Otomasi Pengujian Antarmuka Pengguna pada Aplikasi Web myITS HumanCapital di dalam Implementasi CI/CD

Muthia Qurrota Akyun, Rizky Januar Akbar, dan Hadziq Fabroyir
Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: rja@its.ac.id

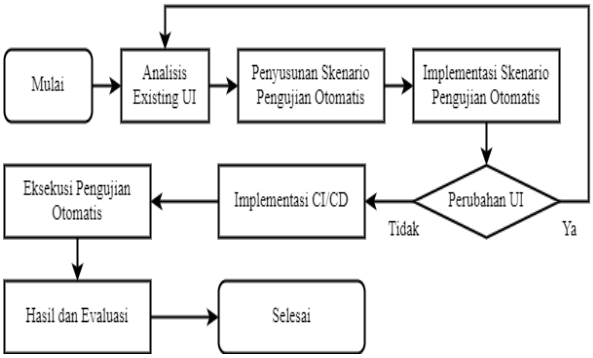
Abstrak*—Institut Teknologi Sepuluh Nopember (ITS) telah mengembangkan salah satu sistem yang bertujuan untuk memenuhi berbagai kebutuhan terkait kepegawaian ITS yang bernama myITS HumanCapital (MIHC). Dalam melakukan proses pengembangan dan pemeliharaan perangkat lunak dalam jangka panjang, dibutuhkan pengujian antarmuka pengguna aplikasi yang memiliki peran penting dalam memberikan produk yang berkualitas tinggi. Proses pengujian secara manual seringkali tidak akurat, kurang dapat diandalkan, dan lebih memakan waktu daripada pengujian otomatis. Penelitian inimengajukan *automated testing* dengan menerapkan *Continuous Integration / Continuous Deployment* (CI/CD). Pada proses pengujian otomatis, aplikasi web MIHC menggunakan *framework* Serenity untuk menuliskan *script automation* dan Jenkins untuk integrasi *script automation* di dalam implementasi CI/CD. Pengujian otomatis di dalam implementasi CI/CD ini dapat mempercepat proses pengujian, membuat pembaruan aplikasi lebih tepat waktu, dan membawa efisiensi waktu pengiriman aplikasi sebagai sebuah manfaat bagi pengguna. Setelah dilakukan uji coba menggunakan *framework* Serenity, didapatkan 108 skenario pengujian. Integrasi pengujian otomatis dengan Jenkins untuk mencapai konsep CI/CD dengan *build trigger* berdasarkan perubahan kode oleh pengembang tidak memungkinkan untuk diterapkan pada aplikasi web MIHC sehingga dilakukan pendekatan lain, yakni *nightly build*. Berdasarkan tahapan pengujian otomatis yang telah dilakukan, pengujian yang dijalankan mengalami keberhasilan 97% dan mengalami kegagalan 3% dikarenakan terdapat bug yang belum dilakukan perbaikan.

Kata Kunci—Automated Testing, Continuous Integration, Continuous Deployment.

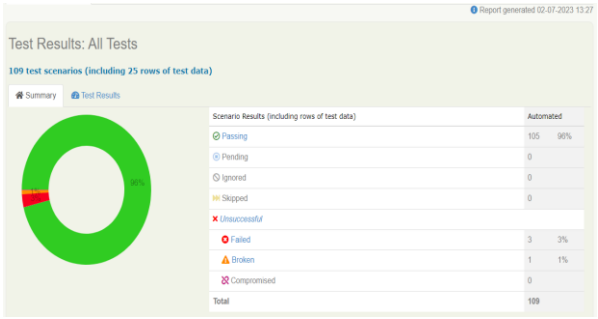
I. PENDAHULUAN

INSTITUT Teknologi Sepuluh Nopember (ITS) telah mengembangkan sebuah aplikasi web bernama myITS HumanCapital (MIHC). MIHC merupakan aplikasi web yang bertujuan untuk memenuhi berbagai kebutuhan terkait kepegawaian ITS. Aplikasi web MIHC digunakan untuk mengelola dan mengembangkan SDM kepegawaian ITS. Dalam pengembangannya, aplikasi web MIHC terbagi menjadi beberapa modul yaitu kinerja, presensipegawai, PAK (Penilaian Angka Kredit), karir tendik, rekrutmen, layanan administrasi kepegawaian, dan portofolio pegawai [1]1]. Dalam proses pengembangan aplikasi web MIHC memerlukan ketepatan mengingat tujuan untuk memenuhi berbagai kebutuhan terkait kepegawaian ITS.

Keterlibatan manusia dalam pengembangan perangkat lunak yang aman dan andal rentan terhadap kesalahan [2]. Oleh karena itu, untuk menjaga kualitas dari perangkat lunak yang dikembangkan, sebagian besar industri melakukan aktivitas *testing* selama proses pengembangan perangkat lunak [3]. *Testing* dilakukan untuk memvalidasi bahwa perangkat lunak dapat dijalankan sesuai ekspektasi dan untuk menemukan bagian yang mungkin gagal [3].



Gambar 1. Alur Pengerjaan Penelitian.



Gambar 2. Report Regression Test.

Testing merupakan proses mengevaluasi program *software* untuk tujuan menemukan *bug* [2]. Tujuan keseluruhan dari *testing* bukan untuk menunjukkan bahwa aplikasi bebas dari kesalahan, tetapi untuk memberikan keyakinan bahwa aplikasi bekerja dengan baik sebelum instalasi [2]. *Testing* juga memastikan perangkat lunak yang diuji berfungsi seperti yang diharapkan. *Testing* merupakan salah satu aktivitas yang memerlukan lebih dari 50% dari *resource* proyek pengembangan perangkat lunak [4]. Melihat kondisi tersebut, meningkatkan efektivitas dari proses *testing* dapat mengurangi biaya dari pengembangan perangkat lunak kedepannya [4].

Aktivitas *testing* dapat dibagi menjadi dua metode, yaitu secara manual maupun *automated* [2]. *Manual testing* pada umumnya dilakukan untuk memverifikasi perangkat lunak pada fase akhir siklus pengembangan maupun untuk menemukan *bug* yang sulit ditemukan dengan menggunakan pengetahuan dan pengalaman seorang SQA *Engineer* [4]. Pengujian otomatis melibatkan *automation tools* dengan atau tidak menggunakan *framework*. Pengujian otomatis meningkatkan akurasi pengujian dan mempersingkat waktu

Tabel 1.
Daftar feature yang diujikan

No	Nama Feature	Skenario yang Diuji
1	Profil	Verfikasi data
2	Biodata	Verfikasi data, cari data, dan edit data
3	Anggota Keluarga	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
4	Riwayat Pendidikan	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
5	Portofolio	Verifikasi data
6	Karya Cipta	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
7	Hak Kekayaan Intelektual	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
8	Keanggotaan Organisasi Profesi	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
9	Kepanitiaan	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
10	Pembicara	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
11	Pengembangan Diri	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
12	Penghargaan	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
13	Riwayat Pekerjaan	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
14	Sertifikasi	Verfikasi data, tambah data, cari data, filter data, hapus data, dan edit data
15	Penelitian	Verfikasi data, tambah data, cari data, hapus data, dan edit data

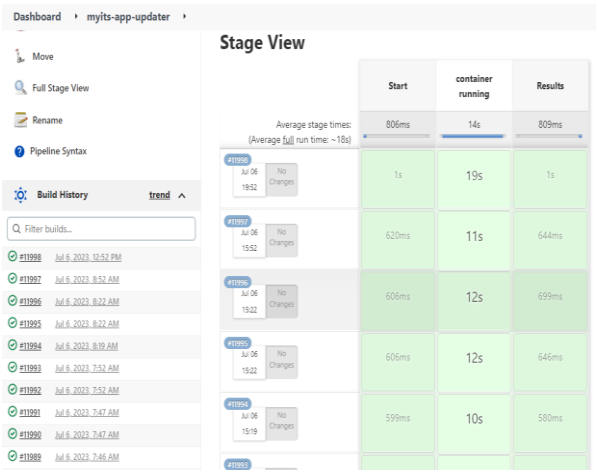
pengujian [2]. Seiring dengan bertambahnya kompleksitas dari perangkat lunak yang diuji, maka biaya yang diperlukan untuk melakukan *testing* menjadi lebih mahal sehingga metode tradisional seperti *manual testing* menjadi tidak *sustainable* secara ekonomi [3].

Pengujian antarmuka pengguna atau dapat disebut dengan *User Interface* (UI) *Testing*, dapat dijalankan secara manual maupun otomatis. Penguian otomatis (*Automated Testing*) melibatkan *automation tools* dengan atau tidak menggunakan *framework*. Keberhasilan pengujian UI menjadi sangat penting. Hal ini dapat menjadi tolak ukur kualitas perangkat lunak secara keseluruhan [2][5].

Pengembangan perangkat lunak tanpa adanya *automation* akan memakan banyak waktu setiap adanya perubahan kode. Konsep *Continuous Integration* / *Continuous Delivery* (CI/CD) dapat membantu pekerjaan *programmer* dalam menjalankan pengujian otomatis secara berkala. CI/CD merupakan bagian dari DevOps. DevOps merupakan praktik dari proses pengembangan aplikasi dimana bagian *development* dan *operations* dilakukan secara otomatis dan terintegrasi satu sama lain [5]. DevOps membuat proses pengembangan aplikasi menjadi efektif dan efisien [6]. Oleh karena itu, pada penelitian ini akan membahas mengenai "Otomasi Pengujian Antarmuka Pengguna pada Aplikasi Web myITS HumanCapital di dalam Implementasi CI/CD".

Tabel 2.
Hasil Waktu Pengujian Otomatis

No	Tanggal	Waktu Eksekusi Pengujian Otomatis	Tes Tercepat (detik)	Tes Terlama (detik)	Rata-rata Waktu Tes (detik)
1	2/7/2023	51 menit 44 detik	0	63	28
2	3/7/2023	57 menit 52 detik	0	112	32
3	4/7/2023	52 menit 6 detik	0	60	28
4	5/7/2023	1 jam 10 menit 5 detik	0	85	38
5	6/7/2023	1 jam 8 menit 48 detik	0	88	38
Rata-rata		1 jam 7 detik	0	81,6	32,8



Gambar 3. Tampilan *History Build Job* myits-app-updater.

Penelitian ini mengimplementasikan pengujian otomatis di dalam implementasi CI/CD sehingga diperoleh pengembangan perangkat lunak yang cepat dan hasil pengujian yang baik untuk meminimalisir *bug* yang ditemukan dalam aplikasi web MIHC pada fase *production.*

II. TINJAUAN PUSTAKA

A. *Pengujian Otomatis*

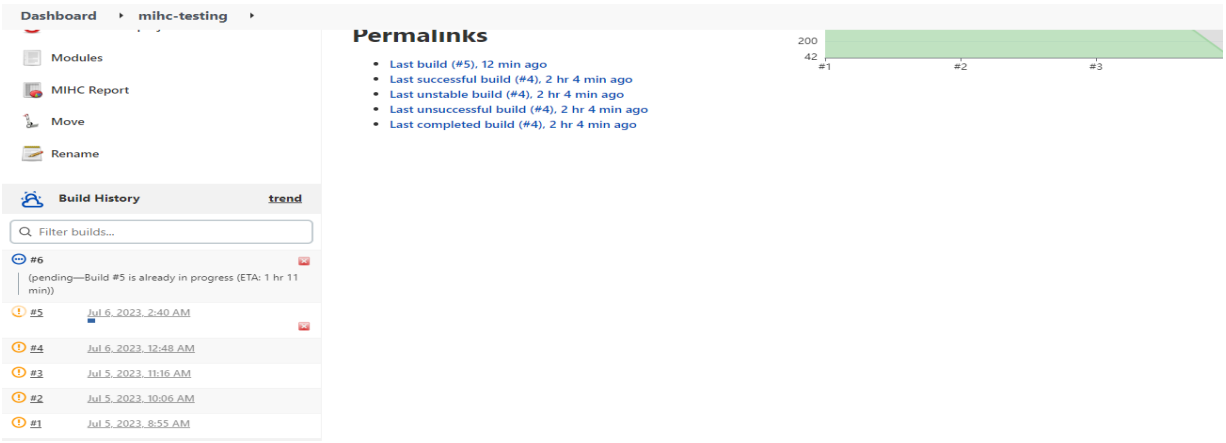
Pengujian otomatis atau *automated testing* adalah salah satu metode dalam Software Testing dimana perangkat lunak diuji dengan perangkat lunak maupun *tools*. Pengujian ini berbeda dengan pengujian manual karena tidak memerlukan intervensi manusia saat pengujian sedang berjalan .

B. *Serenity*

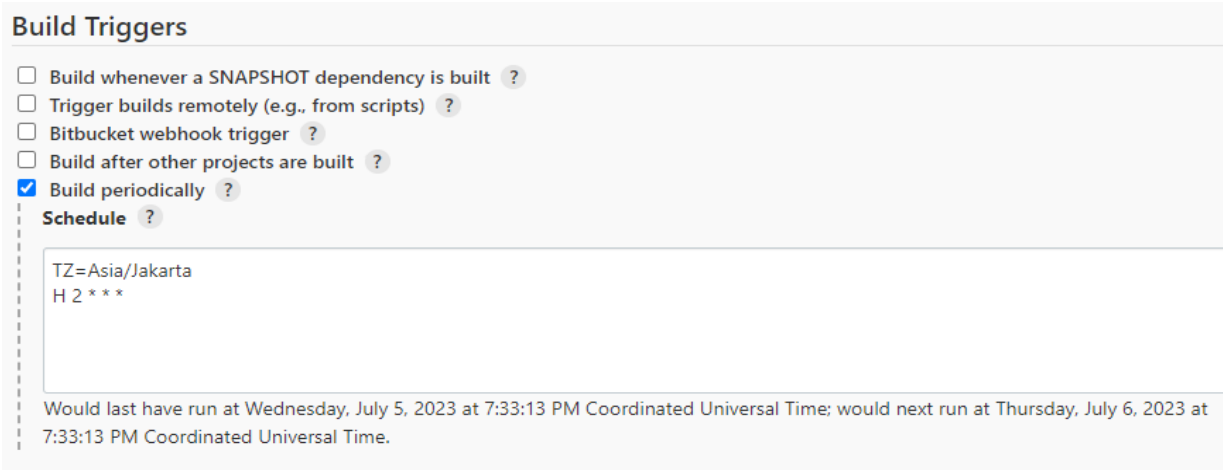
Serenity merupakan *open source framework* yang bertujuan untuk membuat ide *livingdocumentation* menjadi kenyataan. *Living documentation* artinya dokumentasi yang otomatis ter-update setiap kali aplikasi dibangun. Serenity dapat membantu menuliskan *acceptance test*menjadi lebih bersih dan mudah dipelihara. Selain itu, Serenity juga dapat menjalankan *regression test* menjadi lebih cepat. Serenity juga menggunakan hasil tes untuk menghasilkan laporan yang mendokumentasikan dan menjelaskan apa yang dilakukan aplikasi dan cara kerjanya. Selain itu, Serenity juga terintegrasi dengan beberapa tools untuk melakukan tes secara otomatis seperti Selenium, RestAssured, Cucumber, dan lain-lain.

C. *Cucumber*

Cucumber merupakan alat yang mendukung *Behaviour-*



Gambar 4. Tampilan *History Build Job* mihc-testing.



Gambar 5. Penambahan *Job Schedule* pada *Job* mihc-testing.

Driven Development (BDD). BDD berfokus pada perilaku sistem. BDD menggunakan bahasa asli mereka sebagai jembatan antara *business people* dan *technical people*. Notasi original dari BDD, khususnya Given-When-Then, dekat dengan bahasa sehari-hari .

D. Selenium

Selenium adalah tools untuk melakukan otomasi browser pada platform web. Selenium memungkinkan pengguna untuk merancang *scripts* dalam bahasa pemrograman seperti Ruby, Java, NodeJS, PHP.

E. Selenium dengan Java

Dari semua bahasa yang dapat digunakan untuk menulis *script* di Selenium, Java menjadi bahasa pemrograman yang paling populer digunakan. Selenium dan Java menjadi perpaduan sempurna untuk menjalankan pengujian otomatis pada *web browser* yang berbeda. Java memilikijaringan luas *active web developer* yang secara aktif berkontribusi dalam menulis *test case*. Eksekusi program juga lebih cepat menggunakan bahasa pemrograman Java dibandingkan dengan bahasa pemrograman lainnya.

F. Selenium dengan Page Object Model

Page Object Model (POM) adalah *design pattern* di Selenium yang membuat sebuah *object repository* untuk menyimpan semua elemen web. Hal ini berguna dalam mengurangi duplikasikode dan meningkatkan pemeliharaan *test case*. Dalam POM, setiap halaman aplikasi web dipertimbangkan sebagai *file class*. Setiap file class hanya akan berisi elemen halaman web yang sesuai. Kemudian penguji membuat *script* pengujian menggunakan elemen ini untuk melakukan berbagai tindakan. POM membantu dalam mempermudah penggunaan kembali *test code* untuk suatu halaman dan menggunakannya kembali untuk *test case* lainnya sehingga mengurangi duplikasi kode. Kode juga menjadi lebih mudah dipelihara karena setiap elemen web halaman berada di *file class* yang berbeda.

G. CI/CD dengan Jenkins

Jenkins adalah server otomatisasi *open source* mandiri yang dapat digunakan untuk mengotomatisasi semua jenis tugas yang berkaitan dengan membangun, menguji, dan *delivery*atau *deploy* perangkat lunak. Jenkins dapat diinstal melalui paket sistem asli, Docker, atau bahkan menjalankan standalone oleh mesin apa pun dengan Java Runtime Environment (JRE)yang diinstal. Jenkins digunakan untuk mengimplementasikan alur kerja CI/CD atau biasa disebut dengan pipelines.

H. MIHC

MIHC merupakan aplikasi web yang bertujuan untuk memenuhi berbagai kebutuhan terkait kepegawaian ITS. Aplikasi web MIHC digunakan untuk mengelola dan mengembangkan SDM kepegawaian ITS. Dalam pengembangannya, aplikasi web MIHC terbagi menjadi beberapa modul yaitu kinerja, presensi pegawai, PAK (Penilaian Angka Kredit), karir tendik, rekrutmen, layanan administrasi kepegawaian, dan portofolio pegawai [1].

III. METODOLOGI

Tahapan proses yang dilakukan untuk mengerjakan penelitian ini dapat dilihat pada Gambar 1.

A. Analisis Existing UI

Pada penelitian ini, sistem yang digunakan sebagai bahan uji adalah UI aplikasi web MIHC pada modul Profil dan Portfolio. MIHC berfungsi untuk menyediakan informasi profil, portofolio, dan juga data kepegawaian secara online bagi dosen dan tenaga kependidikan di ITS. Web aplikasi ini mengintegrasikan kebutuhan data kepegawaian untuk mempermudah pencatatan dan pemantauan produktivitas individual sumber daya manusia ITS.

Fitur yang akan diuji secara otomatis pada penelitian ini meliputi fitur-fitur yang ada pada modul profil dan modul portofolio. Fitur-fitur yang ada pada modul profil meliputi fitur biodata, fitur anggota keluarga, dan fitur riwayat pendidikan. Modul profil digunakan untuk menyimpan data pribadi dari pegawai. Fitur-fitur yang ada pada modul portofolio meliputi fitur karya cipta, keanggotanan organisasi profesi, kepanitiaan, hak kekayaan intelektual, penelitian, pembicara, pengembangan diri, penghargaan, riwayat pekerjaan, dan sertifikasi. Modul portofolio digunakan untuk mempermudah pencatatan dan pemantauan produktivitas individual sumber daya manusia ITS**.**

B. Penyusunan Skenario Pengujian Otomatis

Skenario pengujian adalah rancangan keadaan yang akan diujikan untuk memastikan fungsionalitas sistem sudah sesuai dengan kebutuhan atau masih terdapat kesalahan yang terjadi sehingga masih dibutuhkan perbaikan. Skenario pengujian otomatis yang akan diuji terdiri dari skenario memverifikasi data, menambahkan data, mencari data, memfilter data, menghapus data, dan mengedit data.

C. Implementasi Skenario Pengujian Otomatis

Implementasi skenario pengujian otomatis meliputi pembuatan kode pemrograman dari skenario pengujian yang telah dibuat sebelumnya. Dalam membangun proyek pengujian otomatis ini digunakan bahasa pemograman Java dengan kerangka kerja Serenity yang terintegrasi Selenium. Sebelum dilakukan implementasi skenario pengujian, perlu dilakukan *setting* untuk membuat proyek Serenity dengan POM *design pattern*. Berikut merupakan langkah-langkah yang perlu dilakukan untuk menyiapkan proyek.

1) Instalasi Java Development Kit (JDK)

Versi JDK yang digunakan adalah JDK 8.

2) Instalasi Maven

Versi Maven yang digunakan adalah Maven 3.8.4.

3) Instalasi IntelliJ

IntelliJ digunakan sebagai lingkungan pengembangan pengujian otomatis yang akan dibangun

4) Pembuatan Projek Serenity

Pengujian otomatis yang akan dibangun menggunakan Serenity sebagai kerangka kerja penulisan

5) Pembuatan File Feature

File feature merupakan file yang berisikan langkah-langkah skenario pengujian yang dilakukan.

6) Pembuatan Page Object

Pada pengujian UI ini digunakan POM Design Pattern (satu Class Page mewakili satu halaman UI).

7) Pembuatan Step Class

Step class merupakan *class* yang digunakan untuk menghubungkan antara *file* feature dengan kode java pada page *class.*

8) Konfigurasi Webdriver

Webdriver diperlukan untuk mengendalikan peramban web secara terprogram. Peramban digunakan sebagai tempat untuk menjalankan skenario pengujian. WebDriver yang digunakan pada pegujian otomatis penelitian ini adalah ChromeDriver. Penelitian ini menggunakan Bonigarcia webdriver manager.

9) Implementasi Properties

Penelitian Akhir ini memiliki 2 *file* properties, yaitu serenity.properties dan application.properties. *File* serenity.properties digunakan untuk mengkonfigurasi webdriver seperti pada penjelasan konfigurasi webdriver. *File* application,properties digunakan untuk menyimpan data uji yang akan dipakai dalam pengujian otomatis. *File* properties ini hanya berisikan variabel-variabel yang digunakan untuk data uji.

Langkah-langkah diatas merupakan implementasi yang dilakukan pada pengerjaan penelitian ini. Pengujian otomatis dilakukan berdasarkan skenario yang telah dibuat. Kasus pengujian dalam penelitian ini ditulis dalam bentuk BDD dimana bentuk ini terdiri dari kondisi awal, perlakuan yang diberikan dan hasil yang didapatkan. Skenario dibangun dalam beberapa *file* .feature berdasarkan fitur yang ada pada modul profil dan modul portofolio. *File* .feature yang diuji pada penelitian ini dapat dilihat pada Tabel 1.

D. Perubahan UI

Kondisi perubahan UI meliputi kondisi ‘ya’ dan ‘tidak’. Kondisi ‘ya’ menunjukkan adanya perubahan pada UI saat sprint sehingga diperlukan penyusunan skenario baru yang sesuai dengan perubahan yang ada pada UI. Sedangkan kondisi ‘tidak’ menunjukkan tidak adanya perubahan pada UI sehingga proses dapat dilanjutkan.

E. Implementasi CI/CD

Platform Jenkins digunakan untuk mengotomatisasi penjalanan pengujian. Pengujian berjalan pada saat terdapat perubahan kode yang dilakukan oleh pengembang. Pengujian ini nantinya akan menjalankan kasus uji pada aplikasi web MIHC. Dengan kata lain, pengujian otomatis ini akan menjalankan kasus uji terhadap sistem web yang ada pada server. Sistem pengujian ini dijalankan dalam platform Jenkins. Jenkins bertugas menjalankan proses pengujian otomatis ini ketika terjadi perubahan dalam kode.

F. Eksekusi Pengujian Otomatis

Tahap eksekusi pengujian otomatis dilakukan dengan menjalankan *script automation* pada aplikasi web MIHC. Pengujian ini dijalankan dalam platform Jenkins. Jenkins bertugas menjalankan proses pengujian otomatis ketika terjadi perubahan dalam kode.

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian Otomatis

Regression Test dilakukan pada 108 skenario pengujian. *Regression test* tersebut menghasilkan keberhasilan sebesar 97%. Hal ini dikarenakan terdapat bug yang belum diperbaiki sehingga aplikasi web MIHC belum berhasil memenuhi lingkup pengujian yang telah dibuat. *Report regression test* dapat dilihat pada Gambar 2.

B. Evaluasi Perubahan UI

Tahap ini berisi perubahan UI yang dilakukan selama sprint berlangsung. Perubahan-perubahan yang terjadi antara lain;(1)Penghapusan fitur bahan ajar, detasering, hak kekayaan intelektual, penelitian, pengabdian masyarakat, pengelola jurnal, tes, dan visiting scientist. Penghapusan fitur-fitur ini dilakukan karena terdapat kesalahan analisis dari supervisor;(2)Perubahan tampilan sidebar aplikasi web MIHC pada saat window dalam keadaan *maximize*;(3)Penambahan fitur hak kekayaan intelektual dan penelitian, serta penghapusan fitur karya ilmiah.

C. Evaluasi Hasil Pengujian

Dari hasil pengujian otomatis yang telah dilakukan dihitung rata-rata waktu berjalannya pengujian otomatis. Perhitungan rata-rata diambil dari data pada tanggal 2 Juli 2023 sampai 6 Juli 2023. Skenario pengujian berjalan dalam waktu yang berbeda-beda dikarenakan langkah-langkah untuk setiap skenario juga berbeda. Skenario pengujian yang memiliki banyak langkah pengujian akan berjalan lebih lama dibandingkan dengan skenario pengujian yang memiliki sedikit langkah pengujian. Lama pengujian juga dipengaruhi oleh kecepatan sistem dan koneksi yang digunakan ketika pengujian otomatis berjalan. Koneksi internet yang lambat akan memperlambat proses pengujian yang dijalankan. Hal ini dikarenakan pengujian dilakukan pada *website* yang membutuhkan koneksi internet untuk menjalankannya. Hasil perhitungan dapat dilihat pada Tabel 2.

Berdasarkan Tabel 2 didapatkan rata-rata waktu berjalannya pengujian otomatis 1 jam 7 detik. Jika dibandingkan dengan selisih waktu *build job* myits-app-updater satu dengan *build job* selanjutnya yang ada pada Gambar 3, waktu rata-rata pengujian otomatis memakan waktu lebih lama. Lamanya waktu yang dibutuhkan untuk menjalankan pengujian otomatis menyebabkan potensi tumpang tindih berjalannya pengujian otomatis akan sering terjadi. Hal ini dikarenakan pada Gambar 3 terlihat bahwa *build job* myits-app-updater dijalankan berkali-kali pada tanggal 6 juli 2023 dengan selisih waktu kurang dari satu jam, sedangkan rata-rata waktu berjalannya pengujian otomatis adalah 1 jam 7 detik yang artinya pengujian otomatis memakan waktu lebih lama sehingga saat pengujian otomatis masih berjalan, pengujian otomatis lainnya akan antri terus-menerus seperti pada Gambar 4.

Berdasarkan evaluasi yang telah dilakukan, implementasi CI/CD dengan *build trigger* berdasarkan perubahan kode oleh pengembang tidak memungkinkan jika diterapkan pada aplikasi web MIHC. Hal ini dikarenakan oleh:

1. Waktu yang dibutuhkan untuk menjalankan pengujian otomatis memakan waktu lebih lama dibandingkan selisih waktu *build job* myits-app-updater satu dengan *build job* selanjutnya sehingga potensi tumpang tindih berpotensi sering terjadi.
2. *Job* Jenkins yang bertugas untuk me-*build* aplikasi web DPTSI, yaitu myits-app-updater, tidak dipisah untuk setiap aplikasinya sehingga meskipun bukan aplikasi web MIHC yang sedang di *build*, *testing* akan tetap berjalan karena *trigger job testing* berdasarkan *job* myits-app-updater. Hal ini juga menyebabkan *deploy* otomatis tidak dapat dilakukan karena *job testing* tidak dapat membedakan aplikasi web mana yang sedang di *build*. *Job* myits-app-updater juga berguna me-*build* untuk *environment* dev dan *production* sehingga semua proses *build* aplikasi web di DPTSI hanya menggunakan *job* ini.

Dengan implementasi CI/CD yang tidak memungkinkan dengan *build trigger* berdasarkan perubahan kode oleh pengembang, dilakukan pendekatan lain, yakni *nightly build*. *Nightly build* menjalankan *job testing* secara berkala dengan menggunakan *job schedule*. Penambahan jadwal pada *job* ini dilakukan dengan memilih *Build periodically* pada *section build trigger* kemudian jadwal ditulis dalam bentuk cron. Cron yang diatur pada *job* ini yakni setiap hari pada jam 2 malam. Cron yang digunakan dapat dilihat pada Gambar 5. Cron dapat diatur dalam *input* pada kolom *schedule* sesuai dengan Gambar 5. Dengan menggunakan pendekatan *nightly build*, *developer* dan *tester* tetap dapat mengontrol kualitas aplikasi web MIHC dengan melihat *report* dari hasil pengujian otomatis yang telah dilakukan pada malam hari.

V. KESIMPULAN DAN SARAN

Kesimpulan yang didapatkan setelah melakukan seluruh proses adalah sebagai berikut:(1)Pengujian diawali dengan analisis UI aplikasi web MIHC pada modul profil dan portofolio, dilanjutkan dengan pembuatan skenario pengujian. Skenario pengujian tersebut diimplementasikan dalam bentuk kode pemrograman dengan menggunakan *framework* Serenity dan didapatkan 108 skenario pengujian. Pengujian otomatis dengan framework Serenity dapat diterapkan seacara berkepanjangan pada aplikasi web MIHC karena framework ini bersifat *open source* dan *report* dihasilkan secara otomatis setelah pengujian selesai dilakukan;(2)Integrasi pengujian otomatis dengan Jenkins untuk mencapai konsep CI/CD dilakukan dengan membuat job dan melakukan setting environment pada Jenkins untuk lingkungan pengujian. Implementasi CI/CD dengan build trigger berdasarkan perubahan kode oleh pengembang tidak memungkinkan untuk diterapkan pada aplikasi web MIHC karena job myits-app-updater melakukan build untuk semua aplikasi di DPTSI. Selain itu, build job myits-app-updater satu dan selanjutnya dijalankan dengan selisih waktu kurang dari satu jam, sedangkan rata-rata waktu berjalannya pengujian otomatis adalah 1 jam 7 detik sehingga dapat menyebabkan potensi tumpang tindih berjalannya pengujian otomatis;(3)Berdasarkan tahapan pengujian otomatis yang telah dilakukan, pengujian otomatis yang dijalankan mengalami keberhasilan 97%. Pengujian otomatis mengalami kegagalan 3% dikarenakan terdapat *bug* yang belum dilakukan perbaikan. Pengujian otomatis tidak memungkinkan untuk diintegrasikan di dalam implementasi CI/CD dengan *build trigger* berdasarkan perubahan kode oleh pengembang sehingga dilakukan pendekatan lain, yakni

nightly build untuk mengontrol kualitas aplikasi web MIHC.

Saran yang dapat penulis sampaikan adalah pemisahan *job* Jenkins untuk proses *build* aplikasi web MIHC sehingga konsep CI/CD dengan *build trigger* berdasarkan perubahan kode oleh pengembang dapat diimplementasikan.

DAFTAR PUSTAKA

[1] Rida Adila, Rizky Januar Akbar, and Hudan Studiawan, "Rancang bangun modul portofolio pegawai pada aplikasi myits human capital management dengan arsitektur event driven," *Jurnal Teknik ITS*, vol. 11, no. 1, pp. A29–A34, 2022.

[2] M. Albarka Umar and C. Zhanfang, "A study of automated software testing: automation tools and frameworks," *International Journal of Computer Science Engineering (IJCSE)*, vol. 8, no. 6, pp. 217–225, 2019.

[3] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," in *in Future of Software Engineering (FOSE'07)* , Italy: in Future of Software Engineering (FOSE'07) , 2007, pp. 85–103. doi: 10.1109/FOSE.2007.25.

[4] J. Kasurinen, O. Taipale, and K. Smolander, "Software test automation in practice: Empirical observations," *Data Structure and Software Engineering: Challenges and Improvements*, pp. 110–148, Apr. 2016, doi: 10.1155/2010/620836.

[5] X. Liu, "Research on Graphic User Interface Automation Testing Technology Based on Cloud Platform," in *Journal of Physics: Conference Series*, China: Institute of Physics Publishing, 2019. doi: 10.1088/1742-6596/1345/5/052082.

[6] A. Farid and I. Gita Anugrah, "Implementasi ci/cd pipeline pada framework androbase menggunakan jenkins (studi kasus: PT. Andromedia)," *Jurnal Nasional Komputasi dan Teknologi Informasi*, vol. 4, no. 6, 2021.