun sistem ini memfasilitasi untuk dapat berkerjasama lebih baik.

Secara umum dapat disimpulkan sistem dinyatakan lebih efektif, lebih produktif, membantu penyelesaian terhadap kasus dan meningkatkan kolaborasi tim. Semua responden juga berpendapat bahwa sistem membantu kinerja menjadi lebih baik, sehingga perlu diterapkan secara menyeluruh dan dikembangkan kebagian lain.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan dapat ditarik kesimpulan bahwa sistem ChatOps telah berhasil dibuat untuk Badan Sistem Informasi Universitas Islam Indonesia, sesuai kebutuhan tim DevOps. Sistem ChatOps digunakan untuk mengelola Gitlab dalam perilisan aplikasi ke *production*. Dari lima variabel yang digunakan untuk menganalisis usabilitas sistem ChatOps, diketahui bahwa sistem belum memenuhi variabel memorability, namun sudah memenuhi empat variabel lainnya yaitu learnability, efficiency, errors dan satisfaction. Oleh karena itu dapat disimpulkan bahwa Sistem ChatOPs secara umum sudah memenuhi faktor usabilitas atau usable.

Dampak implementasi sistem otomasi ChatOps membuat kinerja menjadi lebih baik, sistem ChatOps meningkatkan kerjasama tim untuk menyelesaikan masalah-masalah dalam perilisan. Sistem juga dinilai lebih efektif dari pada sistem lama karena menghemat waktu dalam penyelesaian masalah, sehingga membantu lebih produktif. Pada pengujian usabilitas juga didapatkan bahwa sistem memuaskan, sehingga dapat diterapkan secara menyeluruh ke tim-tim yang lain.

5.2 Saran

Penelitian yang telah dilakukan penulis masih dapat dikembangkan dibeberapa aspek. Berikut ini saran yang dapat digunakan untuk mengembangkan sistem ChatOps:

a. Sistem otomasi ChatOps dapat dikembangkan lagi, dengan collaboration tools lainnya. Seperti Ansible, Kubernetes dan *monitoring tools*.
b. Hubot dapat dikembangkan secara multiuser, sehingga dapat membagi beberapa peran bot ke berbagai perintah yang krusial dan non-krusial, sehingga lebih aman.
c. Skenario perancangan sistem ChatOps perlu dioptimalkan, dengan penambahan berbagai kasus yang sesuai dan dapat diotomasikan.
d. Penelitian berikutnya diharapkan dapat menghasilkan sebuah dokumentasi yang *readable* dan lebih terstruktur.

DAFTAR PUSTAKA

Atalay, A. (2017). ChatOps with Mattermost and Hubot. Retrieved March 21, 2018, from https://medium.com/@ahmetatalay/chatops-with-mattermost-and-hubot-59d0b141b220

Aypay, A. (2010). Informatıon and Communıcatıon Technology (ICT) Usage and Achıvement of Turkısh Students in PISA 2006. *TOJET: The Turkish Online Journal of Educational Technology*, *9*(2), 116–124.

Balaji, S. (2018). What are Webhooks and Why You Can't Afford to Ignore Them - Chargebee's SaaS Dispatch. Retrieved October 9, 2018, from https://www.chargebee.com/blog/what-are-webhooks-explained/

Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, *33*(3), 42–52. http://doi.org/10.1109/MS.2016.64

Calcado, P. (2014). Backstage Blog - Building Products at SoundCloud —Part I: Dealing with the Monolith - SoundCloud Developers. Retrieved May 8, 2018, from https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-1-dealing-with-the-monolith

CaTechnology. (2013). TechInsights Report : What Smart Businesses Know About DevOps, (September), 300.

de Feijter, R. (2017). Towards the adoption of DevOps in software product organizations: A Maturity Model Approach, (May), 1–173.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2016). Microservices: yesterday, today, and tomorrow, (June). http://doi.org/10.13140/RG.2.1.3257.4961

Ellen, L., Riungu-kalliosaari, L., Mäkinen, S., & Lwakatare, L. E. (2016). DevOps Adoption Benefits and Challenges in Practice : A Case Study, 590–597.

Ercan, T. (2010). Effective use of cloud computing in educational institutions. *Procedia - Social and Behavioral Sciences*, *2*(2), 938–942. http://doi.org/10.1016/j.sbspro.2010.03.130

Fell, S. (2017). Are You Talking To Me? ChatOps and The Rise of Conversation-Driven DevOps - Electric Cloud. Retrieved October 9, 2018, from https://electric-cloud.com/blog/2017/03/talking-chatops-rise-conversation-driven-devops/

Fowler, M., Self-testing, M. Y. B., Machine, I., Deployment, A., Integration, I. C., Thoughts,

F., & Reading, F. (2014). Continuous Integration, 1–13.

Goldberg, Y. (2014). Scaling Gilt: from Monolithic Ruby Application to Distributed Scala Micro-Services Architecture. Retrieved May 8, 2018, from https://www.infoq.com/presentations/scale-gilt

Gupta, R. (2018). Chatbots Driving Enterprise Value in 2018 - DZone AI. Retrieved March 14, 2018, from https://dzone.com/articles/chatbots-driving-enterprise-value-in-2018-techjini-1

Hameed, T. (2006). ICT as an enabler of Socio-Economic Development. *Digital Opportunity Forum*, 26. http://doi.org/10.1007/978-0-387-35695-2

Hamunen, J. (2016). Challenges in adopting a Devops approach to software development and operations. Retrieved from https://aaltodoc.aalto.fi/handle/123456789/20766

Handiwidjojo, W., & Ernawati, L. (2016). Pengukuran Tingkat Ketergunaan (Usability) Sistem Informasi Keuangan Studi Kasus : Duta Wacana Internal Transaction (Duwit). *Juisi*, *02*(01), 49–55.

Ihde, S., & Parikh, K. (2015). From a Monolith to Microservices + REST: the Evolution of LinkedIn's Service Architecture. Retrieved May 8, 2018, from https://www.infoq.com/presentations/linkedin-microservices-urn

Jaffer, S., Ng'ambi, D., & Czerniewicz, L. (2007). The role of ICTs in higher education in South Africa: One strategy for addressing teaching and learning challenges. *International Journal of Education and Development Using Information and Communication Technology (IJEDICT)*, *3*(4), 131–142. Retrieved from http://www.google.co.ke/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&sqi=2&ved=0CFEQFjAE&url=http://ijedict.dec.uwi.edu/include/getdoc.php?id=2566&articl&ei=H1obUtWhJcTG0QXcioCoBA&usg=AFQjCNHBcmkKgyRr2xsck6BjvdJRk07fXA&sig2=fGh8vvHy2-nmCuJ-8GABnw

James, M. (n.d.). An Empirical Framework For Learning (Not a Methodology). Retrieved July 25, 2018, from http://scrummethodology.com/

James, M., & Walter, L. (2010). Scrum Reference Card Scrum Meetings. *New Society*, 1–6.

Jenhani, A. (2011). Cloud Computing in German Higher Education Institutions.

Jones, I. (2017). Improving your IT service delivery and operations with ChatOps. Retrieved September 8, 2018, from http://www.ianjones.co/2017/06/improving-your-it-service-delivery-and_89.html

Kihara, T., & Gichoya, D. (2014). Use of cloud computing platform for e-learning in

institutions of higher learning in Kenya. *2014 IST-Africa Conference Proceedings*, 1–6. http://doi.org/10.1109/ISTAFRICA.2014.6880638

Kong, S. C., & Li, K. M. (2009). Collaboration between school and parents to foster information literacy: Learning in the information society. *Computers and Education*, *52*(2), 275–282. http://doi.org/10.1016/j.compedu.2008.08.004

Kramer, S. (2011). Gigaom | The Biggest Thing Amazon Got Right: The Platform. Retrieved May 8, 2018, from https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/

Kratzke, N. (2014). A Lightweight Virtualization Cluster Reference Architecture Derived from Open Source PaaS Platforms. *Open Journal of Mobile Computing and Cloud Computing*, *1*(2), 17–30.

Kyrpychenko, L. (2018). Power up your organization with Chatops. Retrieved March 21, 2018, from https://hackernoon.com/power-up-your-organization-with-chatops-d2a1f33a022a

Lewis, J., & Fowler, M. (2014). Microservices. Retrieved May 8, 2018, from https://martinfowler.com/articles/microservices.html

Low, C., Chen, Y., & Wu, M. (2011). Understanding the determinants of cloud computing adoption. *Industrial Management & Data Systems*, *111*(7), 1006–1023. http://doi.org/10.1108/02635571111161262

Mak, R. (2017). Want to make the deployment process less scary? Build ChatOps in Slack. - Oursky Code Blog. Retrieved October 9, 2018, from https://code.oursky.com/human-and-cat-friendly-chatops/

Makoza, F. (2015). Cloud computing adoption in Higher Education Institutions of Malawi: An exploratory study. *Journal of Computing and ICT Research*, *9*(2), 37–54. http://doi.org/10.1007/s10586-015-0490-4

Masood, M. (2010). An initial comparison of educational technology courses for training teachers at malaysian universities: A comparative study. *Turkish Online Journal of Educational Technology*, *9*(1), 23–27.

Mauro, T. (2015). Microservices at Netflix: Lessons for Architectural Design. Retrieved May 8, 2018, from https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, *145*, 7. http://doi.org/10.1136/emj.2010.096966

Metz, C. (2015). The Most Important Startup's Hardest Worker Isn't a Person | WIRED. Retrieved October 9, 2018, from https://www.wired.com/2015/10/the-most-important-startups-hardest-worker-isnt-a-person/

Morgan, B. (2018). 3 Ways Slack Is A Customer-Centric Company. Retrieved October 9, 2018, from https://www.forbes.com/sites/blakemorgan/2018/08/29/3-ways-slack-is-a-customer-centric-company/#263b4e44455b

Morris, K. (2016). *Infrastructure as Code*. United States of America: O'Reilly Media, Inc. http://doi.org/10.1016/j.dss.2003.08.004

Nugroho, L. E. (2013). Kerangka Pengembangan Pendidikan Tinggi di Indonesia, 112–113. Retrieved from http://lukito.staff.ugm.ac.id/files/2013/02/Pemanfaatan-TI-di-Perguruan-Tinggi-Final.pdf

Null, C. (2015). Infrastructure as code is essential to your DevOps practices. Retrieved March 19, 2018, from https://techbeacon.com/infrastructure-code-engine-heart-devops

Okai, S., Uddin, M., Arshad, A., Alsaqour, R., & Shah, A. (2014). Cloud computing adoption model for universities to increase ICT proficiency. *SAGE Open*, *4*(3), 1–10. http://doi.org/10.1177/2158244014546461

Partogi, J. (2018). Scrum And DevOps. Retrieved August 5, 2018, from https://www.business2community.com/strategy/scrum-and-devops-02040373

Peham, T. (2017). GitLab vs GitHub: What Are The Key Differences? Retrieved October 9, 2018, from https://usersnap.com/blog/gitlab-github/

Richardson, J. W. (2008). ICT in Education Reform in Cambodia: Problems, Politics, and Policies Impacting Implementation. *Information Technologies and International Development*, *4*(4), 67–82. http://doi.org/10.1162/itid.2008.00027

Sachdeva, S. (2016). *Agile Methodologies*. Retrieved from http://www.agilemanifesto.org

Sarkar, S. (2012). The Role of Information and Communication Technology (ICT) in Higher Education for the 21st Century. *The Science Probe*, *1*(1), 30–41.

Shaik, Z. A., & Khoja, S. A. (2011). The role of ICT in shaping the future of Pakistan higher education system. *The Turkish Online Journal of Educational Technology*, *10*(1), 2011.

Shaikh, Zaffar, A. (2009). Usage, Acceptance, Adoption, and Diffusion of Information & Communication Technologies in Higher Education: A Measurement of Critical Factors, *9*(2), 63–80.

Soto, A. (2012). Installing And Running Hubot - A Bot for Automation and Fun! - DZone DevOps. Retrieved October 9, 2018, from https://dzone.com/articles/installing-and-

running-hubot

Streule, T., Miserini, N., Bartlomé, O., Klippel, M., & De Soto, B. G. (2016). Implementation of Scrum in the Construction Industry. *Procedia Engineering*, *164*(June), 269–276. http://doi.org/10.1016/j.proeng.2016.11.619

Sultan, N. (2010). Cloud computing for education: A new dawn? *International Journal of Information Management*, *30*(2), 109–116. http://doi.org/10.1016/j.ijinfomgt.2009.09.004

Unesco. (2009). *Guide To Measuring Information And Communication Technologies (ICT) In Education*. http://doi.org/10.15220/978-92-9189-078-1-en

Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., & Gil, S. (2015). Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud Evaluando el Patrón de Arquitectura Monolítica y de Micro Servicios Para Desplegar Aplicaciones en la Nube. *10th Computing Colombian Conference*, 583–590. http://doi.org/10.1109/ColumbianCC.2015.7333476

Villamizar, M., Garces, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M., … Lang, M. (2016). Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures. *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, 179–182. http://doi.org/10.1109/CCGrid.2016.37

Wilsenach, R. (2015). DevOpsCulture. Retrieved April 15, 2018, from https://martinfowler.com/bliki/DevOpsCulture.html#footnote-tools

Ya ’u Gital, A., & Zambuk, F. U. (2011). Cloud Computing: Solution to ICT in Higher Education in Nigeria. *Advances in Applied Science Research*, *2*(6), 364–369. Retrieved from http://www.researchgate.net/publication/265226046_Cloud_Computing_Solution_to_ICT_in_Higher_Education_in_Nigeria

Yusuf, M. O., & Afolabi, A. O. (2010). Effects of computer assisted instruction (CAI) on secondary school students’ performance in biology. *Turkish Online Journal of Educational Technology*, *9*(1), 62–69.

Zyane, R. (2017). How ChatOps Can Help You DevOps Better – Chatbots Magazine. Retrieved October 9, 2018, from https://chatbotsmagazine.com/how-chatops-can-help-you-devops-better-5-minutes-read-507438c156bf

LAMPIRAN

a. File Dockerfile

```
FROM node:carbon

RUN mkdir /opt/hubot
WORKDIR /opt/hubot

RUN npm install -g hubot coffee-script yo generator-hubot

RUN useradd -ms /bin/bash hubot-slack
RUN chown -R hubot-slack /opt/hubot
USER hubot-slack

RUN yo hubot --owner="BSI" --name="Gandalf" --
description="Delightfully aware robutt" --adapter=slack --defaults
--allow-root

RUN npm install hubot-slack
RUN npm install hubot-urban
RUN npm install hubot-youtube
RUN yarn add hubot-command-log
RUN npm i hubot-command-log

ADD external-scripts.json /opt/hubot/external-scripts.json
ADD scripts/* /opt/hubot/scripts/

RUN rm hubot-scripts.json

ENV HUBOT_SLACK_TOKEN=[redacted]

CMD ["./bin/hubot", "--adapter", "slack"]
```

b. File docker-compose.yml

```
version: "2"
services:
  hubot-slack:
    build:
      context: .
    restart: always
    user: hubot-slack
    environment:
      - HUBOT_GITLAB_URL=[redacted]
      - HUBOT_GITLAB_TOKEN=[redacted]
```

c. File group_gitlab.coffee

```
url = process.env.HUBOT_GITLAB_URL
git_token = process.env.HUBOT_GITLAB_TOKEN

module.exports = (robot) ->
  robot.respond /list (projects|users|members) in
(projects|groups) (.*)/i, (msg) ->
    pilih1 = msg.match[1]
    pilih2 = msg.match[2]
    msgname = msg.match[3]
    name = msgname.split('/').join('%2F');

    path =
"#{url}/#{pilih2}/#{name}/#{pilih1}?private_token=#{git_token}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
      if 404 == res.statusCode
        content = JSON.parse body
        msg.reply "Gitlab cannot found #{content.message}, cek
your path \n using command 'search #{pilih2} <keyword>' to know
the right path"
      else if 200 <= res.statusCode < 400
        if body in ['[]']
```

```
          msg.reply "#{pilih1} in #{pilih2} is empty"
        else
          if pilih1 in ['projects']
            for project in JSON.parse body
              msg.reply "projects name : #{project.name}, path :
#{project.path_with_namespace}, with id: #{project.id}, url :
#{project.web_url}"
          else
            for user in JSON.parse body
              msg.reply "username : #{user.username}, with id :
#{user.id}"
      else
        content = JSON.parse body
        msg.reply "Gitlab says: Status #{content.message}"

  robot.respond /share projects (.*) to groups (.*)/i, (msg) ->
    projectname = msg.match[1]
    groupname = msg.match[2]
    project = projectname.split('/').join('%2F');
    group = groupname.split('/').join('%2F');

    path =
"#{url}/groups/#{group}/projects/#{project}?private_token=#{git_to
ken}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.post() (err,res,body) ->
      if 404 == res.statusCode
        content = JSON.parse body
        msg.reply "Gitlab says #{content.message}, cek your path
\n using command 'search projects <keyword>' and 'search groups
<keyword>' to know the right path"
      else if 200 <= res.statusCode < 400
        content = JSON.parse body
        msg.reply "project successfully add to group #{groupname},
you can check in #{content.web_url}"
```

```
      else
        content = JSON.parse body
        msg.reply "Gitlab says: Status #{content.message}, check
again your command"

  robot.respond /search (projects|groups|users|issues) by (.*)/i,
(msg) ->
    pilih = msg.match[1]
    key = msg.match[2]

    path =
"#{url}/#{pilih}?private_token=#{git_token}&search=#{key}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
      if 200 <= res.statusCode < 400
        if body in ['[]']
          msg.reply "#{pilih} is empty"
        else
          if pilih in ['users']
            for user in JSON.parse body
              msg.reply "username : #{user.username}, with id :
#{user.id}"
          else if pilih in ['projects']
            for project in JSON.parse body
              msg.reply "projects name : #{project.name}, path :
#{project.path_with_namespace}, with id : #{project.id}, url :
#{project.web_url}"
          else if pilih in ['issues']
            for issue in JSON.parse body
              msg.reply "issues iid : #{issue.id}, description :
#{issue.description}, url : #{issue.web_url}"
          else
            for group in JSON.parse body
              msg.reply "group name : #{group.name}, url :
#{group.web_url}, with id : #{group.id}"
```

```
      else
        content = JSON.parse body
        msg.reply "Gitlab says: Status #{content.message}"
```

d. File issue-scheduling.coffee

```
url = process.env.HUBOT_GITLAB_URL
git_token = process.env.HUBOT_GITLAB_TOKEN

module.exports = (robot) ->

  robot.respond /create issue (.*) for (.*), (.*)/i, (msg) ->
    issuename = msg.match[1]
    projectname = msg.match[2]
    description = msg.match[3]
    project = projectname.split('/').join('%2F')
    issue = issuename.split(' ').join('%20');

    path =
"#{url}/projects/#{project}/issues?private_token=#{git_token}&titl
e=#{issue}&description=#{description}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.post() (err,res,body) ->
      if 200 <= res.statusCode < 400
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot create issue :#{content.message}"

  robot.respond /comment issue (.*) (.*), (.*)/i, (msg) ->
    issueid = msg.match[1]
    projectname = msg.match[2]
    description = msg.match[3]
    project = projectname.split('/').join('%2F');
    path =
"#{url}/projects/#{project}/issues/#{issueid}/notes?private_token=
#{git_token}&body=#{description}"
```

```
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.post() (err,res,body) ->
      if 200 <= res.statusCode < 400
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot comment issue
:#{content.message}"

  robot.respond /closed issue (.*) (.*)/i, (msg) ->
    id = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F')

    path =
"#{url}/projects/#{project}/issues/#{id}?private_token=#{git_token
}&state_event=close"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.put() (err,res,body) ->
      if 200 <= res.statusCode < 400
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot closed issue :#{content.message}"

  robot.respond /assignee issue (.*) (.*) for (.*)/i, (msg) ->
    id = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F')
    username = msg.match[3]

    path =
"#{url}/users?private_token=#{git_token}&search=#{username}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
```

```
      if 404 == res.statusCode
        content = JSON.parse body
        msg.reply "Gitlab says #{content.message}, cek your path
\n using command 'search users <keyword> to know the right path"
      else if 200 <= res.statusCode < 400
        if body in ['[]']
          msg.reply "username cannot found, check your input for
username gitlab"
        else
          for user in JSON.parse body
            user_id = user.id
            path =
"#{url}/projects/#{project}/issues/#{id}?private_token=#{git_token
}&assignee_ids=#{user_id}"
            #msg.send path
            req = msg.http(path)
            req.header('Content-Length', 0)
            req.put() (err,res,body) ->
              if 200 <= res.statusCode < 400
                content = JSON.parse body
                msg.reply "assignee this issue #{content.title}
for #{username}"
              else
                content = JSON.parse body
                msg.reply "Gitlab cannot assignee issue
:#{content.message}"
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot assignee Status
#{content.message}"

  robot.respond /scheduling pipeline (.*) for (.*)/i, (msg) ->
    pipeline = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F');
```

```
    path =
"#{url}/projects/#{project}/pipeline_schedules?private_token=#{git
_token}&description=#{pipeline}&ref=master&active=true&cron_timezo
ne=Asia/Jakarta"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.query(_render: "json", cron: "0 1 * * 5")
    req.post() (err,res,body) ->
      if 200 <= res.statusCode < 400
        content = JSON.parse body
        msg.reply "pipelines scheduling created
#{content.description}, cron #{content.cron}, timezone
#{content.cron_timezone}, status : #{content.active}"
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot create schedule
:#{content.message}"
```

e. File project_gitlab.coffee

```
url = process.env.HUBOT_GITLAB_URL
git_token = process.env.HUBOT_GITLAB_TOKEN

module.exports = (robot) ->
  robot.respond /create projects (.*) for (.*)/i, (msg) ->
    project = msg.match[1]
    username = msg.match[2]

    path =
"#{url}/users?private_token=#{git_token}&search=#{username}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
      if 404 == res.statusCode
        content = JSON.parse body
        msg.reply "Gitlab says #{content.message}, cek your path
\n using command 'search users <keyword> to know the right path"
```

```
      else if 200 <= res.statusCode < 400
        if body in ['[]']
          msg.reply "username cannot found, check your input for
username gitlab"
        else
          for user in JSON.parse body
            user_id = user.id
            path =
"#{url}/projects/user/#{user_id}?private_token=#{git_token}"
            req = msg.http(path)
            req.query(_render: "json", name: project)
            req.header('Content-Length', 0)
            req.post() (err, res, body) ->
              if 200 <= res.statusCode < 400
                content = JSON.parse body
                id_project = content.id
                msg.reply "project created with id: #{content.id},
name: #{content.name}, path namespace:
#{content.path_with_namespace}, url repo:
#{content.http_url_to_repo} "

                path =
"#{url}/projects/#{id_project}/triggers?private_token=#{git_token}
"
                req = msg.http(path)
                req.query(_render: "json", description: project)
                req.header('Content-Length', 0)
                req.post()
              else
                content = JSON.parse body
                msg.reply "Gitlab cannot create project Status
#{content.message.name}"
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot create Status #{content.message},
check configuration access token"
```

```
  robot.respond /join (.*) to (groups|projects) (.*)/i, (msg) ->
    username = msg.match[1]
    pilih = msg.match[2]
    address = msg.match[3]
    alamat = address.split('/').join('%2F');

    path =
"#{url}/users?private_token=#{git_token}&username=#{username}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
      if 404 == res.statusCode
        content = JSON.parse body
        msg.reply "Gitlab says #{content.message}, cek your path
\n using command 'search groups <keyword>', 'search projects
<keyword>' and 'search users <keyword>' to know the right path"
      else if 200 <= res.statusCode < 400
        if body in ['[]']
          msg.reply "username cannot found, check your input for
username gitlab"
        else
          for user in JSON.parse body
            user_id = user.id
            path =
"#{url}/#{pilih}/#{alamat}/members?private_token=#{git_token}&user
_id=#{user_id}&access_level=30"
            req = msg.http(path)
            req.query(_render: "json")
            req.header('Accept','application/json')
            req.post() (err, res, body) ->
              if 200 <= res.statusCode < 400
                content = JSON.parse body
                msg.reply "user #{content.name}, with access level
#{content.access_level} successfully join in #{pilih}: #{address}"
              else
                content = JSON.parse body
```

```
                msg.reply "Gitlab says: Status #{content.message}"

      else
        content = JSON.parse body
        msg.reply "Gitlab cannot found users Status
#{content.message}, check configuration access token"
```

f. File re-build_gitlab.coffee

```
url = process.env.HUBOT_GITLAB_URL
git_token = process.env.HUBOT_GITLAB_TOKEN

module.exports = (robot) ->

  robot.respond /build projects (.*) (.*)/i, (msg) ->
    projectname = msg.match[1]
    branch = msg.match[2]
    project = projectname.split('/').join('%2F');

    path =
"#{url}/projects/#{project}/triggers?private_token=#{git_token}"
    req = msg.http(path)
    req.header('Content-Length', 0)
    req.get() (err,res,body) ->
      if 404 == res.statusCode
        content = JSON.parse body
        msg.reply "Gitlab says #{content.message}, cek your path
\n using command 'search projects <keyword>' to know the right
path"
      else if 200 <= res.statusCode < 400
        if body in ['[]']
          path =
"#{url}/projects/#{project}/triggers?private_token=#{git_token}"
          req = msg.http(path)
          req.query(_render: "json", description: branch)
          req.header('Content-Length', 0)
          req.post() (err, res, body) ->
```

```
            if 200 <= res.statusCode < 400
              content = JSON.parse body
              token = content.token
              path =
"#{url}/projects/#{project}/trigger/pipeline?token=#{token}"
              req = msg.http(path)
              req.query(_render: "json", ref: branch)
              req.header('Content-Length', 0)
              req.post() (err, res, body) ->
                if 200 <= res.statusCode < 400
                  content = JSON.parse body
                  msg.reply "build successfully with id:
#{content.id}, branch: #{content.ref}, status: #{content.status},
url: #{content.web_url}"
                else
                  content = JSON.parse body
                  msg.reply "Gitlab cannot build Status
#{content.message.base}"
            else
              content = JSON.parse body
              msg.reply "Gitlab cannot create trigger
#{content.message}"
        else
          for trigger in JSON.parse body
            token = trigger.token
            path =
"#{url}/projects/#{project}/trigger/pipeline?token=#{token}"
            req = msg.http(path)
            req.query(_render: "json", ref: branch)
            req.header('Content-Length', 0)
            req.post() (err, res, body) ->
              if 200 <= res.statusCode < 400
                content = JSON.parse body
                msg.reply "build successfully with id:
#{content.id}, branch: #{content.ref}, status: #{content.status},
url: #{content.web_url}"
```

```
              else
                content = JSON.parse body
                msg.reply "Gitlab cannot build Status
#{content.message.base}, check your branch or using tag for
rebuild -> create new tag using this comment (create tag
<tag_name> for <project_path> <massage>)"
      else
        content = JSON.parse body
        msg.reply "Gitlab cannot build Status #{content.message}"

  robot.respond /create tag (.*) for (.*) (.*)/i, (msg) ->
    tagname = msg.match[1]
    projectname = msg.match[2]
    project = projectname.split('/').join('%2F');
    massage = msg.match[3]

    path =
"#{url}/projects/#{project}/repository/tags?private_token=#{git_to
ken}&tag_name=#{tagname}&ref=master&message=#{massage}"
    req = msg.http(path)
    req.query(_render: "json")
    req.header('Content-Length', 0)
    req.post() (err, res, body) ->
        if 200 <= res.statusCode < 400
        else
          content = JSON.parse body
          msg.reply "Gitlab cannot create tag Status
#{content.message}"
```