PENERAPAN *BEHAVIOR DRIVEN DEVELOPMENT* UNTUK UJI VALIDASI

Siti Nadiah Hijriyani[1], Sri Widowati[2], Dana Sulistyo Kusumo[3]
[1,2,3]Universitas Telkom, Bandung
sitinadiahh@students.telkomuniversity.ac.id[1], sriwidowati@telkomuniversity.ac.id[2], danaksumo@telkomuniversity.ac.id[3]

Abstrak

Sebelum perangkat lunak diserahkan kepada pengguna, diperlukan pengujian untuk memastikan bahwa fungsionalitas yang dibuat memenuhi kebutuhan, maka dilakukan pengujian validasi. Pada penelitian ini, proses pengujian validasi dilakukan dengan menggunakan *metode Behavior Driven Development (*BDD*)* dengan mendefinisikan *requirement* ke dalam skenario *step by step* sesuai perilaku pengguna terhadap perangkat lunak. Objek yang diuji adalah proyek sistem informasi untuk Binus School di Radya Labs. Pada proses pengujian ini dibuat *automated testing* yang terintegrasi BDD menggunakan *tools* Katalon Studio. Hasil penelitian ini, menunjukan bahwa implementasi BDD dapat diimplementasikan pada sistem informasi sekolah. Dampak yang paling terlihat adalah pendeskripsian *requirement* yang jauh lebih terarah dan terperinci.

Kata kunci : *behavior driven development* , uji validasi, katalon studio

Abstract

Before the software is handed over to users, testing is needed to ensure that the functionality created meets the needs, then validation testing is carried out. In this research, the validation testing process is carried out using the Behavior Driven Development (BDD) method by defining requirements into step-by-step scenarios according to user behavior towards software. The object tested was an information system project for Binus School at Radya Labs. In this testing process, automated testing integrated with BDD is made using Katalon Studio tools. The results of this research show that the implementation of BDD can be implemented in school information systems. The most visible impact is the description of requirements that are much more focused and detailed.

Keywords: behavior driven development , validation test, katalon studio

I. PENDAHULUAN

Pada pengembangan perangkat lunak, sebelum perangkat lunak diserah terima ke pengguna diperlukan pengujian untuk memastikan fitur yang dibuat sudah sesuai dengan kebutuhan pengguna serta memastikan perangkat lunak dapat berjalan sesuai dengan fungsionalitasnya, maka dari itu perlu dilakukan uji validasi (*validation test*). Uji validasi merupakan tahap terakhir dalam pengembangan perangkat lunak sebelum sampai ke pengguna. Tujuan uji validasi untuk memberikan jaminan akhir bahwa aplikasi yang dirancang telah memenuhi kriteria yang dibutuhkan [1].

Pengujian validasi pada sebuah perangkat lunak biasanya diilakukan secara manual dengan mengacu pada *test case* yang didefinisikan berdasarkan poin-poin *requirement* yang sudah ditentukan diawal oleh tim pengembang dan klien [2]. Pengujian ini masih memiliki kelemahan pada bagian pendefenisian *requirement* yang terkadang tidak mencakup semua kebutuhan, dikarenakan *requirement* yang sudah ditentukan berbentuk kalimat yang menggambarkan hasil yang ingin dicapai. Bentuk ini tidak menggambarkan secara menyeluruh tentang proses dan kondisi yang diinginkan, sehingga sering terjadi ketidaksesuaian pemahaman *requirement* antara tim pengembang dan klien (contohnya ketika pembuatan fitur master level yang tidak boleh ada duplikasi data). Maka, dibutuhkan sebuah pendekatan untuk membuat *requirement* perangkat lunak dapat didefinisikan secara jelas, terperinci dan dapat dimengerti oleh semua *stakeholder*. Pendekatan tersebut yaitu *Behavior Driven Development* (BDD). BDD merupakan sebuah pendekatan pengembangan perangkat lunak yang mempunyai fokus pada pendefinisian *requirement* dengan cara membuat berbagai skenario *step-by-step* dari perilaku *user*. Pembuatan skenario akan dilakukan secara terperinci dan menggunakan bahasa yang mudah dipahami oleh semua *stakeholder* [3]. Sehingga sangat cocok untuk dipakai dalam pengujian validasi karena karakteristiknya yang membuat *requirement* dalam bentuk skenario penggunaan aplikasi. Sehingga penguji dapat lebih mudah membuat *test case* yang sesuai dengan *requirement*.

Pada penelitian sebelumnya, tools yang digunakan untuk mengkombinasikan *automated testing* pada penerapan BDD menggunakan tools selenium , dimana fungsinya hanya khusus untuk *web desktop application* [8], sedangkan pada penelitian ini *tools* yang digunakan adalah katalon studio, yang bisa digunakan untuk *mobile web,* web, desktop.

Pada penelitian ini, pengujian validasi dilakukan pada proyek yang sedang dikerjakan di Radya Labs yaitu Sistem Informasi Binus School. Pengujian ini menerapkan *Behavior Driven Development* dengan mengkombinasikan *automated testing* menggunakan tools Katalon Studio. Katalon studio digunakan untuk mengeksekusi *test case* berdasarkan *file behavior driven development* yang telah dibuat.

Pada penelitian sebelumnya, penerapan BDD dalam *project software* skala besar digunakan untuk memfasilitasi praktisi uji validasi dalam membantu *improve* proses yang diusulkan [12], Menggunakan *action research* mengidentifikasi *refactoring* di BDD berdasarkan kesamaan *project* menggunakan *Normalized Compression Similarity*

(NCS) dan *Similarity Ratio* (SR) [13], Menggunakan metode *microtask crowdsourcing* untuk mengurangi biaya *onboarding* dan memungkingkan para *developer* untuk berkontribusi dengan cepat [14], dan menggunakan tiga fase yang berbeda untuk menganalisis BDD *Framework* yang digunakan di *project open source* yang menggunakan lima bahasa pemrograman yang berbeda [15].

Tujuan penelitian ini adalah penerapan *Behavior Driven Development* pada tahap uji validasi di proyek pembuatan sistem informasi Binus School. Selain itu akan dilakukan juga pembuatan *automated testing* yang mengacu pada skenario BDD dengan menggunakan Katalon Studio.

II. KAJIAN TEORI

A. *Behavior Driven Development* (BDD)

Behavior Driven Development (BDD) merupakan metode agile yang mendukung kolaborasi antara pemangku kepentingan bisnis, *developer* dan semua pihak yang terlibat dalam projek. Fokus utama BDD adalah menuliskan skenario yang menjelaskan *behavior* (kebiasaan) sistem dari perspektif pengguna dan mendorong kolaborasi antar stakeholder [4, 5]. Keunggulan BDD adalah menyediakan bahasa umum berdasarkan kalimat sederhana dan terstruktur yang memfasilitasi komunikasi antara *stakeholder*. Hal ini tentunya akan mengurangi kesalahpahaman dan menghilangkan kode yang tidak perlu dan pemborosan fungsional [5] . Dengan menerapkan BDD berarti semua fitur memiliki skenario pengujian yang dapat memastikan alur kerja perangkat lunak sesuai dengan kebutuhan pengguna [5].

Untuk menemukan titik awal komunikasi antara klien dengan tim projek dalam pengumpulan *requirement*, maka dibutuhkan identifikasi kebutuhan dan fitur yang akan didahulukan untuk dikerjakan. Sebuah fitur akan direalisasikan menjadi *user story*. *I*nterkasi antara pengguna dan sistem digambarkan melalui *User Story*. *Template* yang digunakan untuk membuat *user story* dapat dilihat pada gambar 1.

As a <type of user>, I want <some goals> so that <some reasons>

GAMBAR 1
TEMPLATE USER STORY

Meskipun *user story* memberikan kemudahan dalam memahami fitur, tetapi masih memiliki kendala dalam penerapan. Penggunaan *user story* dan ruang lingkup yang dijelaskan tidak terlalu jelas menyebabkan kerap terjadinya ambiguitas. Untuk mengatasi hal ini, *user story* dapat dirinci menggunakan serangkaian skenario BDD yang menggambarkan *acceptance criteria* [4]. Skenario menjelaskan bagaimana sistem harus berperilaku ketika dalam keadaan tertentu. Skenario ini menggunakan bahasa *Domain Specific Language* (DSL) atau yang dikenal dengan *Gherkin* [6]. *Gherkin language* adalah bahasa khusus yang membantu perilaku bisnis tanpa perlu menjelaskan implementasi secara detail. Di dalam *Gherkin*, ada tiga *keyword* penting yaitu *Given* (Initial state sebelum skenario berjalan), *When* (kondisi yang dilakukan pengguna untuk menjalankan sistem), *Then* (hasil respon dari sistem). Istilah penting di Gherkin, yaitu:

Feature: mendeskripsikan fitur aplikasi, setiap file hanya boleh memiliki satu fitur

1. Skenario: contoh yang mengambarkan aturan bisnis
2. Step: mendefinisikan proses menggunakan *keyword Given, When, Then, And, But*
3. *Background*: digunakan untuk memahami beberapa langkah yang diberikan
4. *Other keyword*: mendefinisikan dengan penambahan *keyword* bisa berupa *tags* (@) atau *comments* (#)

```
@tag
Feature: Login

#Scenario 1
  Scenario: User cannot login because not registered
    Given user at login page
    When user fill unregistered username
    And user fill unregistered password
    And user click login button
    Then application show message User not registered
```

GAMBAR 2
ILUSTRASI *BEHAVIOR DRIVEN DEVELOPMENT*

B. Pengujian Validasi

Pengujian validasi dilakukan untuk menjamin kesesuaian perangkat lunak dengan desain yang ditetapkan sebelumnya. Indikator keberhasilan pengujian validasi adalah jika fungsi yang ada pada aplikasi sesuai dengan yang diharapkan pengguna [7]. Pengujian validasi merupakan pengujian terakhir sebelum aplikasi digunakan oleh pengguna.

C. Automated Testing

Automated testing digunakan untuk menjalankan pengujian dengan membandingkan hasil aktual dan hasil yang diharapkan. *automated testing* memiliki keuntungan yang utama yaitu *efficiency* (waktu untuk mengeksekusi *test script* lebih cepat dibanding pengujian manual karena dijalankan oleh *tools*), *repeatability* (pengujian *test script* yang sama dapat dieksekusi dengan cara yang sebelumnya tanpa mengulangnya lagi), *scope of testing* (menjadi sangat mudah untuk mengetahui jumlah *code* yang telah dicakup oleh *test case*), dan *decrease of expenses* (menggunakan *automated* mengurangi pekerjaan yang dilakukan *manual testing*) [8].

Automated testing dapat mencakup permasalahan yang tidak dapat dilakukan oleh *manual testing*. Karena *manual testing* memiliki permasalahan diantaranya terlalu banyak mengkonsumsi waktu, tidak bisa digunakan kembali, usaha yang dikeluarkan lebih besar dan terkadang beberapa kesalahan yang tidak terlihat [6].

D. Katalon Studio

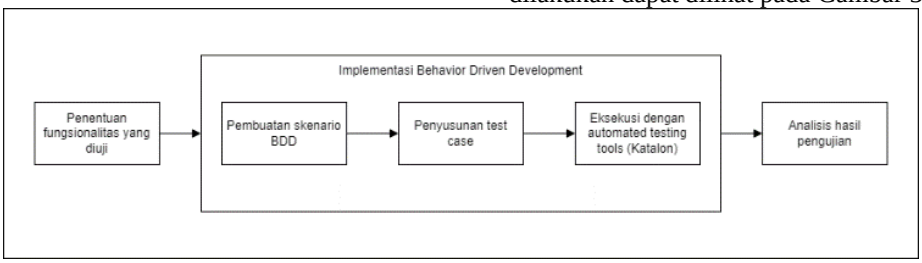
Katalon studio merupakan aplikasi *open source* yang digunakan untuk *automated testing*. Aplikasi ini dikembangkan oleh Katalon LLC. Katalon studio dapat digunakan untuk pengujian *desktop, website, mobile*

applications dan API. Bahasa yang dapat digunakan Katalon yaitu *groovy, java, javascript* dan lain-lain. Katalon ini dapat dieksekusi di berbagai *browser* (*chrome, mozila firefox*), IOS dan android [9].

Katalon memiliki kelebihan yaitu mendukung pembuatan BDD yang terintegrasi dengan *cucumber*. Pengguna semakin mudah untuk membuat BDD karena sudah tersedia template pembuatan BDD ketika membuat *feature file*. Sehingga, akan mengurangi waktu dan usaha pegguna dalam membuat format BDD yang benar [10].

III. METODE

Pengerjaan penelitian ini akan mengacu pada metode *Behavior Driven Development* (BDD). Kontribusi penerapan BDD terletak pada tahap pembuatan skenario BDD, penyusunan test case dan eksekusi. Tahap penelitian yang dilakukan dapat dilihat pada Gambar 3.



GAMBAR 3
TAHAP PENELITIAN

A. Penentuan Fungsionalitas Yang Diuji

Penelitian ini dilakukan pada proyek yang sedang dikerjakan Radya Labs yaitu Sistem Informasi Binus School. Seperti sistem informasi pada umumnya, sistem informasi ini adalah sistem informasi manajemen sekolah berbasis *website* dan *mobile app* sebagai solusi administrasi terpadu untuk sekolah yang memiliki beberapa modul terintegrasi yang dapat diakses oleh semua anggota sekolah (guru, wali kelas, pegawai sekolah dan siswa) dan orang tua/wali siswa [11] .

Pada penelitian ini, objek yang dipakai adalah modul *subject document*. Modul ini memiliki banyak fitur yang dapat diakses oleh aktor tertentu dan setiap fitur yang ada pada modul ini memiliki hubungan dengan fitur yang lain. Ada 9 *functional requirement* yang diuji pada penelitian ini. Penjelasan dari setiap fungsionalitas dan aktor yang terlibat dapat ditunjukan pada tabel 1.

TABEL 1
FUNCTIONAL REQUIREMENT

No.	Fitur	Penjelasan	User yang terlibat
1.	*Teacher can filter list document*	*User* yang sudah *login* sebagai guru dapat memfilter daftar dokumen berdasarkan kriteria tertentu	*Teacher*
2.	*Teacher can see list document*	*User* yang sudah *login* sebagai guru dapat melihat daftar dokumen yang telah dibuat	
3.	*Teacher can fill out documents with draft status*	*User* yang sudah *login* sebagai guru dapat mengisi dokumen yang berstatus *draft* sehingga dokumen dapat direview oleh koordinator mata pelajaran	
4.	*Teacher can edit documents with need approval (SH) status*	*User* yang sudah *login* sebagai guru dapat mengedit dokumen selama status dokumen tersebut masih need approval (SH)	
5.	*Teacher can see the history of approval document*	*User* yang sudah *login* sebagai guru dapat melihat riwayat persetujuan dokumen	
6.	*Subject Head can see a list of documents with need approval (SH) status*	*User* yang sudah *login* sebagai koordinator mata pelajaran dapat melihat daftar dokumen yang berstatus *need approval* (SH)	*Subject head*
7.	*Subject Head can approve documents with need approval (SH) status*	*User* yang sudah *login* sebagai koordinator mata pelajaran dapat menyetujui dokumen yang berstatus membutuhkan persetujuan	
8.	*Subject Head can change document with need approval (SH) status to need revision*	*User* yang sudah *login* sebagai koordinator mata pelajaran dapat memberi revisi dokumen sehingga guru dapat mengupdate kembali isi dokumen	
9.	*Head of department can approve documents with need approval (HOD) status*	*User* yang sudah *login* sebagai kepala departemen dapat menyetujui dokumen yang berstatus membutuhkan persetujuan	*Head of department*

Pada modul *subject document* terdapat tiga aktor yang mempunyai hak akses yang berbeda untuk setiap fitur, yaitu *teacher*, *subject head*, dan *head of department*. *Teacher* atau guru di modul ini dapat memasukan dokumen pelajaran untuk ditinjau oleh *subject head*. *Subject head* atau koordinator mata pelajaran dapat menyetujui dan merevisi dokumen pelajaran dengan tingkatan *approval level 1* yang diajukan oleh guru. Sedangkan *head of department* memiliki hak akses yang sama dengan *subject head*, ditambah dengan dapat menyetujui dan merevisi dokumen pelajaran dengan tingkatan *approval level 2*.

B. Pembuatan Skenario BDD

Setelah ditentukan fitur apa saja yang diuji pada tabel 1. Selanjutnya proses penentuan skenario yang diawali dengan pembuatan *user story* berdasarkan perilaku pengguna dan fungsionalitas. Kemudian menentukan skenario berdasarkan setiap kemungkinan *input* dan *output* yang ada pada *user story*. Penjelasan dari setiap fitur beserta *user story* dan skenarionya dapat dilihat pada tabel 2

TABEL 2
SKENARIO PENGUJIAN

No.	***Fitur***	***User Story***	***Skenario***
1.	*Teacher can filter list document*	*As a teacher, I can filter the list of documents by academic year, level, grade, term, subject, approval type so that I can see the list according to the filter*	*Teacher can filter by academic year*
			Teacher can filter by academic year and level
			Teacher can filter by academic year, level and grade
			Teacher can filter by academic year, level, grade and term
			Teacher can filter by academic year, level, grade, term and subject
			Teacher can filter by academic year, level, grade, term, subject and approval type
			Teacher can't filter by level
			Teacher can't filter by grade
			Teacher can't filter by term
			Teacher can't filter by subject
			Teacher can't filter by approval type
2	*Teacher can see list document*	*As a teacher, I can see the list of documents so I can fill in the document*	*Teacher can see list document*
			Teacher can see an empty table if the document list does not exist
3	*Teacher can fill out documents with draft status*	*As a teacher, I can fill out documents with draft status so that the document can be reviewed by the subject head and head of department*	*Teacher can open document page*
			Teacher does not fill in one of the fields in the document
			Teacher cancels filling out the document
			Teacher can fill in all the fields in the document
4	*Teacher can edit documents with need approval (SH)*	*As a teacher, I can edit documents that need approval (SH)*	*Teacher can edit documents*
			Teacher cancels edit document
5	*Teacher can see the history of approval document*	*As a teacher, I can see the history of approval documents*	*Teacher can see the history of approval documents*
			Teacher can see detailed history
6	*Subject Head can see a list of documents with need approval (SH) status*	*As a subject head, I can see a list of documents that need approval so that the document can be approved*	*Subject head can see list of document status need approval (SH)*
7	*Subject Head can approve documents with need approval (SH) status*	*As a subject head, I can approve documents with the status of need approval (SH)*	*Subject head can see document detail page with status need approval*
			Subject head can cancel approve document
			Subject head can approve document
8	*Subject Head can change document with need approval (SH) status to need revision*	*As a subject head, I can change the status of the document to need revision so that the teacher can revise the document*	*Subject head see pop up document need revision*
			Subject head can cancel filling in revision notes
			Subject head can give revision
9	*Head of department can approve documents with need approval (HOD) status*	*As a head of department, I can approve documents that need approval status so that the document is approved*	*Head of department can see document detail page with status of need approval (HOD)*
			Head of department can cancel approve document
			Head of department can approve document

C. Penyusunan *Test Case*

Hasil dari skenario yang sudah dibuat pada tabel 2 akan menjadi panduan untuk pembuatan dokumentasi *test case*. *Test* case diperlukan untuk memastikan bahwa setiap fitur yang dijalankan sudah sesuai dengan kebutuhan. Sebuah test case yang baik mampu mencakup keseluruhan kasus uji dan

memenuhi persyaratan perangkat lunak, tidak membuat test case yang berulang, dan test case harus menghasilkan hasil yang sama setiap kali pengujian dengan tidak memerhatikan siapa pengujinya.

Test case dibagi menjadi beberapa komponen, seperti *test case id, description, pre-step, step dan expected result*, dapat dilihat pada tabel 3. Pada umumnya format test case terdapat test data, tetapi pada pengujian ini tidak menggunakannya karena pengujian berupa *behavior* (perilaku) bukan data *input*. Penulisan *pre-step*, *step* dan *expected result* menggunakan bahasa gherkin dengan *keyword* utama yaitu *GIVEN* untuk menjelaskan konteks dari skenario masukan. *WHEN* untuk menjelaskan proses yang dilakukan oleh pengguna. *THEN* untuk menjelaskan hasil proses yang sudah dilakukan oleh pengguna.

TABEL 3
TEST CASE CONTOH FITUR *TEACHER CAN FILTER LIST DOCUMENT*

Test case ID	***Description***	***Pre-Step***	***Step***	***Expected Result***
Scenario 1	*Teacher can filter by academic year*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter academic year dropdown* *And teacher click apply button*	*Then application displays a list based on the academic year filter*
Scenario 2	*Teacher can filter by academic year and level*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter academic year dropdown* *And teacher select filter level dropdown* *And teacher click apply button*	*Then application displays a list based on the academic year and level filter*
Scenario 3	*Teacher can filter by academic year, level and grade*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter academic year dropdown* *And teacher select filter level dropdown* *And teacher select filter grade dropdown* *And teacher click apply button*	*Then application displays a list based on the academic year, level and grade filter*
Scenario 4	*Teacher can filter by academic year, level, grade and term*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter academic year dropdown* *And teacher select filter level dropdown* *And teacher select filter grade dropdown* *And teacher select filter term dropdown* *And teacher click apply button*	*Then application displays a list based on the academic year, level, grade and term filter*
Scenario 5	*Teacher can filter by academic year, level, grade, term and subject*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter academic year dropdown* *And teacher select filter level dropdown* *And teacher select filter grade dropdown* *And teacher select filter term dropdown* *And teacher select filter subject dropdown* *And teacher click apply button*	*Then application displays a list based on the academic year, level, grade, term and subject filter*
Scenario 6	*Teacher can filter by academic year, level, grade, term, subject and approval type*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter academic year dropdown* *And teacher select filter level dropdown* *And teacher select filter grade dropdown* *And teacher select filter term dropdown* *And teacher select filter subject dropdown* *And teacher select filter approval type dropdown* *And teacher click apply button*	*Then application displays a list based on the academic year, level, grade, term, subject and approval type filter*
Scenario 7	*Teacher can't filter by level*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter level*	*Then application display field level disabled*
Scenario 8	*Teacher can't filter by grade*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter grade*	*Then application display field level disabled*
Scenario 9	*Teacher can't filter by term*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter term*	*Then application display field level disabled*

Test case ID	*Description*	*Pre-Step*	*Step*	*Expected Result*
Scenario 10	*Teacher can't filter by subject*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter subject*	*Then application display field level disabled*
Scenario 11	*Teacher can't filter by approval type*	*Given teacher is logged in*	*When teacher select document menu in sidebar* *And teacher select subject document menu* *And teacher select filter approval type*	*Then application display field level disabled*

D. Eksekusi dengan *Automated Testing*

Tahap pertama *automated testing* adalah membuat *feature file*. *Test case* yang telah dibuat sebelumnya pada tabel 3 dipindahkan ke dalam *feature file*. Sebuah *feature file* terdiri dari satu atau lebih skenario yang menjelaskan apa yang seharusnya dilakukan oleh sistem dalam situasi tertentu. Penulisan test case pada *feature file* dapat dilihat pada gambar 4.

```
Feature: Teacher can filter list document
USER STORY: As a teacher, I can filter the list of documents by academic year, level, grade,
term, subject, approval type so that I can see the list according to the filter

  #Scenario 1
  Scenario: Teacher can filter by academic year
    Given teacher is logged in
    When teacher select document menu in sidebar
    And teacher select subject document menu
    And teacher select filter academic year dropdown
    And teacher click apply button

  #Scenario 2
  Scenario: Teacher can filter by academic year and level
    Given teacher is logged in
    When teacher select document menu in sidebar
    And teacher select subject document menu
    And teacher select filter academic year dropdown
    And teacher select filter level dropdown
    And teacher click apply button
    Then application displays a list based on the academic year and level filter
```

GAMBAR 4
FEATURE FILE TEACHER CAN FILTER LIST DOCUMENT

Setelah selesai membuat *feature file*, tahap selanjutnya adalah membuat *test code* dengan bahasa *groovy* yang berisi *function-function* dari tiap *line* di skenario BDD. Di dalam setiap *function* ini akan berisi *code step testing* yang berasal dari *script* hasil perekaman objek. Perekaman objek ini menggunakan teknik *record web*, *spy web* dan penambahan *test object* secara manual. *Record web* dilakukan dengan cara merekam setiap aksi berdasarkan skenario dan menghasilkan sebuah *script* objek. Sedangkan *spy web* digunakan untuk mengambil objek yang tidak terdeteksi ketika melakukan *record web*. Pada kasus objek yang sulit untuk diambil dikarenakan objek yang bergerak, maka dilakukan penambahan objek secara manual. Penambahan objek secara manual bisa menggunakan *xpath* atau dengan memasukan atribut dari *tag* yang akan diambil. Selanjutnya *script* objek akan dimasukan ke dalam setiap *function*. Berikut contoh *test code* yang sudah terintegerasi dengan BDD dilihat pada Gambar 5.

```
class TeacherCanFilterListDocument {
    /**
     * The step definitions below match with Katalon sample Gherkin steps
     */

    @Given("teacher is logged in")
    def teacher_is_logged_in(){
        WebUI.openBrowser('')
        WebUI.maximizeWindow()
        WebUI.navigateToUrl('https://bss-webclient.azurewebsites.net/auth/login?ReturnUrl=%2F')
        WebUI.setText(findTestObject('Object Repository/input_username'), 'teacher1')
        WebUI.setEncryptedText(findTestObject('Object Repository/input_password'), 'OyqPNYkJe4LEt+jNUgtSAg==')
        WebUI.click(findTestObject('Object Repository/button_LOGIN'))
    }

    @When("teacher select document menu in sidebar")
    def teacher_select_document_menu_in_sidebar(){
        WebUI.scrollToElement(findTestObject('sidebar_document menu'), 0)
        WebUI.click(findTestObject('Object Repository/sidebar_document menu'))
    }

    @And("teacher select subject document menu")
    def teacher_select_subject_document_menu(){
        WebUI.click(findTestObject('Object Repository/subject document menu'))
    }

    @And("teacher select filter academic year dropdown")
    def teacher_select_filter_academic_year_dropdown() {
        WebUI.click(findTestObject('Object Repository/Page_Binus School/filter_Academic Year'))

        WebUI.click(findTestObject('Object Repository/Page_Binus School/tahun_2021-2022'))
    }
```

GAMBAR 5
SCRIPT HASIL PEREKAMAN PENGUJIAN

Selain melakukan perekaman objek, pada beberapa skenario dibutuhkan logika perulangan, dimana ketika skenario yang membutuhkan pengecekan status dokumen di dalam tabel daftar dokumen. Misal skenario "*Subject head can see document detail page with status need approval*" disini *subject head* bisa mengakses halaman detail dokumen jika berstatus *need approval.*

Jika *test code* sudah dilengkapi, maka skenario BDD dapat dijalankan. Tahap pengujian akan diulang jika terdapat perubahan pada aplikasi, baik karena adanya *error* dan *bug*, maupun karena adanya perubahan *requirement* dari *stakeholder*. Untuk *test case* yang sudah dibuat rekaman *automate tes*t-nya, maka pengujian dilakukan dengan menggunakan rekaman yang sudah ada.

IV. HASIL DAN PEMBAHASAN

Bagian ini berisi tentang bagaimana hasil dari *automate testing* yang dibuat berdasarkan skenario BDD pada bab sebelumnya beserta analisis dari hasil pengujian.

A. Hasil Pengujian

Hasil pengujian akan dibuat otomatis menggunakan katalon. Format file dari hasil pengujian berupa file HTML yang berisi informasi tentang keberhasilan dan kegagalan dari setiap skenario di fitur yang diuji. Selain HTML, format hasil pengujian dapat disesuaikan dengan format lain sesuai kebutuhan. Hasil skenario pengujian yang berhasil akan berwarna hijau dan hasil skenario yang gagal akan berwarna merah.

1. Hasil pengujian yang berhasil (*Passed*)

Sebagian besar fitur yang diuji berstatus PASSED yang berarti berhasil. Ini menandakan sistem yang dibangun berhasil menampilkan *output* sesuai dengan skenario yang telah dibuat sebelumnya. *Report* yang dibuat berisi informasi tentang fitur beserta skenario pengujian yang dijalankan. Penjelasan untuk hasil dari setiap pengujian fitur yang berhasil dapat dilihat pada daftar di bawah.

GAMBAR 6
CONTOH *REPORT TEST*

a. Teacher can filter list document : Aplikasi berhasil menampilkan tabel yang berisi *subject document* sesuai dengan nilai *parameter filt*er seperti *year*, *level*, *grade*, *term*, *subject* dan *approval type* yang telah diatur sebelumnya.

b. Teacher can fill out documents with draft status : Aplikasi berhasil mengubah status *document* dari *draft* menjadi *need approval* pada document yang diisi dengan lengkap, dan tetap berstatus *draft* pada document yang diisi dengan tidak lengkap atau dibatalkan.

c. Teacher can edit documents with need approval (SH) : Aplikasi berhasil menampilkan pesan sukses ketika aktor mengubah *text* pada *form document* dan menenekan *button submit*.

d. Teacher can see the history of approval document : Aplikasi berhasil menampilkan riwayat status document dari status *draft* sampai *approved*.

e. Subject Head can see a list of documents with need approval (SH) status : Aplikasi berhasil menampilkan tabel yang document dengan status *need approval (SH)*.

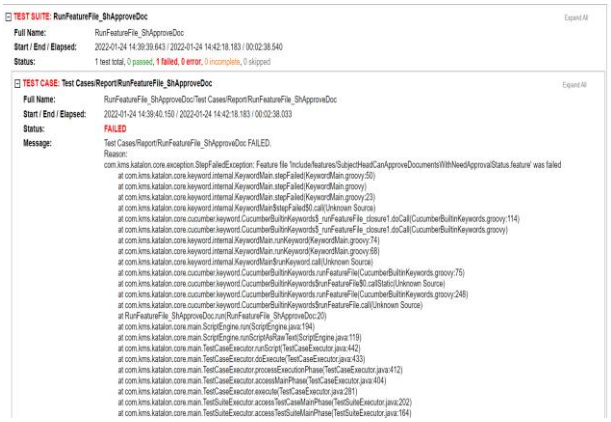
f. Subject Head can change document with need approval (SH) status to need revision : Aplikasi berhasil menampilkan informasi tentang *subject document* yang dipilih, mengubah status *document* menjadi *need revision* apabila aktor menekan *button revision* dan mengisi catatan perbaikan.

g. Head of department can approve documents with need approval (HOD) status : Aplikasi berhasil menampilkan informasi *document*, mengubah status *document* menjadi *approved* dan kembali ke halaman detail dokumen jika aktor menekan *button cancel*.

2. Hasil pengujian yang gagal (*Failed*)

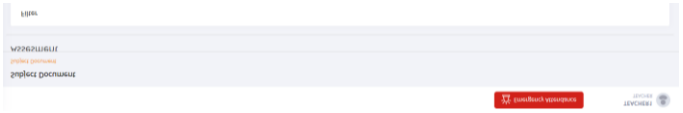
Pada beberapa skenario, sistem gagal menampilkan *output* yang sudah ditentukan di skenario pengujian. Hal ini bisa disebabkan oleh kesalahan pengiriman data dari *server*, logika pada sisi *front-end* dan pengaruh revisi pada fitur lain. Dari sisi teknis, aplikasi gagal menampilkan suatu element HTML dengan nilai *value* atau *text* yang diharapkan, sehingga Katalon tidak bisa menemukan element yang sudah ditentukan. Skenario yang memiliki *bug* akan dilaporkan kepada tim terkait untuk segera dilakukan *debug* atau revisi. Penjelasan untuk skenario yang mengalami kegagalan dapat dilihat pada daftar di bawah.

a. *Subject head can see document detail page with status need approval*: Aplikasi gagal menampilkan informasi nama, alamat, *upload*, jenis kelamin, hobi dan *example*. *Error* ini disebabkan perubahan format *key* yang dikirim melalui *REST API* dari *server*. *Error* ini diperbaiki dengan cara menyamakan *key* pada *function consume* API di sisi *front-end*. Tampilan halaman yang mengalami *error* ini dapat dilihat pada gambar 6.



GAMBAR 7
CONTOH KE-1 SKENARIO GAGAL

b. *Teacher can see list document* : Aplikasi gagal menampilkan list dokumen untuk guru. *Error* ini disebabkan pada sisi *front-end* di mana *link* API yang dimasukan salah, sehingga tidak ada respon dari *server* yang dikirim. *Error* ini diperbaiki dengan mengganti *link* api pada sisi *front-end*.



GAMBAR 8
CONTOH KE-2 SKENARIO GAGAL

B. Analisis Hasil Pengujian

Dari sembilan fitur yang dijadikan objek pengujian, terdapat dua fitur yang mengalami kegagalan. Ini berarti sekitar 92% atau 23 skenario berhasil dijalankan tanpa mengalami *error*, detail dari hasil pengujian dapat dilihat pada tabel 4. Skenario yang gagal akan langsung perbaiki oleh pihak terkait.

TABEL 4
HASIL PENGUJIAN

No.	Fitur	Jumlah Skenario	Status Sebelum *Fixing*		Status Sesudah *Fixing*	
			Passed	***Failed***	***Passed***	***Failed***
1	*Teacher can filter list document*	11	11	0	11	0
2	*Teacher can see list document*	2	1	1	2	0
3	*Teacher can fill out documents with draft status*	4	4	0	4	0
4	*Teacher can edit documents with need approval (SH)*	2	2	0	2	0
5	*Teacher can see the history of approval document*	2	2	0	2	0
6	*Subject Head can see a list of documents with need approval (SH) status*	1	1	0	1	0
7	*Subject Head can approve documents with need approval (SH) status*	3	2	1	3	0
8	*Subject Head can change document with need approval (SH) status to need revision*	3	3	0	3	0
9	*Head of department can approve documents with need approval (HOD) status*	3	3	0	3	0
	Total	31	23	2	31	0

Pada tahap pengambilan *test object*, terjadi kesulitan dalam mengambil *test object* yang hanya muncul sementara seperti *toast*. Hal ini dikarenakan ketika *object toast* menghilang maka *tag toast* pada HTML pun akan hilang, sehingga Katalon tidak bisa mengambil *data tag* tersebut. Masalah ini dapat diatasi dengan menambahkan *object test* secara *manual* dengan memanfaatkan alat *inspect element* yang ada di *browser* untuk mendapatkan informasi *tag* dari atributnya.

Untuk pengecekan perubahan status pada dokumen, dibuat sebuah *function* yang akan melakukan perulangan untuk mencari *row* dari tabel daftar dokumen, yang memiliki status sama dengan status awal di skenario dan menyimpan *index* dari *row* tersebut. Lalu sesudah pengisian dokumen, akan dilakukan pengecekan status pada *row* yang sudah tersimpan sebelumnya, apakah sama dengan status akhir pada skenario. Pembuatan *function* ini efektif untuk skenario yang diharus kan mengecek perubahan element di banyak dokumen yang sama.

Secara keseluruhan, implementasi *behavior driven development* pada *automate testing* di modul *subject document* dari proyek Binus School dapat disimpulkan berhasil. Karena semua skenario BDD yang ada bisa dijadikan *automated testing* dengan menggunakan *tools* Katalon.

V. KESIMPULAN

Berdasarkan penelitian yang sudah dilakukan dapat disimpulkan bahwa *Behavior Driven Development* (BDD) dapat diimplementasikan pada modul *subject document* proyek pembuatan sistem informasi sekolah dengan cara merumuskan BDD ditahap pembuatan skenario berdasarkan *user story* sampai dengan pengujian. *Penggunaan automated testing* yang terintegrasi dengan BDD dapat dilakukan dengan menggunakan *tools* Katalon. Berhasil mengimplementasikan 9 fitur dengan 25 skenario BDD. Terdapat 2 skenario *failed* dan 23 *passed*. Diperlukan logika perulangan untuk mengecek perubahan status di daftar dokumen.

REFERENSI

[1] M. Nurudin, W. Jayanti, R. Dwi Saputro and M. Priadyan Saputra, "Pengujian *Black Box* pada Aplikasi Penjualan Berbasis Web Menggunakan Teknik *Boundary Value Analysis*", Jurnal Informatika Universitas Pamulang, vol. 4, p. 145, 2019. doi: 10.32493/informatika.v4i4.3841 [Accessed 29 November 2020].

[2] G. Setiawan, "Pengujian Perangkat Lunak Menggunakan Metode *Black Box* Studi Kasus Exelsa Universitas Sanata Dharma", 2011. [Accessed 1 September 2021].

[3] U. Nugraha, S. Nurduha Robaiah and D. Rospinoedji, *"Testing The Information System Software Using Behavior Driven Development Method"*, Journal Of Archaeology Of Egypt/Egyptology, 2021.

[4] C. Solis and X. Wang, *"A Study of the Characteristics of Behaviour Driven Development"*, 2011 37th EUROMICRO *Conference on Software Engineering and Advanced Applications, 2011*. doi: 10.1109/seaa.2011.76 [Accessed 25 November 2020].

[5] I. Raharjana, F. Harris and A. Justitia, *"Tool for Generating Behavior-Driven Development Test-Cases", Journal of Information Systems Engineering and Business Intelligence*, vol. 6, no. 1, p. 27, 2020. doi: 10.20473/jisebi.6.1.27-36.

[6] M. Kaur and R. Kumari, *"Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro", International Journal of Computer Applications*, vol. 24, no. 1, pp. 1-7, 2011. doi: 10.5120/2918-3844

[7] H. Sulistyanto and A. SN, "Urgensi Pengujian Pada Kemajemukan Perangkat Lunak Dalam Multi Perspektif", Komuniti: Jurnal Komunikasi dan Teknologi Informasi, vol. 6, no. 1, 2014. doi: 10.23917/komuniti.v6i1.2944 [Accessed 29 November 2020].

[8] E. Vila, G. Novakova and D. Todorova, *"Automation Testing Framework for Web Applications with Selenium WebDriver", Proceedings of the International Conference on Advances in Image Processing*, 2017. doi: 10.1145/3133264.3133300 [Accessed 30 August 2021].

[9] S. Tjandra, I. Maryati and J. Theopilus, *Automated Software Testing For Multi Platformapplications Using Katalon Studio*, vol. 20, no. 1, 2021. doi: https://doi.org/10.33508/wt.v20i1.3114 [Accessed 30 August 2021].

[10] "BDD *Testing Framework* (*Cucumber integration*)", https://docs.katalon.com, 2021. [Online]. *Available*: https://docs.katalon.com/katalon-studio/docs/cucumber-features-file.html#add-feature-files. [Accessed: 31- Aug- 2021].

[11] 2019. [*Online*]. *Available*: https://simsekolah.com/. [*Accessed*: 01- Aug- 2021]

[12] M. Irshad, R. Britto, K. Petersen, "*Adapting Behavior Driven Development (BDD) for large-scale software systems*". *The Journal of Systems & Software*, vol. 177, 2021. doi: https://doi.org/10.1016/j.jss.2021.110944.

[13] M. Irshad, J. Börstler, K. Petersen, "*Supporting refactoring of BDD specifications—An empirical study". Information and Software Technology*, vol. 141, 2022. doi: https://doi.org/10.1016/j.infsof.2021.106717.

[14] E. Aghayi, T.D. LaToza, P. Surendra, S. Abolghasemi, "*Crowdsourced Behavior-Driven Development*". *Journal of Systems and Software*, vol. 171, 2021. doi: https://doi.org/10.1016/j.jss.2020.110840.

[15] F. Zampetti, A.D. Sorbo. C. A. Vissagio, G. Canfora, M. D. Penta, "*Demystifying the adoption of behavior-driven development in open source projects*". *Information and Software Technology*, vol. 123, 2020. doi: https://doi.org/10.1016/j.infsof.2020.106311.