Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 7, Juli 2019, hlm. 7127-7135 http://j-ptiik.ub.ac.id

Pengembangan Aplikasi Pemindaian Kode Pengujian Unit (Studi Kasus: PT Global Digital Niaga)

Ade Wija Nugraha[1], Bayu Priyambadha[2], Arief Andy Soebroto[2]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]adewija@student.ub.ac.id, [2]bayu_priyambadha@ub.ac.id, [2]ariefas@ub.ac.id

Abstrak

Pengujian unit merupakan salah satu fase pada pengujian perangkat lunak, tetapi masih terdapat beberapa *programmer* yang enggan melakukan fase pengujian unit. Permasalahan tersebut juga terjadi pada perusahaan PT. Global Digital Niaga. Masih terdapat beberapa *programmer* pada perusahaan yang tidak memperhatikan kualitas dari kode pengujian unit yang dibuat dan hanya memperhatikan nilai dari cakupan kode pengujian unitnya saja. Maka dari itu, peneliti melakukan pengembangan aplikasi yang dapat digunakan untuk melakukan pemindaian kode pengujian unit dan dapat menampilkan kesalahan penulisan berdasarkan aturan yang telah dibuat oleh perusahaan. Proses pemindaian dimulai dengan mencari informasi dari proyek yang diunggah. Kemudian, aplikasi memindai kode produksi dan kode pengujian unit dengan menggunakan *library JavaParser*. Setelah itu, aplikasi menganalisa setiap kode pengujian unit yang dipindai apakah kode tersebut sesuai dengan aturan penulisan atau tidak. Kemudian, aplikasi menyimpan hasil pemindaian dan menampilkannya. Tahap pertama dari penelitian ini yaitu studi literatur. Kemudian, tahap rekayasa kebutuhan yang menghasilkan 25 kebutuhan fungsional dan 2 kebutuhan non-fungsional. Selanjutnya tahap perancangan dan diikuti tahap implementasi dengan membuat aplikasi berbasis *website*. Setelah itu, tahap pengujian dengan nilai 100% *pass*, 100% valid, dan 100% akurat untuk pengujian unit, validasi, dan akurasi. Aplikasi juga dapat dijalankan pada delapan *browser* berbeda untuk pengujian kompatibilitas.

Kata kunci: *pengujian unit, java, perangkat lunak, aplikasi website, pemindaian kode program*

Abstract

Unit testing is one phase in the software testing process, but there are still some programmers who are reluctant to do the unit testing. These problems also occur in PT. Global Digital Niaga. Some programmers don't pay attention to the quality of the unit testing's code and only pay attention to the value of the unit testing code coverage. Researchers develop applications that can be used to scan unit testing codes and identify the incorrect unit testing code based on rules. The scanning process starts with searching for information on the uploaded project. Then, application scans production code and unit testing code using JavaParser library. After that, the application analyzes each scanned unit test code whether the code matches with writing rules or not. Then, the application saves the scan results and displays them. The first phase of this research is the study of literature. Then, the requirements engineering produced 25 functional and 2 non-functional. Next is the design and followed by the implementation by creating a website application. After that, the testing with a 100% pass, 100% valid, and 100% accurate for unit, validation, and accuracy testing. The application can run on eight different browsers for compatibility testing.

Keywords: *unit testing, java, software, website application, scanning code*

1. PENDAHULUAN

Pada sebuah proses *software development* terdapat tahapan pengujian terhadap perangkat lunak yang dikembangkan. Tahapan pengujian dari perangkat lunak yang telah dikembangkan merupakan sebuah tahapan yang penting dimana pengembang dapat memastikan aplikasi yang dikembangkan telah sesuai dengan kebutuhan yang telah didefinisikan sebelumnya.

Pada tahap pengujian dari perangkat lunak yang dikembangan terdapat sebuah fase yang bernama fase pengujian unit, fase tersebut digunakan untuk memastikan setiap unit telah dibuat dengan baik dan dapat berjalan sesuai dengan fungsinya masing – masing. Fase pengujian unit dianggap sebagai sebuah fase yang penting namun meskipun demikian, masih ada pengembang perangkat lunak yang tidak suka dalam membuat kode pengujian unit. Terdapat sebuah survei kepada 171 pengembang dari dua puluh sembilan negara dan survei tersebut menyatakan bahwa proses pengujian unit merupakan sebuah proses yang memiliki peran penting dalam proses pengembangan perangkat lunak. Meskipun demikian, pengembang perangkat lunak tersebut menyatakan bahwa proses menulis kode pengujian unit merupakan sebuah proses yang menghabiskan banyak waktu dan dua belas persen dari pengembang perangkat lunak tersebut menyatakan enggan untuk melakukan fase pengujian unit. Pengembang perangkat lunak tersebut juga menyatakan bahwa penulisan kode pengujian unit dilakukan hanya untuk mengetahui hasil cakupan kode dari pengujian unit yang sudah dibuat, tanpa memikirkan cara untuk membuat kode pengujian unit dengan baik dan benar (Daka & Fraser, 2014).

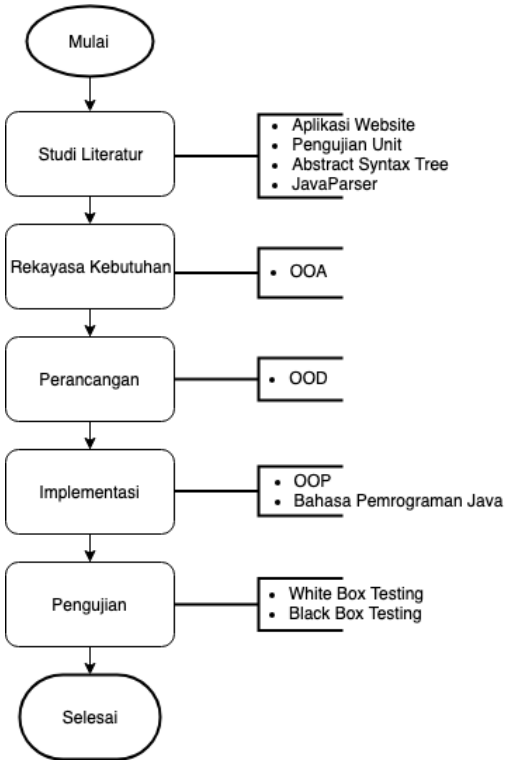
Permasalahan yang terdapat pada penelitian yang dilakukan oleh Daka dan Fraser pada tahun 2014 tersebut juga terjadi pada PT. Global Digital Niaga. Pada perusahaan tersebut, masih terdapat beberapa pengembang perangkat lunak pada perusahaan tersebut yang tidak memperhatikan kualitas dari kode pengujian unit yang dibuat dan hanya memperhatikan nilai dari cakupan kode pengujian unitnya saja. Maka dari itu pada penelitian ini, peneliti melakukan pengembangan dari aplikasi yang dapat digunakan untuk melakukan pemindaian kode pengujian unit dan dapat melakukan identifikasi terhadap kode pengujian unit yang benar atau tidak berdasarkan aturan penulisan kode pengujian unit yang telah dibuat oleh PT Global Digital Niaga. Dengan menggunakan aplikasi tersebut, nantinya dapat diketahui kode pengujian unit mana yang tidak sesuai dengan aturan – aturan penulisan kode pengujian unit dan melalui penelitian ini diharapkan dapat membantu dalam meningkatkan kualitas dari kode pengujian unit yang dibuat oleh pengembang perangkat lunak pada PT Global Digital Niaga.

Terdapat beberapa tahapan penelitian yang harus dilakukan oleh peneliti pada penelitian ini. Tahapan – tahapan tersebut dimulai dengan studi literatur, tahap kedua dari penelitian ini yaitu rekayasa kebutuhan, setelah itu terdapat tahap ketiga yaitu perancangan, diikuti dengan tahap keempat yang merupakan implementasi, lalu tahap kelima berupa pengujian, dan tahap yang terakhir yaitu penarikan kesimpulan.

2. METODOLOGI

Metodologi penelitian yang digunakan oleh peneliti menggunakan prinsip dari SDLC yang bernama *Waterfall*. Hal tersebut dikarenakan pada model *waterfall*, setiap fase dilakukan secara berurutan dari atas ke bawah. Hasil yang diperoleh dari masing – masing fase akan digunakan sebagai dasar dari pengerjaan fase berikutnya (Sommerville, 2011).

Peneliti melakukan beberapa proses dalam melakukan penelitian ini. Proses tersebut terdiri dari lima tahapan. Gambar 1 merupakan penggambaran dari lima tahapan tersebut.



Gambar 1. Tahapan Metodologi Penelitian

Tahap pertama yaitu studi literatur. Pada tahap ini, proses pencarian referensi - referensi dari berbagai literatur yang dapat membantu proses penelitian pengembangan aplikasi pemindaian kode pengujian unit dilakukan oleh peneliti guna memudahkan proses penelitian.

Literatur yang digunakan oleh peneliti dapat diperoleh dari dokumentasi resmi, buku, buku elektronik, *proceeding*, maupun jurnal yang dapat dicari melalui internet.

Tahap rekayasa kebutuhan merupakan tahap kedua yang dilakukan oleh peneliti. Proses rekayasa dari kebutuhan aplikasi dilakukan oleh peneliti dengan menggunakan pendekatan berorientasi objek atau OOA. Peneliti melakukan analisa kebutuhan fungsional beserta kebutuhan non-fungsional yang merupakan representasi kemampuan dari aplikasi pemindaian kode pengujian unit. Selain itu, pada tahap rekayasa kebutuhan dari penelitian pengembangan aplikasi pemindaian kode pengujian unit peneliti juga melakukan proses analisa aktor dan melakukan proses pemodelan *use case* dari aplikasi pemindaian kode pengujian unit pada sebuah *use case diagram*. *Use case* yang telah dimodelkan tersebut dijelaskan lebih detail terkait proses interaksi antara aktor dengan sistem pada *use case scenario*.

Tahap ketiga dari pengembangan aplikasi pemindaian kode pengujian unit yaitu proses perancangan. Proses perancangan yang dilakukan oleh peneliti menggunakan pendekatan berorientasi objek atau OOD. Peneliti membuat perancangan arsitektur yang terdiri dari pemodelan *sequence diagram* dan pemodelan *class diagram*, selain itu terdapat perancangan data, perancangan algoritme dari *method* yang akan diimplementasi, dan yang yang terakhir terdapat perancangan dari antarmuka pengguna yang akan diimplementasikan.

Tahap keempat yaitu proses implementasi. Proses implementasi dari pengembangan aplikasi pemindaian kode pengujian unit dilakukan oleh peneliti dengan membuat realisasi dari proses perancangan yang telah dilakukan. Perancangan tersebut dituangkan pada sebuah kode program dari aplikasi pemindaian kode pengujian unit yang akan dikembangkan. Aplikasi yang akan dikembangkan berbasis *website* dan dikembangkan dengan menggunakan bahasa pemrograman yang berorientasi objek atau OOP yaitu bahasa java. Peneliti memanfaatkan *library JavaParser* yang menggunakan konsep *abstract syntax tree* dalam melakukan pengembangan aplikasi pemindaian kode pengujian unit. *Abstract Syntax Tree* digunakan oleh peneliti karena dapat merepresentasikan suatu *source code* atau kode program pada sebuah pohon atau *tree* (Smith, et al., 2018). Hal tersebut membuat peneliti menjadi lebih mudah dalam melakukan pengambilan data maupun manipulasi struktur dasar dari kode program tersebut. Pada tahap implementasi, peneliti membuat implementasi algoritme, implementasi data, dan implementasi antarmuka pengguna.

Proses pengujian merupakan tahap kelima yang dilakukan oleh peneliti. Pada proses pengujian, peneliti melakukan pengecekan pada unit maupun kebutuhan fungsional dan kebutuhan non-fungsional dari proses pengembangan aplikasi pemindaian kode pengujian unit.

Peneliti melakukan validasi terhadap setiap modul atau unit apakah modul atau unit tersebut dapat berjalan dengan baik dan sesuai dengan fungsinya atau tidak melalui proses pengujian unit (Hunt & Thomas, 2003). Peneliti menggunakan metode yang bernama *whitebox testing* untuk melakukan proses pengujian pada pengujian unit. Selain melakukan pengujian unit, peneliti juga melakukan pengujian terhadap kebutuhan fungsional maupun kebutuhan non-fungsional melalui pengujian validasi. Peneliti dapat menguji keseluruhan fungsi yang didapat pada proses rekayasa kebutuhan dan dapat memastikan bahwa apakah aplikasi dapat dijalankan dengan baik sesuai dengan harapan pengguna atau tidak (Pressman, 2010). Peneliti menggunakan metode yang bernama *blackbox testing* untuk melakukan pengujian validasi.

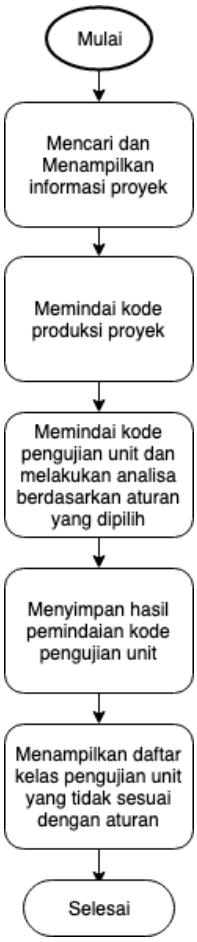
3. HASIL DAN PEMBAHASAN

3.1. Rekayasa Kebutuhan

Proses rekayasa kebutuhan digunakan untuk mendeskripsikan aplikasi yang akan dikembangkan, melakukan analisa aktor, maupun melakukan analisa kebutuhan yang ada dalam proses pengembangan aplikasi pemindaian kode pengujian unit.

Aplikasi yang dikembangkan pada penelitian ini bernama Unit Test Scanner. Aplikasi tersebut berbasis *website*. Hal tersebut mengungkinkan aplikasi Unit Test Scanner dapat diakses menggunakan berbagai aplikasi peramban web (Al-Fedaghi, 2011). Aplikasi Unit Test Scanner dapat dikategorikan sebagai kakas bantu yang dapat digunakan untuk mendeteksi kesalahan penulisan kode pada pengujian unit yang ditulis menggunakan

bahasa pemrograman java. Kesalahan penulisan kode tersebut ditentukan berdasarkan aturan penulisan kode pengujian unit yang telah ditetapkan oleh PT Global Digital Niaga.



Gambar 2. Proses Pemindaian Kode Pengujian Unit dengan Aplikasi Unit Test Scanner

Aplikasi Unit Test Scanner memiliki beberapa proses dalam melakukan pemindaian kode pengujian unit seperti pada Gambar 2. Pertama – tama aplikasi akan mulai melakukan pencarian terdahap informasi dari proyek yang diunggah dan informasi proyek tersebut akan ditampilkan pada *team leader*. Kemudian, aplikasi akan melakukan pemindaian terhadap kode produksi dari proyek dengan menggunakan *library JavaParser*. Setelah itu, akan dilakukan proses pemindaian pada kode pengujian unit dengan menggunakan *library JavaParser* dan aplikasi akan menganalisa setiap kode yang dipindai apakah kode tersebut sesuai dengan aturan penulisan kode pengujian unit atau tidak. Setelah itu, aplikasi akan menyimpan data hasil pemindaian kode pengujian unit dan aplikasi akan menampilkan kode yang tidak sesuai dengan aturan yang dipilih beserta keterangan kenapa kode tersebut dianggap tidak sesuai dengan aturan tersebut.

Terdapat dua aktor yang berinteraksi dengan aplikasi Unit Test Scanner yaitu *user* dan *team leader*. *User* memiliki lima *use case* antara lain *register*, menampilkan daftar kelas pengujian unit yang tidak sesuai aturan untuk umum, memperbarui kata sandi, menampilkan detail hasil pemindaian untuk umum, dan *login*. Sedangkan *team leader* memiliki dua puluh *use case* antara lain menampilkan informasi proyek Java, menyimpan unggahan proyek Java, menampilkan detail hasil pemindaian, menampilkan daftar proyek yang telah dipindai, memindai kode pengujian unit, mengirim hasil pemindaian, menampilkan daftar kelas pengujian unit yang tidak sesuai aturan, menampilkan daftar *rule*, memperbarui *rule group*, menambah *rule group*, menghapus *rule group*, memperbarui *rule*, menambah *rule*, menghapus *rule*, memperbarui profil, memindai kode produksi proyek, menyimpan hasil pemindaian kode pengujian unit, memindai kode pengujian unit berdasarkan aturan, menampilkan hasil pemindaian kode pengujian unit, dan *logout*.

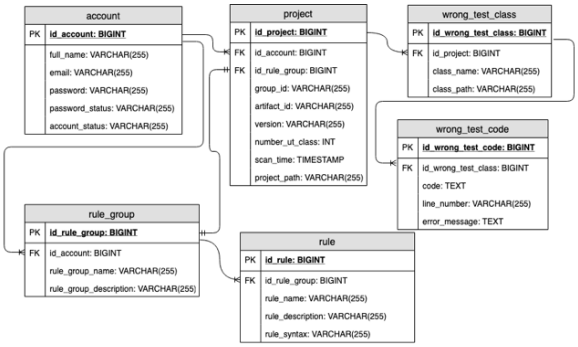
Selain itu, pengembangan aplikasi Unit Test Scanner juga memiliki dua kebutuhan non-fungsional yaitu kompatibilitas dan akurasi. Kebutuhan non-fungsional kompatibilitas yaitu kemampuan sistem yang dapat dijalankan pada peramban web *Safari*, *Google Chrome*, dan *Mozilla Firefox*. Sedangkan kebutuhan non-fungsional akurasi yaitu kemampuan sistem yang dapat melakukan pemindaian terhadap kode pengujian unit berdasarkan aturan yang telah ditetapkan oleh PT Global Digital Niaga dengan nilai akurasi minimal 90%. Gambar 3 dibawah adalah *use case diagram* milik aplikasi Unit Test Scanner.

Pada proses rekayasa kebutuhan, terdapat pemodelan *use case diagram*. Pemodelan *use case diagram* tersebut dilakukan oleh peneliti supaya peneliti dapat mengetahui interaksi antara sistem dengan pengguna dari sistem (Whitten & Bentley, 2007). *Use case diagram* terdiri dari aktor, *use case*, serta relasi yang menghubungkan antara aktor dengan masing - masing *use case*. Masing – masing *use case* yang ada pada *use case diagram* dijelaskan lebih rinci mengenai langkah – langkah atau skenario yang dijalankan oleh sistem ketika pengguna menjalankan *use case* tersebut pada bagian *use case scenario* (Dooley, 2011). Pemodelan *use case diagram* pada aplikasi Unit Test Scanner ditunjukkan pada Gambar 3.

Gambar 3. *Use Case Diagram* dari Aplikasi Unit Test Scanner

3.2. Perancangan dan Implementasi

Pada proses perancangan menghasilkan perancangan data, perancangan arsitektur sistem, perancangan algoritme sistem, serta perancangan antarmuka pengguna. Pada perancangan data menghasilkan sebuah *physical data model* atau PDM. PDM tersebut merupakan pemodelan *database* yang kemudian digunakan dalam aplikasi Unit Test Scanner. *Database* yang akan diimplementasikan dalam aplikasi Unit Test Scanner memiliki enam tabel yang terdiri dari tabel *account*, tabel *project*, tabel *wrong_test_class*, tabel *wrong_test_code*, tabel *rule_group*, dan tabel *rule*. Pada Gambar 4 merupakan gambaran dari rancangan *database* yang akan digunakan pada proses pengembangan aplikasi Unit Test Scanner.



Gambar 4. *Physical Data Model* Aplikasi Unit Test Scanner

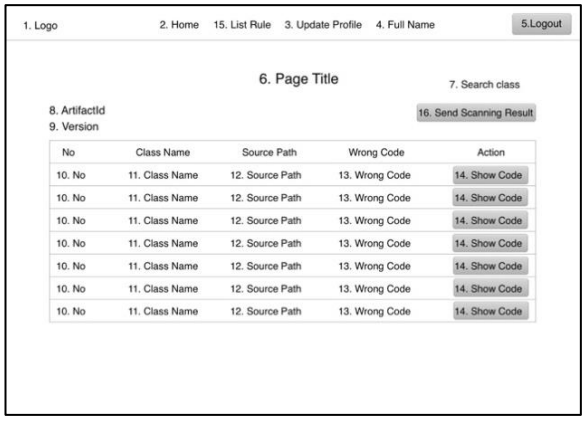
Hasil dari perancangan arsitektur berupa pemodelan dari *sequence diagram* serta *class diagram*. Pemodelan *sequence diagram* dirancang berdasarkan tiga *use case scenario* yang telah dibuat pada bagian sebelumnya. *Use case scenario* yang akan digunakan dalam untuk merancang pemodelan *sequence diagram* diambil dari *use case* menyimpan unggahan proyek java, menampilkan informasi proyek java, dan memindai kode pengujian unit. Dengan melakukan pemodelan *sequence diagram*, peneliti dapat mengetahui gambaran mengenai interaksi antar objek melalui proses pertukaran pesan pada setiap objek (Rumbaugh, et al., 1998).

Pada pemodelan *class diagram* terdiri dari kelas java yang memiliki relasi antara satu kelas dengan kelas lainnya dan didalam pemodelan *class diagram* tersebut juga terdapat nama method serta nama variabel yang ada pada masing – masing kelas. Struktur objek dalam sistem yang dikembangkan dapat diketahui dengan adanya pemodelan *class diagram* ini (Whitten & Bentley, 2007). Terdapat 36 kelas java yang dibuat dalam proses pengembangan aplikasi Unit Test Scanner. Kelas – kelas tersebut dibagi dalam 7 package berdasarkan fungsinya. Package yang digunakan dalam perancangan tersebut antara lain *controllers*, *dtos*, *entities*, *helper*, *pojos*, *repositories*, *services*. Selain membuat 36 kelas java, pada proses pengembangan aplikasi Unit Test Scanner juga menggunakan beberapa kelas bawaan dari java dan *library* yang digunakan. Kelas – kelas tersebut seperti *MultipartFile*, *HttpSession*, *Model*, *File*, *FileInputStream*, *BufferedOutputStream*, *Node*, *NodeList*, *VoidVisitorAdapter*, dan lain sebagainya.

Pada perancangan algoritme menghasilkan rancangan *pseudocode* dari lima method yang nantinya akan diimplementasikan pada aplikasi Unit Test Scanner. Method – method tersebut

antara lain terdiri dari *method collect project information*, *method check main pom file*, *method update rule group*, *method find scan result file created*, dan *method scan test code by rules*.

Pada perancangan antarmuka pengguna menghasilkan rancangan dari tampilan antarmuka atau *user interface* yang akan digunakan pada proses pengembangan aplikasi Unit Test Scanner. Rancangan antarmuka pengguna tersebut akan dibuat dalam betuk *wireframe*.



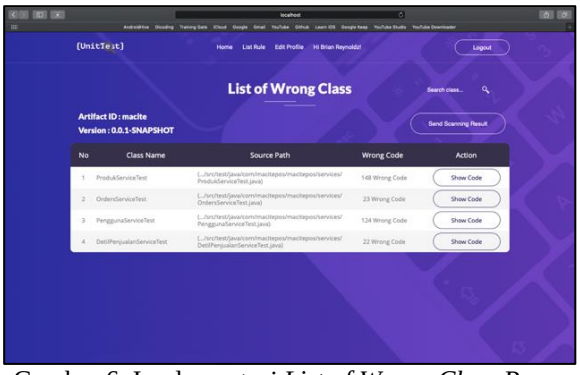
Gambar 5. Rancangan Antarmuka *List of Wrong Class Page*

Hasil perancangan antarmuka pengguna *list of wrong class page* digambarkan dalam Gambar 5. Halaman tersebut digunakan untuk menampilkan kelas pengujian unit yang didalamnya terdapat kode pengujian unit yang tidak sesuai dengan aturan penulisan yang ditetapkan oleh PT Global Digital Niaga. Perancangan *user interface* tersebut digunakan sebagai landasan dalam tahap implementasi *user interface*.

Proses implementasi memberikan hasil berupa implementasi kode program, implementasi data, serta implementasi antarmuka pengguna. Tahap ini mengubah rancangan algoritme pada tahap sebelumnya menjadi kode program. Kode program yang dihasilkan dari penelitian ini dibuat menggunakan bahasa pemrograman java. Pemilihan bahasa pemrograman java untuk implementasi ini dikarenakan bahasa tersebut merupakan salah satu bahasa pemrograman yang menerapkan konsep object oriented (Wu, 2006). Implementasi kode program yang dihasilkan pada penelitian ini antara lain yaitu implementasi *method collect project information*, *method check main pom file*, *method update rule group*, *method find scan result file created*, dan *method scan test code by rules*.

Pada implementasi data, menghasilkan kode yang digunakan untuk melakukan pembuatan *database* dari aplikasi pemindaian kode pengujian unit. Tabel tersebut terdiri dari tabel *account*, tabel *project*, tabel *wrong_test_class*, tabel *wrong_test_code*, tabel *rule_group*, dan tabel *rule.* Tabel – tabel tersebut diperoleh pada proses perancangan data.

Pada implementasi antarmuka pengguna menghasilkan implementasi dari setiap halaman yang ada dalam aplikasi Unit Test Scanner. Halaman – halaman tersebut merupakan hasil implementasi dari perancangan antarmuka pengguna yang telah dibuat. Pada aplikasi Unit Test Scanner terdapat 21 halaman antarmuka pengguna yang diimplementasikan. Halaman – halaman tersebut terdiri dari *login page, register page, register success page, send email in update password page, update password page, home page, upload project page, scanning preparation page, project information page, list of project page, list of wrong class page, list of wrong code page, update profile page, update profile success page, list of rule page, add rule group page, update rule group page, add rule page, update rule page, list of wrong class public page*, dan *list of wrong code public page*. Salah satu hasil implementasi antarmuka dalam sistem ini digambarkan ke dalam Gambar 6 di bawah ini. Dimana gambar tersebut merupakan implementasi *list of wrong class page*.



Gambar 6. Implementasi *List of Wrong Class Page*

3.3. Pengujian

Penelitian ini menggunakan dua tahap pengujian, tahap pertama menggunakan pengujian unit kemudian dilanjutkan dengan pengujian validasi. Metode pengujian kotak putih (*white-box testing)* digunakan untuk

pelaksanaan pengujian unit. Melalui penggunaan metode *white-box testing*, peneliti dapat membuat kasus uji yang mampu memastikan bahwa seluruh statement yang ada pada program dapat dijalankan setidaknya satu kali, menjalankan keputusan logika pada saat kondisi benar maupun saat kondisi salah, menjalankan seluruh proses perulangan dengan batas – batas operasional yang sudah ditentukan, serta menggunakan struktur data internal untuk memastikan valid atau tidaknya program tersebut (Pressman, 2010). Setiap kasus uji yang dihasilkan pada saat penerapan metode *whitebox testing* ini merupakan sebuah jalur *independent* atau *independent path*.

Peneliti melakukan pengujian pada setiap *independent path* yang dihasilkan dari masing – masing *pseudocode*. *Pseudocode* yang diuji merupakan *pseudocode* dari lima perancangan algoritme antara lain *method check main pom file*, *method collect project information*, *method find scan result file created*, *method update rule group*, dan *method scan test code by rules*. Pada proses pengujian unit yang dilakukan, peneliti memperoleh hasil *pass* pada setiap kasus uji dari masing – masing *method*. Berdasarkan hasil pengujian pada lima *method* tersebut dapat disimpulkan bahwa hasil dari pengujian unit memiliki nilai 100% *pass*. Tabel 1 pada makalah ini merangkum hasil dari proses pengujian unit dari aplikasi Unit Test Scanner.

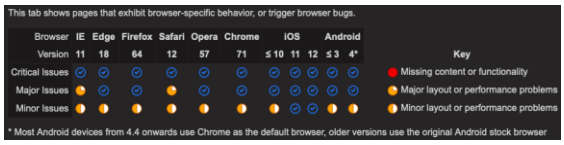
Tabel 1. Hasil Pengujian Unit Aplikasi Unit Test Scanner

Nama Method	Cyclomatic Complexity	Jumlah Kasus Uji	Hasil
checkMain PomFile	4	4	*Pass*
collect Project Information	4	4	*Pass*
findScan ResultFile Created	3	3	*Pass*
updateRule Group	4	4	*Pass*
scanTest CodeBy Rules	9	9	*Pass*

Pengujian validasi pada aplikasi *unit test scanner* ini menerapkan metode *black-box testing*. Melalui penggunaan metode tersebut, peneliti dapat mengetahui ada atau tidaknya fungsionalitas yang salah ataupun hilang, ada atau tidaknya kesalahan pada antarmuka, ada atau tidaknya kesalahan dalam struktur data maupun akses pada basis data ekternal, ada atau tidaknya kesalahan pada prilaku sistem atau kinerja sistem, serta ada atau tidaknya kesalahan pada proses inisialisasi atau terminasi dari sistem tersebut (Pressman, 2010).

Peneliti melakukan pengujian validasi pada dua puluh satu kebutuhan fungsional. Kebutuhan fungsional yang diuji antara lain *login*, *register*, memperbarui kata sandi, menampilkan detail hasil pemindaian untuk umum, menampilkan daftar kelas pengujian unit yang tidak sesuai aturan untuk umum, menampilkan informasi proyek java, menyimpan unggahan proyek java, menampilkan daftar kelas pengujian unit yang tidak sesuai aturan, memindai kode pengujian unit, menampilkan detail hasil pemindaian, mengirim hasil pemindaian, menampilkan proyek yang telah dipindai, menampilkan daftar *rule*, memperbarui *rule group*, menambah *rule group*, menghapus *rule group*, memperbarui *rule*, menambah *rule*, menghapus *rule*, memperbarui profil, dan *logout*. Pada setiap jalur utama maupun jalur alternatif fungsional yang telah diuji, peneliti mendapatkan nilai valid. Pengujian validasi terhadap kebutuhan fungsional dari aplikasi Unit Test Scanner menghasilkan nilai 100% valid.

Pada pengujian validasi, selain melakukan pengujian untuk kebutuhan fungsional, peneliti juga melakukan pengujian untuk kebutuhan non-fungsional yaitu kompatibilitas dan akurasi. Pada pengujian kompatibilitas, peneliti menggunakan kakas bantu bernama *SortSite*. Hasil dari pengujian kompatibilitas digambarkan dalam Gambar 7.



Gambar 7. Pengujian Kompatibilitas dari Aplikasi Unit Test Scanner

Dari Gambar 7, dapat ditarik kesimpulan bahwa aplikasi Unit Test Scanner yang dikembangkan masih terdapat beberapa *minor* dan *major issue* akan tetapi aplikasi tersebut masih dapat berjalan dengan baik tanpa adanya indikasi kehilangan konten maupun fungsionalitas pada delapan peramban *web*

yang berbeda yaitu *Opera, Firefox, IE, Safari, Chrome, Edge, browser Android,* dan *browser iOS*.

Pada pengujian akurasi, peneliti melakukan pengujian dengan cara membandingkan dua proses pemindaian. Kasus uji pertama, peneliti melakukan pemindaian pada kode pengujian unit yang belum diperbaiki. Sedangkan untuk kasus uji kedua, peneliti melakukan pemindaian kode pengujian unit kode pengujian unit yang telah diperbaiki dan disesuaikan dengan aturan penulisan kode pengujian unit dari PT Global Digital Niaga. Kode pengujian unit yang diuji berasal dari tiga proyek java yang berbeda antara lain yaitu proyek *Macite*, proyek *Jasmine Maven Plugin*, dan proyek *Index Scanner Maven Plugin*. Hasil pengujian akurasi pada kasus uji pertama menghasilkan nilai valid dan hasil pengujian akurasi pada kasus uji kedua juga menghasilkan nilai valid. Kedua nilai valid dari dua kasus uji tersebut menunjukkan bahwa hasil dari pengujian akurasi memiliki nilai 100%.

4. KESIMPULAN DAN SARAN

4.1. Kesimpulan

Berdasarkan penelitian yang berjudul pengembangan aplikasi pemindaian kode pengujian unit, peneliti dapat memberikan beberapa kesimpulan yaitu pada tahap rekayasa kebutuhan aplikasi pemindaian kode pengujian unit yang telah dilakukan, menghasilkan dua puluh lima kebutuhan fungsional. Selain kebutuhan fungsional, dari proses rekayasa kebutuhan juga dihasilkan dua kebutuhan non-fungsional. Terdapat dua aktor dari aplikasi pemindaian kode pengujian unit yaitu *team leader* dan *user*. Selain itu terdapat dua puluh lima *use case*. Keseluruhan use case tersebut telah dimodelkan pada sebuah *use case diagram* yang dapat menunjukkan interaksi antara *team leader* dan *user* dengan aplikasi pemindaian kode pengujian unit. Masing – masing *use case* telah dijabarkan dengan lebih detail dengan menggunakan *use case scenario*.

Pada tahap perancangkan dari aplikasi pemindaian kode pengujian unit menghasilkan perancangan dari beberapa komponen antara lain *physical data model* merupakan hasil yang didapatkan dari proses perancangan data, pemodelan dari tiga *sequence diagram* dan pemodelan dari satu *class diagram* yang merupakan hasil yang diperoleh dari proses perancangan arsitektur, lima *pseudocode* berdasarkan lima method yang merupakan hasil dari proses perancangan algoritme, serta dua puluh satu *wireframe* yang diperoleh dari proses perancangann antarmuka pengguna.

Pada tahap implementasi aplikasi pemindaian kode pengujian unit yang telah dilakukan, menghasilkan batasan dari aplikasi, penjelasan mengenai lingkungan pengembangan yang digunakan pada saat proses pengembangan aplikasi pemindaian kode pengujian unit, implementasi dari *pseudocode* menjadi kode program, implementasi dari enam tabel yang ada pada proses perancangan data, dan implementasi dari dua puluh satu *wireframe* yang dibuat pada proses perancangan antarmuka pengguna.

Pada proses pengujian dari aplikasi pemindaian kode pengujian unit yang telah dilakukan, menghasilkan pengujian unit yang menghasilkan nilai *pass* dari lima *method* yang diuji, pengujian validasi yang menapatkan nilai 100% valid dari keseluruhan kasus uji dari 21 kebutuhan fungsional, menghasilkan nilai 100% dari proses pengujian akurasi pemindaian kode pengujian unit berdasarkan aturan yang digunakan oleh PT Global Digital Niaga, dan pengujian kompatibilitas yang menunjukkan bahwa aplikasi pemindaian kode pengujian unit dapat dijalankan pada delapan peramban web yang berbeda.

4.2. Saran

Berdasarkan hasil penelitian dari pengembangan aplikasi pemindaian kode pengujian unit, peneliti memberikan beberapa rekomendasi untuk pengembangan aplikasi pemindaian kode pengujian unit selanjutnya. Rekomendasi tersebut antara lain yaitu diharapkan aplikasi yang dikembangkan dapat melakukan pemindaian kode sumber pengujian unit yang ditulis dengan menggunakan bahasa pemrograman yang lain selain bahasa java, ditulis menggunakan *project management framework* yang lain selain *Maven*, dan ditulis dengan menggunakan bantuan library yang lain selain *Mockito* dan *JUnit*.

5. DAFTAR PUSTAKA

Al-Fedaghi, S., 2011. Developing Web Applications. *International Journal of Software Engineering and its Applications,* 5(2), pp. 57-68.

Daka, E. & Fraser, G., 2014. *A Survey on Unit*

Testing Practices and Problems. s.l., s.n., pp. 201-210.

Dooley, J., 2011. *Software Development and Professional Practice*. s.l.:Apress.

Hunt, A. & Thomas, D., 2003. *Pragmatic Unit Testing in Java with JUnit.* s.l.:The Pragmatic Programmers, LLC.

Pressman, R. S., 2010. *Software Engineering : a practitioner's approach*. New York: McGraw-Hill.

Rumbaugh, J., Jacobson, I. & Booch, G., 1998. *The Unified Modeling Language Reference Manual*. Canada: Addison Wesley Longman, Inc.

Smith, N., Tomassetti, F. & Bruggen, D. v., 2018. *JavaParser: Visited.* s.l.:Leanpub.

Sommerville, I., 2011. *Software Engineering Ninth Edition*. s.l.:s.n.

Whitten, J. L. & Bentley, L. D., 2007. *System analysis & Design Methods*. New York: McGraw-Hill Irwin.

Wu, C. T., 2006. *An Introduction to Object-Oriented Programming with Java Fifth Edition*. New York: Higher Education.