BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Perubahan cara komunikasi dan akses informasi telah diubah oleh teknologi informasi. Tujuan penulisan ini adalah untuk meninjau penelitian terdahulu dalam bidang ini guna memperkuat landasan teoritis penelitian yang dilakukan. Diharapkan dengan memahami kontribusi penelitian sebelumnya, penelitian ini dapat memberikan kontribusi yang lebih berarti dalam pemecahan permasalahan yang ada dan pengembangan pemahaman kita terhadap subjek yang diteliti.

Tabel 2. 1 Penelitian terdahulu

Penelitian 1	
Judul Penelitian	*A Study of Automated Software Testing: Automation Tools and Frameworks*
Nama Peneliti	Mubarak Albarka Umar, Chen Zhanfang
Nama Jurnal	*International Journal of Computer Science Engineering (IJCSE)* Vol 8(6)
Tahun	2019
Hasil Penelitian	Penelitian ini menjelaskan tentang penggunan alat untuk melakukan pengujian secara otomatis yang dikenal sebagai automation testing. Pengujian perangkat lunak merupakan kegiatan yang repetitive maka dari itu automation testing hadir sebagai solusi untuk membuat pengujian perangkat lunak menjadi lebih efisien. Penelitian ini menyajikan studi komperhensif dari beberapa alat yang digunakan dalam melakukan automation testing. Hasil dari penelitian ini berupa perbandingan dari beberapa alat pengujian otomatis yang sering digunakan pada pengujian perangkat lunak [16].
Adopsi Penelitian	Penggunaan alat pengujian otomatis dalam proses pengujian sebuah perangkat lunak merupakan sebuah aspek yang terus berkembang seiring waktu. Automation testing tool merupakan aspek yang akan di terapkan pada penelitian ini sebagai lanjutan dari penelitian sebelumnya yang juga menerapkan pengujian otomatis untuk melakukan proses pengujian sebuah software.
Penelitian 2	
Judul Penelitian	*Experiences and Practices in GUI Functional Testing: A Software Practitioners' View*
Nama Peneliti	Junior, Nailton; Costa, Heitor; Karita, Leila; Machado, Ivan; Soares, Larissa
Nama Jurnal	USA: Association for Computing Machinery
Tahun	2021
Hasil Penelitian	Penelitian ini menunjukkan bahwa pengujian fungsional GUI merupakan bagian yang penting dalam proses pengembangan

	perangkat lunak. Penelitian ini menyarankan bahwa pengembangan perangkat lunak harus menggunakan pengujian fungsional GUI sebagai bagian dari proses pengembangan perangkat lunak. Pengujian fungsional GUI merupakan bagian yang penting dalam proses pengujian perangkat lunak, dan harus dilakukan secara sistematis dan teratur [17].
Adopsi Penelitian	Penelitian ini akan mengadopsi aspek pengujian fungsional yang digunakan juga pada penelitian sebelumnya. Pengujian fungsional merupakan sebuah aspek yang krusial dalam proses pengembangan perangkat lunak karena bertujuan untuk memastikan perangkat lunak telah berjalan dengan baik sesuai fungsi yang diharapkan.
Penelitian 3	
Judul Penelitian	Implementasi Aplikasi Manajemen Mes (AMM) Berbasis Web
Nama Peneliti	Cahyani, Rahma Dwi; Utomo, Hendrik Setyo
Nama Jurnal	Ultima InfoSys : Jurnal Ilmu Sistem Informasi Vol 12(1)
Tahun	2021
Hasil Penelitian	Dalam penelitian ini menjelaskan tentang implementasi sebuah sistem informasi manajemen berbasis website pada perusahaan PT. PPA yang digunakan untuk melakukan akomodasi pendataan secara realtime oleh ketua kelompok. Dalam tahap pengembangan ssistem informasi terdapat proses pengujian fungsional yang dilakukan dengan metode pengujian black box testing yang disertakan juga dengan user acceptance testing untuk memastikan sistem informasi yang telah dikembangkan berfungsi sesuai denga napa yang diharapkan [18].
Adopsi Penelitian	Penelitian ini akan menggunakan metode pengujian yang sama seperti penelitin tersebut yaitu metode *black box testing* untuk melakukan pengujian pada *software* annotation tool dari perusahan PT. Fata Organa Solusi. Metode black box testing yang digunakan dalam penelitian ini akan kombinasikan dengan beberapa teknik pengujian untuk dapat memperoleh test case yang dapat mencakup semua aspek penujian.
Penelitian 4	
Judul Penelitian	Pengujian Black Box Testing Pada Aplikasi Edu Digital Berbasis Website Menggunakan Metode Equivalence Dan Boundary Value
Nama Peneliti	Dika Pratama, Stevanu; Noviarsyah Dadaprawira, M
Nama Jurnal	Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD Vol 6 (2) – SINTA 5
Tahun	2023
Hasil Penelitian	Hasil dari penelitian ini menunjukkan bahwa penggunaan *equivalence partitioning* pada aplikasi edu digital lebih dominan untuk menguji pada fitur entri data, fitur berupa option, serta berupa tombol. *Boundary value analysis* lebih dominan dalam menguji fitur yang memiliki *range* seperti rekomendasi anggaran. Penelitian ini menunjukkan bahwa pengujian *black box testing* dengan metode *equivalence partitioning* dan *boundary value analysis* dapat digunakan untuk menguji aplikasi edu digital berbasis website, dengan fokus pada langkah pengujiannya [19].
Adopsi Penelitian	Teknik pengujian perangka lunak *equivalence class partitioning* yang membagi nilai menjadi beberapa kelas ekuivalen yang setara berdasarkan permasalahan dan juga teknik pengujian *boundary value analysis* yang dapat menemukan celah di area perbatasan pada fitur *input* dari sebuah perangkat lunak akan menjadi adopsi yang diterapkan dalam penelitian ini untuk merancang *test case*

	yang dapat mencakup kemungkinan sebanyak mungkin.
Penelitian 5	
Judul Penelitian	*Gravitational Search Algorithm Based Strategy for Combinatorial T-Way Test Suite Generation*
Nama Peneliti	Maung Htay, Khin; Razif Othman, Rozmie; Amir, Amiza; Mohammed Hachim Alkanaani, Jalal
Nama Jurnal	Journal of King Saud University - Computer and Information Sciences Vol 34 (8) – Q1
Tahun	2022
Hasil Penelitian	Penelitian ini menganalisis penggunaan algoritma *Gravitational Search Algorithm* (GSA) dalam menghasilkan test case kombinatoris pada pengujian T-way. GSA adalah algoritma optimasi stokastik yang inspirasi dari *Newton's law of gravity and motion* dan telah digunakan untuk menemukan solusi optimal terhadap permasalahan sehari-hari. Hasil penelitian ini menunjukkan bahwa GSA dapat digunakan sebagai strategi pengujian dengan teknik *T-way* testing untuk dapat menghasilkan *test case* yang optimal dan efektif [20].
Adopsi Penelitian	Teknik pengujian *T-Way* atau *Pairwaise* akan di adopsi untuk diterapkan pada penelitian ini. Teknik pengujian *pairwaise* disini akan dimanfaatkan untuk dapat menghasilkan *test case* dari kemungkinan akan kombinasi yang terjadi dari beberapa data pada *software annotation tool*. Teknik pengujian *pairwaise* ini juga akan dikombinasikan dengan beberapa teknik pengujian lainnya untuk dapat memperoleh cakupan yang lebih baik dalam proses pengujian perangkat lunak.
Penelitian 6	
Judul Penelitian	*Test Suite Generation for Boolean Conditions With Equivalence Class Partitioning*
Nama Peneliti	Hallé, Sylvain
Nama Jurnal	Proceedings of the IEEE/ACM 10th International Conference on Formal Methods in *Software* Engineering
Tahun	2022
Hasil Penelitian	Penelitian ini menjelaskan tentang sebuah algoritma yang digunakan untuk menghasilkan rangkaian *test case* dengan cara melakukan reduksi dari permasalahan yang dihadapi. Penelitian ini menerapkan *equivalence class partitioning* pada penghasilan *test case* untuk kondisi boolean, yang merupakan bagian dari pengujian *software*. *Equivalence class partitioning* dapat digunakan untuk meningkatkan efisiensi dan efektivitas pengujian, serta membantu dalam mengurangi jumlah *test case* yang diperlukan. Hasilnya menunjukkan bahwa teknik *equivalence class partitioning* dapat digunakan sebagai strategi yang efektif untuk menghasilkan *test case* pada kondisi *Boolean* [21].
Adopsi Penelitian	Cara untuk menghasilkan rangkaian *test case* dengan melakukan reduksi dari permasalahan akan diadopsi dari penelitian tersebut untuk digunakan pada penelitian ini. Teknik pengujian yang digunakan dalam penelitian ini juga berupa teknik pengujian *equivalence class partitioning* untuk dapat mengelompokkan value yang dimiliki pada beberapa fitur di *software* kedalam kelas yang nantinya akan dihasilkan *test case* dari setiap *valuenya*.
Penelitian 7	
Judul Penelitian	*Black Box Testing on ukmbantul.com Page with Boundary Value Analysis and Equivalence Partitioning Methods*

Nama Peneliti	Sholeh, Muhammad; Gisfas, Irmah; Cahiman; Fauzi, Muhammad Anwar
Nama Jurnal	Journal of Physics: Conference Series – Q4
Tahun	2021
Hasil Penelitian	Ukmbantul.com adalah sebuah website yang digunakan untuk promosi potensi kraftangan/bisnis rumahan di Bantul. Penelitian ini dilakukan untuk mengetahui keberhasilan pengembangan ukmbantul.com dalam hal menangani *input* data yang tidak sesuai dengan aturan. Metode pengujian yang digunakan adalah metode *black box testing* yang melakukan pengujain tanpa perlu mengetahui struktur program dari sistem informasi tersebut. Teknik *boundary value analysis* digunakan dalam penelitian ini untuk merancang *test case* pada bagian yang menangani *input* data [15].
Adopsi Penelitian	Pengujian fungsional pada penelitian ini akan menggunakan metode *black box testing* sama halnya dengan penelitian sebelumnya untuk dapat mengevaluasi program yang dikembangkan oleh perusahaan. Dalam tahap pengujian juga akan digunakan teknik *boundary value analysis* untuk dapat menghasilkan *test case* yang tepat dalam pengujian.
Penelitian 8	
Judul Penelitian	Pengujian Perangkat Lunak Sistem Informasi Peminjaman PlayStation dengan Teknik *Boundary Value Analysis* Menggunakan Metode *Black Box Testing*
Nama Peneliti	Ahrizal, Dekif; Khaerul Miftah, Maula; Kurniawan, Romi; Zaelani, Teguh
Nama Jurnal	Jurnal Informatika Universitas Pamulang Vol 5(1) – SINTA 4
Tahun	2020
Hasil Penelitian	Hasil penelitian ini menunjukkan bahwa pengembangan sistem informasi peminjaman *playstation* telah mempertimbangkan batasan pada data masukan, yang terlihat pada form data *entry* yang telah dilakukan validasi sesuai dengan batasan yang berlaku. Pengujian ini menggunakan metode *black box testing*, yang merupakan pengujian yang tidak mengacu pada bagian dalam sistem, dan menggunakan metode *boundary value analysis* untuk meningkatkan efektivitas dan efisiensi pengujian [22].
Adopsi Penelitian	Penerapan teknik *boundary value analysis* yang dikombinasikan dengan teknik *equivalence partitioning* pada penelitian tersebut akan digunakan dalam penelitian ini sebagai teori untuk membantu perancangan *test case* yang dapat mencakup sebanyak mungkin kemungkinan dengan lebih efisien tanpa perlu melakukan pengujian pada semua *value* yang digunakan untuk pengujian *software annotation tool* pada perusahaan PT. Fata Organa Solusi.
Penelitian 9	
Judul Penelitian	*Boundary Value Analysis Testing Against Library Applications Using the Black Box Method as System Performance Optimization*
Nama Peneliti	Gilang Ryan Fernandes; Ika Mei Lina
Nama Jurnal	Jurnal E-Komtek (Elektro-Komputer-Teknik) Vol 5(1)
Tahun	2021
Hasil Penelitian	Penelitian ini menunjukkan bahwa pengujian *boundary value analysis* dapat digunakan untuk menemukan kesalahan dalam nilai data masukan pada aplikasi perpustakaan, yang dapat mengakibatkan kesalahan dalam data yang disimpan. Setelah

	pengujian *boundary value analysis* dilakukan, semua fungsi aplikasi perpustakaan berjalan sesuai dengan rancangan awal dan tidak ada kesalahan atau *bug* yang ditemukan [23].
Adopsi Penelitian	Teknik pengujian *boundary value analysis* akan digunakan pada penelitian ini sebagai salah satu teknik untuk perancangan *test case* yang efektif memastikan tidak adanya kesalahan ataupun *bug* yang terjadi pada nilai-nilai perbatasan dari fungsi input pada *software annotation tool*.
Penelitian 10	
Judul Penelitian	Rancangan Sistem Informasi Cuti Pegawai pada PT. Samco Farma Berbasis Web
Nama Peneliti	Bayu Septian Erlangga, Angga Aditya Permana
Nama Jurnal	Jurnal Minfo Polgan Vol 12(1)
Tahun	2023
Hasil Penelitian	Penelitian ini menghasilkan sistem informasi cuti pegawai berbasis web yang memudahkan admin dan pegawai di PT. Samco Farma dalam mengelola data pengajuan cuti dengan lebih cepat dan akurat. Penelitian ini menggunakan metode pengembangan RAD dan pendekatan black box testing untuk memastikan fungsionalitas sistem yang dikembangkan. Hasilnya, sistem informasi cuti ini tidak hanya mempercepat proses pengajuan cuti, tetapi juga mengurangi kekeliruan dan keterlambatan laporan karena data tersimpan rapi dan aman di dalam sistem dan database, meningkatkan efisiensi operasional di PT. Samco Farma [24].
Adopsi Penelitian	Pendekatan *black box* testing untuk melakukan pengujian fungsional akan di adopsi pada penelitian ini untuk digunakan sebagai metode pengujian sebuah perangkat lunak tanpa memperhatikan aspek internal ataupun kode dari sistem yang berjalan.
Penelitian 11	
Judul Penelitian	Comparative analysis of solutions used in automated testing of Internet applications
Nama Peneliti	Psujek, Magdalena; Radzik, Aleksandra; Kozieł, Grzegorz
Nama Jurnal	Journal of Computer Sciences Institute Vol 18
Tahun	2021
Hasil Penelitian	Penelitian ini melakukan analisis terhadap beberapa framework automation testing yang sering digunakan oleh perusahaan dalam melakukan pengujian. Beberapa framework tersebut antara lain yaitu TestComplete, Selenium, Cypress, dan juga Robot Framework. Terdapat beberapa kriteria yang digunakan dalam melakukan analisis tersebut antara lain seperti, waktu eksekusi, pengujian pararel, jumlah test case yang dapat di capai. Hasil dari penelitian tersebut menunjukkan bahwa TestComplete merupakan automation testing framework terbaik yang dapat digunakan oleh perusahaan berdasarkan kriteria pengujian.
Adopsi Penelitian	Dalam penelitian ini juga terdapat penggunaan *automation framework* untuk melakukan pengujian pada produk Annotation Tool dari perusahaan PT. Fata Organa Solusi. Perbandingan *automation testing tool* akan dilakukan antara Selenium dan juga Cypress, dimana kriteria yang dipakai untuk melakukan perbandingan disesuaikan berdasarkan kebutuhan perusahaan. Cypress pilih karena merupakan *automation framework* terbaik kedua berdasarkan penelitian sebelumnya serta dapat bersaing ketat dalam beberapa aspek dengan selenium.

Penelitian ini akan melakukan perbandingan antara dua alat pengujian otomatis, yaitu *Selenium* dan *Cypress* dengan berfokus pada pengujian fungsional untuk *Annotation Tool*, *output* yang akan dihasilkan penelitian ini berupa tabel perbandingan antara kedua alat pengujian otomatis dengan beberapa kriteria yang akan dibandingkan yaitu waktu pengujian, jumlah *test case* yang berhasil dijalankan, banyaknya *test case* yang gagal, rata rata waktu pengujian, pengujian pararel dan waktu pengujian pararel dimana mengacu pada penelitian terdahulu 1 dan 2.

Pengujian akan dilakukan dengan pendekatan *black box testing* untuk melakukan pengujian yang berfokus pada fungsi dari *annotation tool* seperti yang dilakukan pada penelitian terdahulu 10, 3 dan 2. Pengujian perangkat lunak dalam kasus ini membutuhkan *test case* sebagai acuan dalam melakukan rangkaian tahapan yang menghasilkan skenario pengujian. Beberapa teknik untuk menghasilkan *test case* yang akan digunakan dalam penelitian yaitu *equivalence class partitioning* seperti yang dilakukan penelitian terdahulu 6 dan 8, kemudian teknik *boundary value analysis* seperti pada penelitian terdahulu 7 dan 9. *Equivalence class partitioning* membagi rentang input menjadi kelas-kelas setara untuk memastikan bahwa setiap kelas dites minimal satu kali. Sementara itu, *boundary value analysis* menitikberatkan pada pengujian pada batas-batas rentang input.

Penelitian ini mengadopsi pendekatan yang unik dengan mengkombinasikan kedua teknik pengujian tersebut, yang umumnya dilakukan secara terpisah. Dalam penelitian terdahulu 4, kedua teknik ini telah terbukti efektif dalam mengidentifikasi kasus uji yang kritis. Namun, penelitian ini bertujuan untuk mengeksplorasi apakah kombinasi kedua teknik tersebut dapat meningkatkan efisiensi dan efektivitas pengujian jika dikombinasikan dengan pengujian otomatis seperti yang diperoleh dari penelitian terdahulu 5.

Hal yang membedakan penelitian ini dengan penelitian sebelumnya adalah menguji *test case* yang dirancang menggunakan *Selenium* dan *Cypress*.

Selenium telah menjadi alat pengujian otomatis yang populer dan mapan dalam industri perangkat lunak, sementara *Cypress* adalah alat yang relatif baru dengan pendekatan yang berbeda dalam pengujian otomatis. Melalui pengujian fungsional, penelitian ini akan mengevaluasi kemampuan kedua alat dalam mengeksekusi *test case* yang dihasilkan dari kombinasi *equivalence class partitioning* dan *boundary value analysis*. Data yang diperoleh dari pengujian akan dianalisis secara komprehensif untuk mengidentifikasi kelebihan dan kekurangan dari masing-masing alat, serta potensi keuntungan yang dapat diperoleh dari penggunaan kombinasi teknik pengujian yang telah diusulkan.

2.2 Teori tentang Topik Skripsi

2.2.1 Software Testing

Pengujian perangkat lunak atau *software testing* merupakan sebuah proses pengevaluasian *software* apakah sesuai dengan yang diharapkan dan juga sesuai pada persyaratan yang ditetapkan [25]. Tujuan dari *software testing* adalah untuk memastikan bahwa produk *software* yang dikembangkan mencapai standar kualitas yang ditetapkan, hal tersebut termasuk memeriksa fungsionalitas, kinerja, keamanan, dan keandalan dari *software* tersebut sehingga ketika sampai di tangan pengguna produk tersebut dapat beroperasi dengan baik untuk memenuhi kebutuhan dari pengguna.

Dalam *software testing* sendiri terdapat beberapa jenis pengujian yang dilakukan dalam siklus pengembangan sebuah *software* antara lain asalah sebagai berikut:

1. Pengujian Fungsional

   Pengujian fungsional merupakan jenis pengujian yang dilakukan untuk memastikan apakah sebuah *software* yang telah dikembangkan dapat berfungsi sesuai dengan spesifikasi dan juga kebutuhan fungsional yang telah ditentukan.

2. Pengujian Non-fungsional

   Pengujian non-fungsional merupakan jenis pengujain yang bertujuan untuk memeriksa aspek seperti performa, keamanan, serta aspek yang lainnya yang tidak termasuk kedalam pengujian fungsional.

3. Pengujian Regresi

   Pengujian regresi merupakan jenis pengujian yang bertujuan untuk memastikan apakah perubahan ataupun pembaruan dalam kode tidak mempengaruhi fungsi yang sudah berjalan pada *software* yang dikembangkan.

4. Pengujian Performa

   Pengujian performa merupakan jenis pengujian untuk memastikan dan pengoptimalkan kinerja *software* pada kondisi yang memberatkan *software* tersebut untuk dapat berjalan dengan optimal.

5. Pengujian kompabilitas

   Pengujian kompabilitas merupakan jenis pengujian yang berfungsi untuk memastikan apakah *software* tersebut dapat berjalan dengan baik pada berbagai platform dan juga environment.

6. Pengujian Keamanan

   Pengujian keamanan merupakan jenis pengujian untuk menidentifikasi dan mitigasi resiko keamana yang mungkin ada dalam *software* yang telah dikembangkan [26].

7. *User Acceptance Testing*

   User acceptance testing merupakan jenis pengujian yang memastikan apakah *software* yang dikembangkan dapat memenuhi kebutuhan dan harapan pengguna akhir [27].

2.2.2 Functional Testing

Functional testing atau pengujian fungsional merupakan salah satu proses yang dilakukan ketika ingin melakukan pengujian pada sebuah *software* [28]. Jenis pengujian ini berfokus untuk mengkaji fungsionalitas serta fitur teknikal dari sebuah *software* ataupun sistem yang dikembangkan. Tujuan utama dari pengujian fungsional adalah untuk mengevaluasi apakah program ataupun sistem yang telah dikembangkan dapat melakukan tugas sesuai dengan apa yang diharapkan dengan benar. Berikut merupakan beberapa langkah yang dilakukan ketika menerapkan pengujian fungsional,

1. Mendefinisikan kebutuhan fungsional, untuk melakukan functional testing, pengujai perlu menetukan terlebih dahulu kebutuhan fungsional yang diharapkan oleh pengguna dari program tersebut.
2. Membuat dokumen testing, setelahnya penguji akan menyusun sebuah dokumen yang mencatat semua kebutuhan fungsional sesuai dengan yang diharapkan oleh pengguna. Dokumen tersebut membantu penguji untuk melakukan pengujian secara berkala.
3. Mengidentifikasi semua kemungkinan kasus penggunaan, penguji akan membuat *test case* yang mencakup semua kemungkinan yang dapat dilakukan oleh pengguna mulai dari langkah yang diperlukan untuk menjalankan sistem hingga ke langkah yang mungkin akan membuat sistem menjadi bermasalah ketika dijalankan [17]. Semua kemungkinan tersebut perlu dieksekusi oleh penguji untuk mengevaluasi program yang dibuat apakah sudah berjalan dengan baik dan sesuai dengan yang diharapkan pengguna.

2.3 Teori tentang Framework / Algoritma yang digunakan

2.3.1 Equivalence Class Partitioning

Equivalence class partitioning merupakan sebuah teknik pengujian perangkat lunak yang digunakan untuk melakukan pengelompokan data input kedalam partisi atau kelas yang setara atau disebut ekuivalensi. Dasar teori

dari teknik *equivalence class partitioning* ini adalah melakukan pengujian pada beberapa perwakilan yang dianggap valid dan tidak valid pada rentang input sehingga tidak perlu melakukan pengujian pada semua nilai [29]. Cara kerja dibalik *equivalence class partitioning* adalah mengambil perilaku dari *user requirement* yang kemudian dijabarkan dan dikelompokkan kedalam kelas masing masing yang dapat mewakili kemungkinan yang setara dengan satu *test case* [15].

Teori *equivalence class partitioning* ini pertama kali dikembangkan pada tahun 1970-an dimana para peneliti memperkenalkan sebuah metode baru yang dapat digunakan untuk mengurangi jumlah *test case* yang dibutuhkan disisi lain tetap memastikan *user requirement* terpenuhi [30]. Penggunaan teknik *equivalence class partitioning* mulai umum diterapkan dalam industri pada akhir tahun 1980-an dikarenakan peningkatan kesadaran akan pentingnya *software* testing yang efektif dan efisien. Teknik pengujian *equivalence class partitioning* ini termasuk kedalam jenis pengujian *black box testing*.

Terdapat beberapa tahapan yang dapat dilakukan untuk menerapkan teknik *equivalence class partitioning*, berikut merupakan penjelasan langkah penerapan teknik equivalencen class partitioning,

1. Identifikasi kondisi, tahapan pertama adalah melakukan identifikasi pada parameter *input* yang akan digunakan dalam *software* yang akan diuji.
2. Pemisahan menjadi kelas ekuivalen, setelah memperoleh informasi dari parameter input yang digunakan pada *software* yang diuji, parameter tersebut akan dipisahkan menjadi beberapa kelas ekuivalensi. Kelas ekuivalen merupakan kumpulan input yang diperlakukan setara oleh *software* yang diuji.
3. Pilih *representative* dari setiap kelas, setelah membentuk kelas ekuivalen selanjutnya adalah memilih nilai yang dapat menjadi *representative* dari masing masing kelas yang telah dibentuk. Representatif dari setiap kelas harus menggambarkan perilaku yang diharapkan dari kelas tersebut

4. Merancang *test case*, nilai *representative* yang telah dipilih akan digunakan sebagai input dalam pembuatan *test case*. *Test case* harus mencakup setiap nilai yang dipilih dari setiap kelas ekuivalen.
5. Eksekusi pengujian, *test case* yang telah dirancang akan dilakukan tahap pengujian langsung pada *software* yang diuji untuk memperhatikan respon dari *software* terhadap setiap *input* yang diberikan. Dalam tahapan ini perlu dipastikan bahwa *software* berperilaku sesuai dengan fungsi yang diharapkan pada setiap kelas ekuivalen.
6. Analisis Hasil, hasil dari pengujian akan di analisis untuk dilihat apakah *software* bereaksi sesuai dengan apa yang diharapkan. Jika terdapat kesalahan atau *output* yang keluar tidak sesuai harapan maka tahapan pengujain dapat diulang dari awal untuk memperbaiki pendekatan *equivalence class partitioning* ini.

Contoh kasus penerapan *equivalence class partiti*oning adalah sebagai berikut, dalam kasus ini terdapat sebuah sistem yang menerima nilai umur dalam bentuk angka. Rentang nilai valid yang diterima oleh sistem adalah orang yang berumur 20 hingga 30 tahun, maka dengan teknik pengujian *equivalence class partitioning* kasus tersebut dapat dijabarkan sebagai berikut:

1. Parameter *input* berupa angka
2. Terdapat 3 kondisi dari kasus tersebut yaitu orang dengan umur 20 hingga 30 tahun (valid), orang dengan umur dibawah 20 tahun (invalid), orang dengan umur diatas 30 tahun (invalid). Dari kondisi ini dapat dibentuk menjadi 3 kelas ekuivalen. Illustrasinya seperti yang ditunjukkan pada gambar 2.1

Age

Acceptance value 20 - 30

Equivalence Partitioning		
Invalid	Valid	Invalid
<=20	20 - 30	>=30

Gambar 2. 1 kelas equivalensi

3. Nilai 5 dan 17 akan menjadi *representative* untuk kelas ekuivalen orang dengan umur di bawah 20 tahun, nilai 38 dan 45 akan menjadi *representative* kelas ekuivalen orang dengan umur diatas 30 tahun, kemudian nilai 22 dan 29 akan menjadi *representative* kelas ekuivalen orang dengan umur 20-30 tahun.
4. Dari nilai *representative* tersebut dapat menghasilkan *test case* seperti berikut,

Test Case		
Invalid	Valid	Invalid
5	22	38
17	29	45

Gambar 2. 2 *test case*

Dari *test case* pada gambar 2.2 tersebut akan dilakukan pengujian pada sistem setelahnya akan dilakukan analisis terhadap hasil yang diperoleh

apakah sistem dapat menangani *input* sesuai dengan *user requirement* yang dimiliki.

2.3.2 Boundary Value Analysis

Boundary value analysis merupakan salah satu teknik pengujian perangkat lunak yang biasanya digunakan untuk mengidentifikasi kesalahan dalam rentang nilai pada sebuah *input*. Dasar teori dari teknik *boundary value analysis* ini adalah sebuah gagasan dimana kesalahan seringkali terjadi pada area sekitar batas nilai pada sebuah *input* [31]. Teknik pengujian *boundary value analysis* ini juga dikategorikan kedalam teknik pengujiian *black box* testing dimana tujuan dari teknik *boundary value analysis* ini adalah untuk menguji nilai perbatasan atau yang disebut *boundary* dari rentang nilai pada *input* [15].

Perkembangan teknik pengujian *boundary value analysis* ini pertama kali dimulai pada tahun 1980-an, dimana para peneliti merespon terhadap kecendrungan bahwa sebagain besar kesalahan terjadi di sekitar batas pada nilai input. Teknik *boundary value analysis* ini sering dikombinasikan dengan teknik pengujian *equivalence class partitioning* untuk memperoleh cakupan testcase yang lebih baik dan masih sangat efektif untuk digunakan hingga saat ini [22].

Ketika menerapkan teknik pengujian *boundary value analysis*, terdapat beberapa tahapan yang perlu dijalani. Beberapa tahapan yang perlu dilakukan dalam penerapan teknik pengujian *boundary value analysis* adalah sebagai berikut:

1. Identifikasi batas, tahapan pertama adalah menentukan nilai batas atas dan batas bawa dari rentang nilai *input* untuk setiap variable *input* yang akan digunakan pada *software*.
2. Pemilihan nilai, tahapan kedua adalah melakukan pemilihan nilai untuk digunakan dalam perancangan *test case*. Terdapat beberapa nilai yang

akan dipilih pada tahapan ini yaitu nilai dari batas itu sendiri, nilai yang berada didekat batas bisa berupa nilai setelah batas atau sebelum batas, serta nilai yang masuk kedalam rentang batasan yang telah ditentukan.

3. Perancangan *test case*, semua nilai yang telah dipilih pada tahapan sebelumnya akan digunakan sebagai perwakilan dalam perancangan sebuah *test case* yang akan digunakan pada proses pengujian.
4. Eksekusi pengujian, *test case* yang telah dirancang akan dilakukan tahap pengujian langsung pada *software* yang diuji untuk memperhatikan respon dari *software* terhadap setiap *input* yang diberikan.
5. Analisis hasil, hasil dari pengujian akan di analisis untuk dilihat apakah *software* bereaksi sesuai dengan apa yang diharapkan. Jika terdapat kesalahan atau *output* yang keluar tidak sesuai harapan maka tahapan pengujin dapat diulang kembali dari awal.

Contoh kasus penerapan teknik *boundary value analysis* adalah sebagai berikut, dalam kasus ini terdapat sebuah sistem yang menerima nilai umur dalam bentuk angka. Rentang nilai valid yang diterima oleh sistem adalah orang yang berumur 18 hingga 56 tahun, maka dengan teknik pengujian *boundary value analysis* kasus tersebut dapat dijabarkan sebagai berikut:

1. Nilai batas pada rentang *input* adalah 18 hingga 56
2. Terdapat 7 nilai yang akan ditentukan dalam tahapan ini
    a. Nilai dibawah batas awal = 17
    b. Nilai batas awal = 18
    c. Nilai diatas batas awal = 19
    d. Nilai dalam rentang input = 37
    e. Nilai dibawah batas akhir = 55
    f. Nilai batas akhir = 56
    g. Nilai diatas batas akhir = 57
3. Jika semua nilai tersebut dijabarkan menjadi *test case* maka berikut gambaran *test case* yang akan terbentuk

Boundary Value Analysis(Age accepts 18 to 56)		
Invalid (min-1)	Valid (min, min + 1, nominal, max – 1, max)	Invalid (max + 1)
17	18, 19, 37, 55, 56	57

Gambar 2. 3 test case

Dari *test case* pada gambar 2.3 tersebut akan dilakukan pengujian pada sistem setelahnya akan dilakukan analisis terhadap hasil yang diperoleh apakah sistem dapat menangani input sesuai dengan user requirement yang dimiliki.

2.4 Teori tentang Tools / Software yang digunakan

2.4.1 Selenium

Selenium merupakan sebuah *library* yang dikembangkan oleh Jason Huggins pada tahun 2004 dengan tujuan sebagai alat untuk membantu proses *automation* khususnya pada tahap pengujian sebuah *software* berbasis web [32]. Pada mulanya, alat tersebut Bernama "JavaScriptTestRunner." Kemudian, Huggins bergabung dengan ThoughtWorks dan bekerja dengan Paul Hammant untuk mengembangkan proyek tersebut menjadi *Selenium* RC (Remote Control). Pada tahun 2007, Simon Stewart mengembangkan *Selenium* WebDriver, yang kemudian menjadi inti dari proyek *Selenium*. Pada tahun 2009, *Selenium WebDriver* diintegrasikan dengan *Selenium RC* untuk membentuk *Selenium* 2 serta *Selenium* 3, yang dirilis pada tahun 2016.

Selenium digunakan untuk melakukan *automation* khususnya pada *functional test* dan *user acceptance test* pada *software* berbasis web. Selain itu *Selenium* juga dapat digunakan untuk *scraping* data dari *website* dan

melakukan *automation* pada aspek lainnya dalam *website* [6]. Cara kerja dari *Selenium WebDriver* adalah berinteraksi dengan *browser* menggunakan protokol yang sesuai dengan *browser* yang akan diuji. *Selenium* akan beroperasi dengan menjalankan *script* berisi serangkaian perintah yang mendefinisikan langkah-langkah yang akan dijalankan oleh *browser*. Kemudian *WebDriver* akan mengirimkan perintah ke *browser* melalui *driver* khusus untuk *browser* tersebut. Setelahnya *browser* akan mengeksekusi perintah dan mengembalikan hasil eksekusi ke *WebDriver*.

Selenium sendiri mendukung pengguna untuk menulis *script* dengan berbagai bahasa pemrograman seperti Java, Python, C#, Ruby, dan lainnya untuk dapat menyesuakan kebutuhan yang digunakan pada project pengguna [33]. Selain itu, *Selenium* juga memiliki beberapa keunggulan seperti mendukung untuk melakukan *automation* pada berbagai jenis *browser*, bersifat *open source* dan memiliki komunitas besar yang aktif, ketersediaan *Selenium Grid* untuk pengujian terdistribusi. Akan tetapi *Selenium* juga memiliki beberapa tantangan yang perlu dihadapi dalam penggunaannya seperti kompleksitas dalam mengatasi elemen yang dinamis atau *asynchronous*, serta memerlukan pemahaman tentang *Document Object Model (DOM)* yang baik [16].

Selenium tidak hanya merupakan satu alat penujian otomatis akan tetapi *selenium* juga memiliki beberapa komponen seperti:

- *Selenium WebDriver*, merupakan alat yang digunakan untuk membantu dalam melakukan proses *automation testing* pada *browser* dengan mengirimkan perintah melalui *driver* ke *browser*.
- *Selenium Grid*, merupakan alat yang digunakan untuk mengeksekusi test secara terdistribusi di beberapa mesin dan *browser* secara simultan.
- *Selenium Integrated Development Environment (IDE)*, merupakan alat yang digunakan untuk melakukan proses *automation* dengan cara *record* sehingga tidak perlu menulisakan *script* secara manual.

2.4.2 Cypress

Cypress merupakan sebuah *automation tool* yang dikembangkan oleh Brian Mann dan Drew Lanham dan pertama kali dirilis pada tahun 2014. Namun, popularitasnya meningkat secara signifikan pada beberapa tahun terakhir. *Cypress* dikenal karena pendekatannya yang berbeda dalam otomatisasi pengujian.

Cypress digunakan untuk melakukan *automation testing* pada *software* berbasis web. Seperti halnya *Selenium*, *Cypress* dapat digunakan untuk *functional test*, *integration test*, dan *user acceptance test*. Akan tetapi, *Cypress* juga menekankan pada otomatisasi *frontend* (*end-to-end*) dengan fokus pada simulasi interaksi yang dilakukan pengguna. Cara kerja dari *Cypress* adalah dengan berjalan langsung pada browser yang akan di uji, hal tersebut memungkinkan *Cypress* untuk memiliki kendali yang lebih besar atas browser yang dijalankan dan memahami lebih baik perilaku *software*. *Cypress* juga dioperasikan dengan menjalankan script berisi perintah yang akan dilakukan oleh browser, akan tetapi *Cypress* menyediakan *built-in developer tools* yang memungkinkan pemantauan, *debugging*, dan pengecekan status *software* saat melakukan pengujian.

2.4.3 Visual Studio Code

Visual Studio Code merupakan sebuah *text editor* yang dikembangkan oleh perusahaan Microsoft pada tahun 2015. *Visual Studio Code* sendiri dirancang sebagai *text editor* dengan ukuran yang ringan agar dapat berjalan dengan baik pada segala jenis perangkat komputer, selain itu *text editor* ini juga telah mendukung untuk dijalankan pada berbagai sistem operasi seperti Windows, MacOS, maupun Linux sehingga *text editor* ini *flexible* untuk digunakan pada berbagai platform [34]. *Text editor* ini memiliki tampilan antarmuka yang sederhana akan tetapi sudah mencakup fungsi yang cukup

kompleks sehingga pengguna tidak akan mengalami kesulitan dalam mengoperasikannya.

Visual Studio Code memiliki beberapa fitur unggulan seperti mendukung berbagai bahasa pemrograman yang dipakai oleh pengguna seperti python, javascript, php, dll. Terdapat juga fitur *extension* dan juga *intellisense* yang dapat membantu pengguna dalam melakukan penulisan kode yang lebih baik dan juga lebih efisien. *Visual Studio Code* sudah terintegrasi dengan sistem Git dan juga terminal langsung didalam *text editor* untuk membantu pengguna melakukan kolaborasi serta mengendalikan jalannya pengembangan secara langsung tanpa perlu *software* tambahan lainnya [35]. Dengan bersifat *open source* sehingga pengguna dapat terlibat langsung dalam pengembangan *Visual Studio Code* sendiri dan juga segudang fitur yang ditawarkannya olehnya menjadikan *Visual Studio Code* sebagai *text editor* yang paling populer untuk digunakan oleh pengguna.