MODUL
PRAKTIKUM KUALITAS PERANGKAT LUNAK
Quality Assurance (QA) *Testing*

Oleh

Nama	:	Rizqillah
NIM	:	1957301020
Kelas	:	TI 4B
Dosen Pembimbing	:	Musta'inul Abdi, SST., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER

TAHUN 2022

LEMBAR PENGESAHAN

No. Praktikum	:	PKPL/4C/TI/2022
Judul	:	*Quality Assurance* (QA) *Testing*
Nama	:	Rizqillah
NIM / Kelas	:	1957301020 / TI 4B
Jurusan	:	Teknologi Informasi Dan Komputer
Prodi	:	Teknik Informatika
Tanggal praktikum	:	September 2022
Tanggal penyerahan	:	Desember 2022
Nilai	:	

Buketrata, Desember 20212

Dosen Pembimbing,

Musta'inul Abdi, SST., M.Kom.
NIP. 19911030 20190310 1 5

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Tujuan

1. Mahasiswa dapat memahami cara melakukan pengujian perangkat lunak
2. Mahasiswa dapat memahami apa yang dimaksud dengan QA (*Quality Assurance*)

1.2. Dasar Teori

Banyaknya projek aplikasi yang dikembangkan oleh tim *software developer* semakin menjamur dan banyak dilakukan seiring perkembangan teknologi di zaman sekarang. Nantinya produk dari proyek aplikasi yang dibangun *developer* akan masuk ke tahap pengujian sebelum dirilis, dan proses ini akan dilakukan oleh *Quality Assurance* atau QA.

Quality Assurance memiliki peran dan tugas yang bisa dibilang agak berbeda dengan tim produksi atau *developer*. Hal ini tak lepas dari keinginan setiap perusahaan yang menginginkan konsumen puas dengan produk yang dijual. Oleh karena itu, proses pengujian atau pengecekan guna memastikan kualitas produk yang akan dipasarkan memiliki kualitas baik dan layak dipakai.

1.2.1. *Quality Assurance* (QA)

Posisi ini nyaris dibutuhkan setiap perusahaan, utamanya di bidang teknologi karena perkembangan teknologi membuat banyak bisnis pembuat aplikasi dan *website* atau laman tersendiri. Profesi QA ini terbilang masih asing di sebagian telinga masyarakat.

Quality Assurance adalah serangkaian proses sistematis guna menentukan apakah suatu produk dan jasa harus memenuhi syarat yang ditentukan. QA akan menentukan serta menetapkan persyaratan dalam membuat atau mengembangkan produk tertentu agar memiliki kualitas yang baik. Bukan tanpa alasan mengapa kualitas suatu produk sangat penting diperhatikan.

Kualitas terbaik adalah cara paling utama menjaga kredibilitas suatu perusahaan, selain itu juga cara meningkatkan kepercayaan konsumen, proses kerja

hingga membuat perusahaan yang mampu membuat mereka bersaing dengan kompetitor. *Quality Assurance* artinya menggunakan pendekatan proses agar tidak memunculkan produk yang cacat.

Itulah mengapa biasanya QA juga akan melakukan *monitoring* pembuatan produk mulai dari tahap perencanaan hingga proses pengujian. Kegiatan itu dilakukan demi mengurangi proses pengulangan pembuatan atau *rework*, sehingga proses kerja menjadi lebih efisien serta dapat menghindari keluhan dari konsumen.

1.2.2. Perbedaan dengan *Quality Control* (QC)

Perbedaan *Quality Control* (QC) dan *Quality Assurance* (QA) terletak pada tugas dan tanggung jawab masing-masing. Walaupun sama-sama dalam satu departemen, QA memiliki peran dalam menjamin kualitas, sementara QC memiliki fungsi sebagai pengendali kualitas dari produk yang dihasilkan oleh perusahaan sesuai dengan standar keandalan, kegunaan, kinerja maupun standar lainnya.

QA memiliki tanggung jawab dalam memastikan sebuah produk sebelum dilepas ke pasaran, sebelum dirilis produk harus sudah memenuhi semua standar kualitas dalam setiap komponen. Pejabat staf QA memiliki kewajiban untuk aktif melakukan *monitoring* dan serangkaian pengujian dalam menetapkan kualitas pada pembeli.

Hal ini berbeda dengan QC dengan tanggung jawab memeriksa produk sebelum dan hingga setelah proses produksi menetapkan standar kualitas yang diperlukan. Pejabat QC memiliki hak menerima atau menolak produk yang akan dilepas ke pasaran, sehingga ketika ditemukan produk cacat maka akan dikembalikan ke bagian produksi.

Kedua posisi ini memiliki keterkaitan dan dapat bekerja secara bersamaan hingga saling berkolaborasi. Jenjang karir dari QA bisa mencapai tingkatan seorang *Project Manager* jika memiliki pengalaman dalam melakukan analisis dan melakukan audit suatu produk. Selain itu jenis pekerjaan ini juga bisa menjadikan DevOps, tugasnya mengotomatisasi proses tahap pengembangan aplikasi.

Untuk bisa menekuni pekerjaan ini seseorang bisa memulai karir lebih dulu menjadi *Customer Experience Leader* atau *IT Management* dengan peran pentingnya dalam memegang kendali penuh dari setiap proses pengembangan

produk serta memiliki kewajiban mengutamakan kebutuhan dari konsumen atau pelanggan.

1.2.3. Peran dan Tanggung Jawab

Pada umumnya tugas dari *Quality Assurance* adalah menjamin kualitas produk dari suatu perusahaan yang akan dijual atau masih dalam proses pengembangan. Selain itu QA juga memiliki beberapa tugas lain, di antaranya adalah sebagai berikut ini:

- Membuat perencanaan terhadap pengujian dan kasus pengujian terperinci serta komprehensi terstruktur.
- Melakukan tafsir, membangun dan mematuhi standar terhadap jaminan kualitas dari perusahaan yang menjual produk atau jasa.
- Melakukan analisis terhadap keluhan konsumen dan ketidaksesuaian kualitas, selain itu juga mencari akar masalah serta tindakan penyelesaian yang sesuai dengan visi perusahaan.
- Melakukan pengembangan standar baru dalam produksi sesuai dengan kebutuhan dan membuat protokol pengujian.
- Melakukan dokumentasi aktivitas jaminan kualitas dalam bentuk laporan dan audit secara internal dalam perusahaan.
- Memastikan produk yang dibuat sudah memenuhi standar perusahaan dan kebutuhan konsumen atau para pelanggan.
- Melakukan dokumentasi perbaikan terhadap produk setelah dilakukan pengujian sebelum nantinya dijual ke pasaran.
- Melakukan dokumentasi berupa catatan perbaikan yang dijadikan sebagai referensi terhadap produk setelah dilakukan pengujian.
- Melakukan penyusunan terkait perencanaan Prosedur Operasi Standar (SOP) proses produksi terhadap produk dan layanan.
- Bekerja sama, berkolaborasi dengan tim internal agar menemukan solusi dalam pemecahan masalah yang dihadapi saat itu.

1.2.4. Pengetahuan Sebagai *Quality Assurance*

Menjadi seseorang dari QA harus memahami skill dan kemampuan dasar untuk bisa bekerja secara profesional dan sesuai dengan kualifikasi kerja di suatu perusahaan, berikut di antara dari kemampuan yang dibutuhkan:

- Memahami, aktif berkomunikasi dan membaca data dengan sangat baik.
- Memiliki keahlian dalam manajemen proyek yang optimal.
- Memahami sebuah produk yang bisa mempengaruhi pola pikir, kebiasaan dan kehidupan manusia.
- Kemampuan berpikir kritis dan kemampuan analisis serta riset.
- Keahlian dalam manajerial hingga penguasaan bahasa asing yang diperlukan oleh perusahaan.
- Kemampuan analisa dan pengujian fungsional serta perbaikan dalam proses yang dilakukan.
- Memiliki pola pikir pengujian, pengujian keamanan, otomasi hingga UAT (*User Acceptance Testing*).

Dalam menjalankan proses atau tugasnya, seorang QA membutuhkan dukungan peralatan atau tools yang kompatibel sesuai dengan pekerjaannya. Seperti salah satunya instalasi aplikasi GIT, selain harus memahami algoritma dan struktur data bahasa pemrograman yang dipakai dalam perusahaan tempat mereka bekerja. Keahlian untuk menjadi QA biasanya mirip, kurang lebih seperti yang disebutkan dan untuk hard skill biasanya disesuaikan dengan posisi yang dilamar. Misalnya *Software Quality Assurance* atau QA *Engineer*, maka harus bisa menguasai SDLC (*Software Development Life Cycle*), bahasa pemrograman seperti (Java, JavaScript, Python, C++ dll), *testing tool*, *manual testing* hingga *automation testing.*

1.3. Regression Testing

Pengujian Regresi adalah jenis pengujian yang dilakukan untuk memverifikasi bahwa perubahan kode dalam perangkat lunak tidak memengaruhi fungsionalitas produk yang ada.

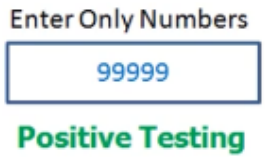
Pengujian ini untuk memastikan bahwa produk berfungsi dengan baik dengan fungsionalitas baru, perbaikan *bug*, atau perubahan apa pun pada fitur yang ada. Kasus uji yang dieksekusi sebelumnya dieksekusi kembali untuk memverifikasi dampak perubahan.

1.3.1. What is Positive and Negative Testing

Pengujian positif menentukan bahwa aplikasi Anda berfungsi seperti yang diharapkan. Jika terjadi kesalahan selama pengujian positif, pengujian gagal. Pengujian negatif memastikan bahwa aplikasi Anda dapat menangani input yang tidak valid atau perilaku pengguna yang tidak terduga dengan baik.

1.3.2. Positive Testing

Pengujian ini berupa menguji inputan nilai yang *valid*. Kemudian melihat apakah aplikasi berjalan sesuai dengan harapan. Contohnya jika inputan sebagai berikut :

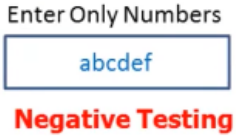


Gambar 1 Input nilai *valid*

Memasukkan nilai dengan 99999 pada kolom tersebut dapat diterima/*valid* oleh sistem dan nila selain angka seharusnya tidak boleh diterima/*invalid.*

1.3.3. Negative Testing

Pada *negative testing*, pengujian yang dilakukan berupa menguji jika inputan salah maka program akan memberikan notifikasi berupa apa. Contohnya jika inputan berikut :



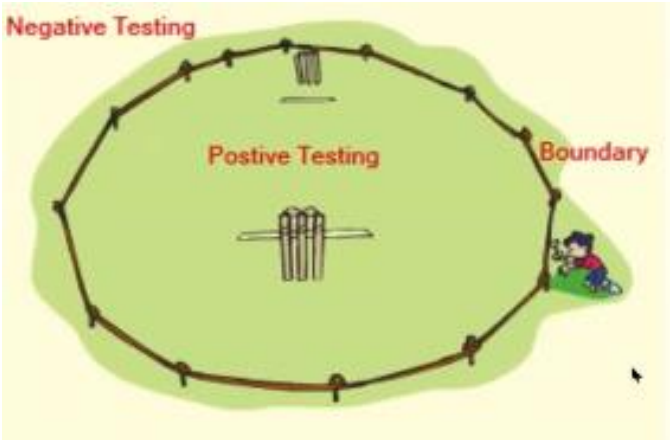
Gambar 2 Input nilai *invalid*

Menginput nilai selain dari angka atau nilai A-Z/a-z maka aplikasi akan memberikan berupa *alert* bahwa inputan tersebut bukanlah angka dan membuat *user* tidak dapat mengirimkan *form invalid* tersebut.

1.3.4. Boundary Value Analysis/Pengujian Batas Nilai

Teknik pengujian di mana kasus uji dirancang untuk memasukkan nilai-nilai pada batas. Pengujian positif pengujian berupa data input berada dalam batas nilai batas. Pengujian negatif adalah pengujian berupa data input berada di luar batas nilai batas. Analisis Nilai Batas didasarkan pada pengujian nilai batas partisi yang *valid* dan tidak *valid*. Perilaku di tepi partisi ekivalensi lebih mungkin salah daripada perilaku di dalam partisi, jadi batas adalah area di mana pengujian cenderung menghasilkan cacat.

Ini memeriksa nilai input di dekat batas yang memiliki peluang kesalahan lebih tinggi. Setiap partisi memiliki nilai maksimum dan minimum dan nilai maksimum dan minimum ini adalah nilai batas dari sebuah partisi. Adapun ilustrasi dari pengujian dapat dilihat pada gambar di bawah ini.



Gambar 3 Ilustrasi pengujian nilai batas

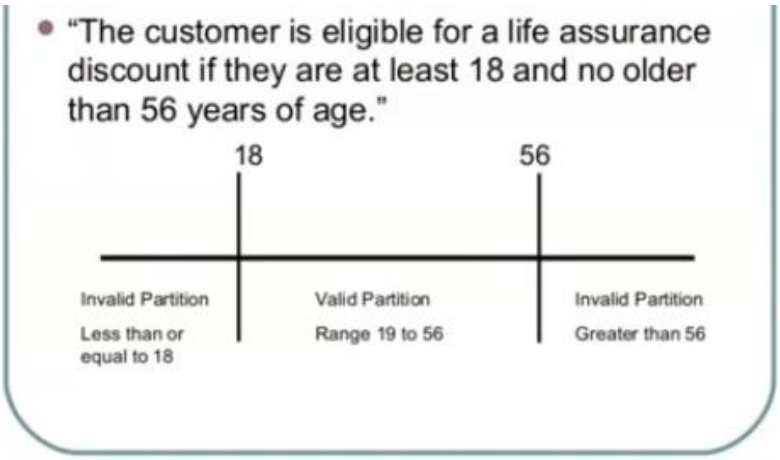
Dari ilustrasi di atas dapat dilihat bahwa pengujian batas adalah melakukan uji di batas antara *positive testing* dengan *negative testing*. Nilai batas untuk partisi yang valid adalah nilai batas yang valid. Oleh karena itu, pengujian dengan *Boundary Value Analysis* melakukan pengujian untuk setiap variabel mulai dari : nilai minimal, tepat di atas minimal, nilai nominal, tepat di bawah nilai max, dan nilai maks.

Contoh pengujian seperti jika sebuah sistem hanya menerima nilai inputan untuk umur hanya dari 18 sampai 56. Nilai minimal dari umur adalah 18, dan nilai maksimal dari umur tersebut adalah 56. Oleh karena itu pengujian batas seperti berikut.

Tabel 1 *Boundary Testing*

Invalid (min-1)	***Valid*** (min, min+1, nominal, max-1, dan max)	***Invalid*** (max+1)
17	18, 19, 37, 55, 56	57

Valid Test Cases: Pengujian yang *valid* dari nilai umur 18 sampai 56 terletak di atas nilai 17 dan di bawah nilai 57. Yaitu *valid* jika menginput nilai 18, 19, 37, 55, 56. ***Invalid Test Cases***: Ketika nilai di bawah nilai 18 dan di atas nilai 56 maka nilai akan invalid. Adapun ilustrasi dari permasalahan di atas sebagai berikut.



Gambar 4 Kasus umur dengan *Boundary*

1.3.5. Equivalence Partitioning

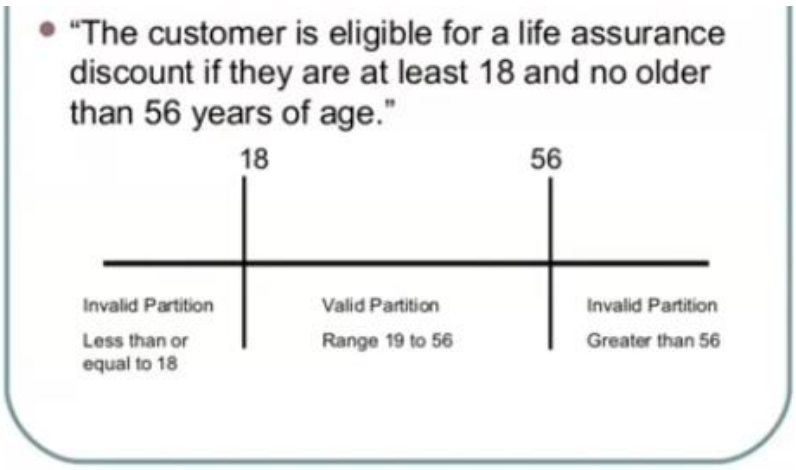
Equivalence Partitioning juga dikenal sebagai *Equivalence class partitioning* (ECP). Ini adalah teknik pengujian perangkat lunak atau pengujian *black box* yang membagi *domain input* ke dalam kelas data, dan dengan bantuan kelas data ini, kasus uji dapat diturunkan. Kasus uji yang ideal mengidentifikasi kelas kesalahan yang mungkin memerlukan banyak kasus uji arbitrer untuk dieksekusi sebelum kesalahan umum diamati.

Dalam partisi kesetaraan, kelas kesetaraan dievaluasi untuk kondisi input yang diberikan. Setiap kali input diberikan, maka jenis kondisi input diperiksa, kemudian untuk kondisi input ini, kelas Ekuivalensi mewakili atau menggambarkan kumpulan status *valid* atau tidak *valid*. Adapun ilustrasi dari *Equivalence Partitioning* sebagai berikut.

Gambar 5 Ilustrasi *Equivalence Partitioning*

Teknik pengujian yang membagi data input menjadi banyak partisi. Nilai dari setiap partisi harus diuji setidaknya sekali pengujian. Partisi dengan nilai yang *valid* digunakan untuk pengujian positif, dan partisi dengan nilai tidak *valid* digunakan untuk pengujian negatif.

Contoh :



Gambar 6 Ilustrasi kasus

Gambar di atas sama seperti kasus pada *boundary value analysis*. Pengujian secara *equivalence partitioning* adalah pengujian yang akan membagi tiap-tiap nilai ke dalam kelompok masing. Contoh dari nilai di atas adalah min = 18 dan max 56. Maka dapat diketahui terdapat 3 partisi yaitu : 0-18, 19-56-, 56+. Contoh nilai inputan : 5, 40, dan 76 dapat diambil dari setiap bagian partisi untuk menguji setiap skenario yang ada.

1.4. Web Architecture

Arsitektur pengujian adalah ilmu disiplin melihat aliran pengiriman dan mencari tahu apa, bagaimana, dan kapan pengujian untuk mencapai hasil terbaik. Pengujian ini menghasilkan kerangka kerja aktivitas pengujian yang kemudian dapat dicakup, disiapkan, dan dieksekusi.

Pengujian web arsitektur terbagi atas dua uji, yaitu secara *Front-end* dan *Back-end*. Pengujian secara *Front-end* meliputi uji aplikasi dari bagian user yang melakukan interaksi secara langsung. Contoh ilustrasi sebagai berikut.

Gambar 7 Ilustrasi interaksi user

Pada gambar di atas dapat diketahui bahwa user adalah pada bagian *Client*, dan *Server Side* adalah bagian yang tidak diketahui oleh user atau biasa disebut dengan *Back-end* aplikasi. Pada bagian *Server Side* user tidak akan mengetahui proses logika, dan hal sensitif lainnya.

Pengujian secara *Back-end* adalah bagian dari aplikasi yang tidak langsung dapat diakses oleh user, biasanya digunakan untuk menyimpan atau memanipulasi data.

1.4.1. Cara Kerja *Web Architecture*

Web Architecture bekerja secara *font-end* dan *back-end*. Adapun proses yang terjadi selama user melakukan *request* untuk membuka sebuah web terbagi atas 2 proses, yaitu proses pada bagian user yang disebut *front-end* dan proses pada bagian yang tidak dapat dilihat oleh user disebut *back-end*.

Frond-end :

- *Browser*

  User mengakses halaman web (HTTP/HTTPS *Protocol*)

Back-end :

- *Back-End*

  Mengirim kembali sebagai HTML (Bahasa Standar untuk *web browser*)

- *Browser*

  Menerjemahkan halaman web

- *Browser*: *Download* seluruh asset yang dibutuhkan seperti *script*, *font*, gambar

  Menghubungi pihak ketiga dari *back-end* seperti (metode pembayaran, *analytics*, maps)

- *Browser*

  Menampilkan halaman web ke user

1.4.2. *Back-End Testing*

Pengujian *back-end* adalah pengujian yang dilakukan dari segi logika atau segi yang tidak dapat dilihat oleh user. Pengujian dari segi *back-end* biasanya melibatkan pengujian seperti *Web Service Testing, Database Testing,* dan *Server Side Testing*. Adapun analogi dari *back-end* dan *front-end* sebagai berikut.

Gambar 8 *Back-end* dan *Front-end*

Dari gambar tersebut dapat diketahui bahwa *Back-end* adalah sebuah hal yang dilakukan untuk membuat *Front-end* sesuai dengan yang diinginkan. Seperti pada gambar tersebut, *developer* ingin menampilkan singa mengaung, akan tetapi pastinya pada bagian *back-end* atau belakang panggung menunjukkan bahwa singa tersebut diikat dan hanya menampilkan kepala.

1.4.3. *Front-end Testing*

Pengujian *front-end* adalah pengujian yang dilakukan dari segi tampilan yang dapat dilihat oleh user. Pengujian dari segi *front-end* biasanya melibatkan pengujian seperti *Web Application Testing*, *Desktop Application Testing*, dan *Mobile Application Testing*. Adapun ilustrasi dari pengujian *front-end* sebagai berikut.

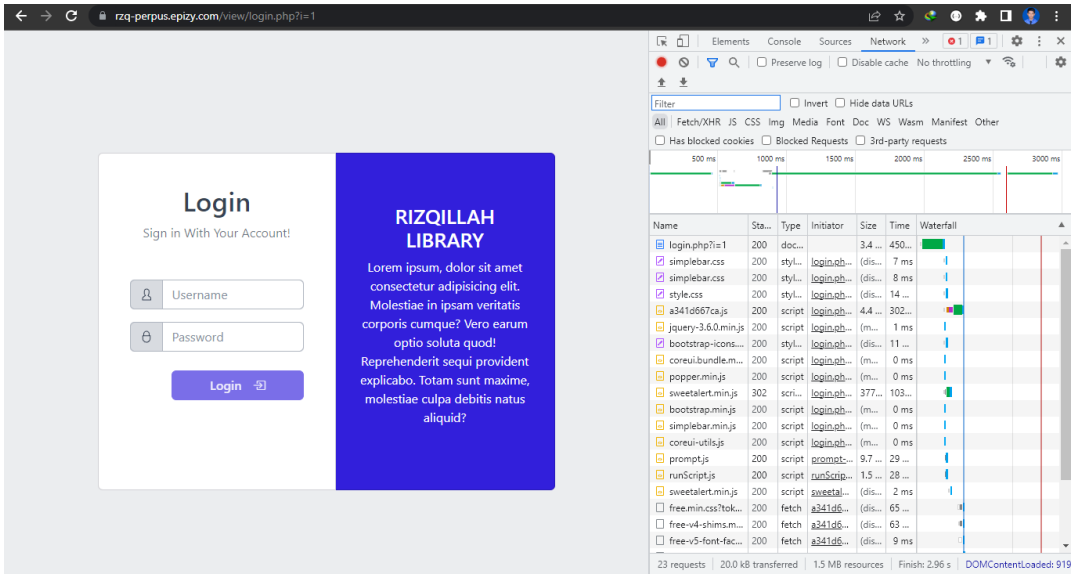
Gambar 9 Ilustrasi *front-end*

Dari gambar tersebut dapat diketahui bahwa pengujian secara *front-end* akan menguji tampilan dari sebuah web, seperti jika user membuka web menggunakan perangkat *mobile* maka tampilannya dengan perangkat *desktop* tentunya tidaklah sama. Oleh karena itu, pengujian secara *front-end* pada *web architecture* sangat dibutuhkan agar *website* yang dibangun *responsive* terhadap segala jenis lingkungan/*platform*.

1.4.4. *Front/Back-End Testing*

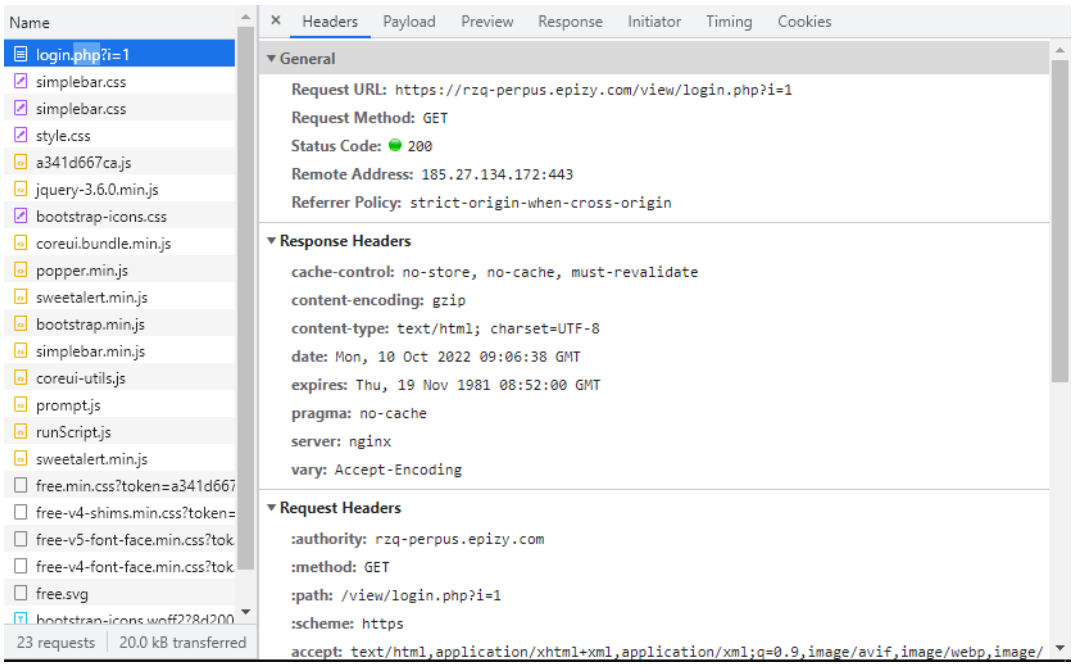
Adapun contoh pengujian terhadap *front-end* dan *back-end* adalah terhadap website https://rzq-perpus.epizy.com. Pengujian yang dilakukan sebagai berikut.

- Adapun kecepatan proses pembukaan halaman web dapat dilihat pada gambar di bawah ini



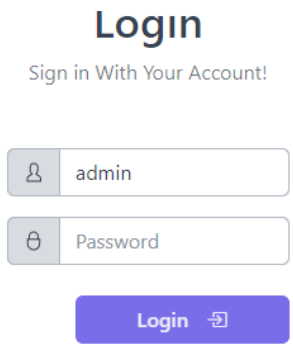
Gambar 10 *Network* dari website

- Pada bagian file login.php, kita dapat melihat informasi *load* dari file sebagai berikut



Gambar 11 *Load* file login.php

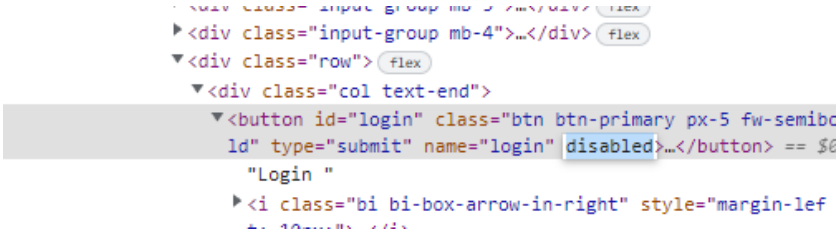
- Mencoba mengosongkan inputan password pada form login, maka hasil dari website seperti berikut



Gambar 12 Tampilan *form* login

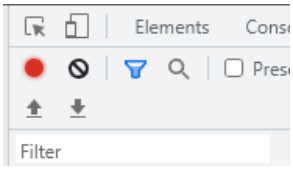
Secara otomatis tombol Login tidak akan bisa di klik jika inputan kosong.

- Adapun jika membuat inspect elemen jika ingin membuat tombol login bisa di klik dengan cara hapus *disable* pada elemen *button*.



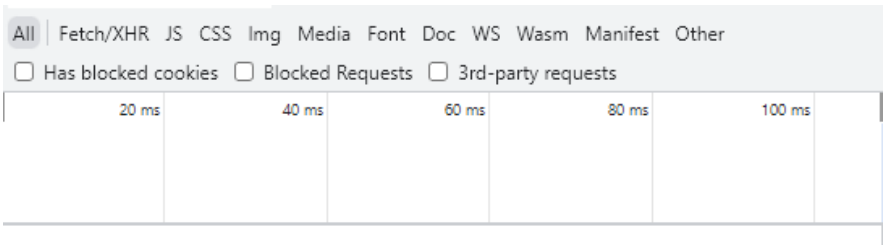
Gambar 13 Hapus *disabled button*

- Buka bagian network, dan hapus seluruh proses yang ada, maka hasilnya seperti berikut



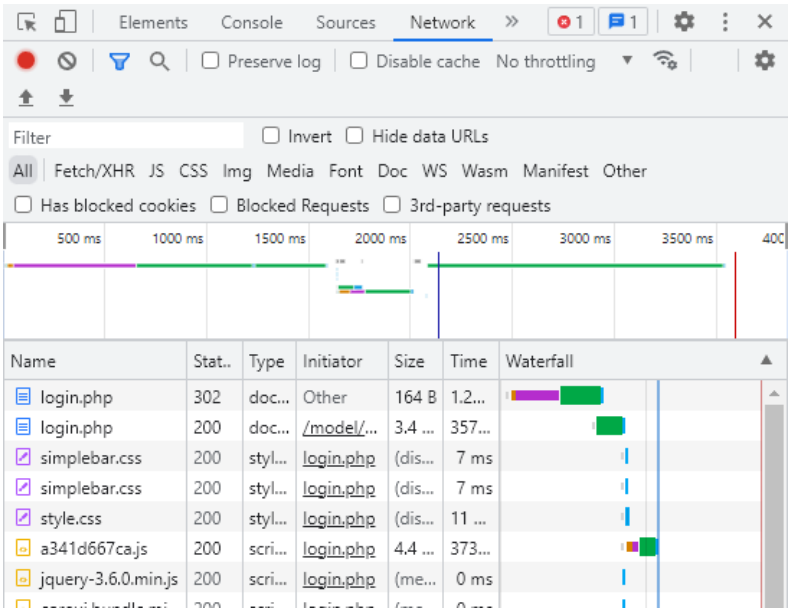
Gambar 14 Tombol *clear* network

Hasil setelah *clear* network



Gambar 15 *Clear network*

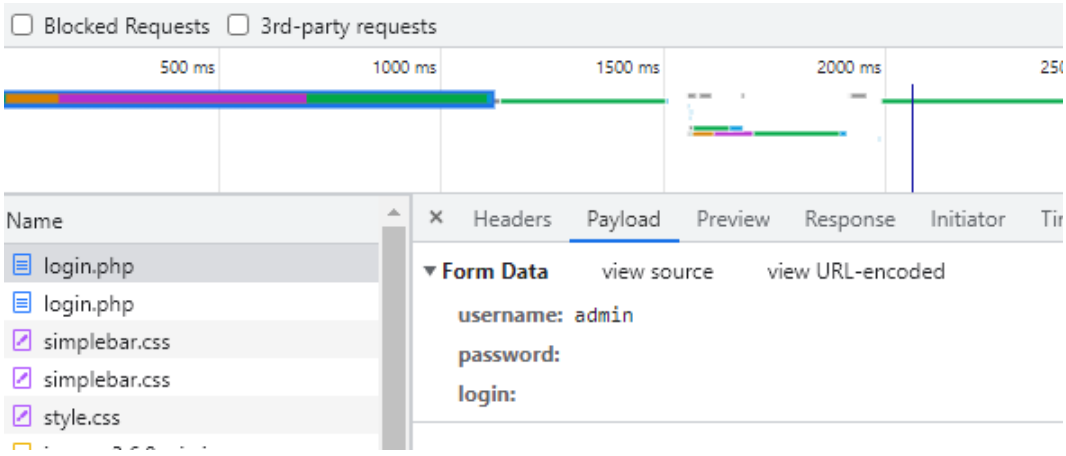
- Kemudian klik tombol login dengan password yang kosong, maka hasil pada network sebagai berikut



Gambar 16 Hasil login tanpa password

Dari hasil di atas dapat diketahui bahwa proses validasi terhadap kolom login masing kurang bagus, dikarenakan harus melakukan proses secara *front-end* dan *back-end*. Alangkah lebih bagus menggunakan proses validasi seperti menggunakan Ajax atau pengecekan javascript terlebih dahulu sebelum melakukan proses login.

- Pada bagian file login dengan status 302 kita akan dapat melihat status *payload* dengan informasi dari *username* dan *password* yang diinput

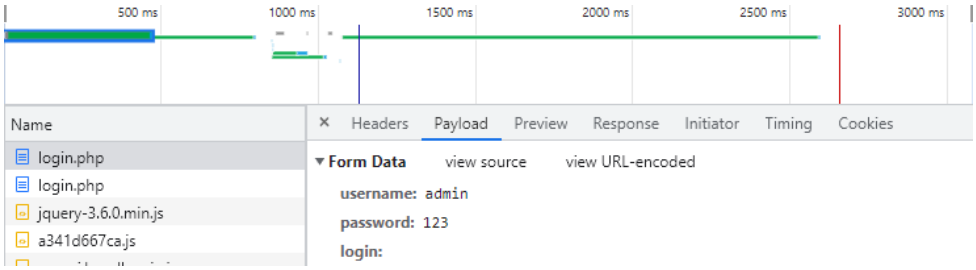


Gambar 17 *Payload Form Data*

- Kemudian jika melakukan login dengan password yang salah menghasilkan tampilan seperti berikut

Gambar 18 Gagal login

- Dan adapun berikut tampilan dari data yang dikirim dengan password yang salah



Gambar 19 *Form data* gagal login

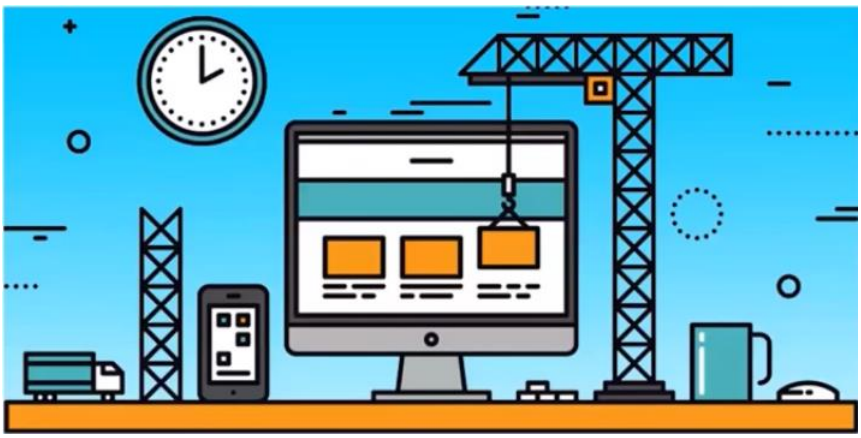
1.5. *Build and Release*

Pengembangan perangkat lunak adalah proses yang rumit. Lebih sulit daripada menulis program komputer biasa karena pelanggan mengoperasikan produk akhir. Oleh karena itu, penting untuk menguji perangkat lunak. Pengujian adalah proses memverifikasi dan memvalidasi bahwa perangkat lunak berfungsi seperti yang diharapkan sesuai dengan persyaratan. Hal ini dapat membantu untuk mengurangi biaya dan untuk memberikan produk yang berfungsi dan dapat digunakan kepada pelanggan. *Build and Release* adalah dua istilah dalam pengembangan dan pengujian.

1.5.1. *Build Definition*

Setelah mengembangkan modul perangkat lunak, pengembang mengubah kode sumber menjadi bentuk mandiri atau kode yang dapat dieksekusi. Kemudian tim pengembang menyerahkan hasil pengembangan kepada tim pengujian untuk melakukan pengujian. *Build* sedang dalam tahap pengujian; mungkin sudah menjalani pengujian atau tidak. Tim penguji perangkat lunak memeriksa aplikasi tersebut. Jika terdapat beberapa *bug* dan jika tidak memenuhi persyaratan, maka tim pengujian perangkat lunak menolak pengembangan itu. *Build* dilakukan sebelum proses perilisan perangkat lunak ke pasar.

Build adalah proses kompilasi, di mana seluruh *source code file* (*text*) diubah menjadi sebuah kode yang dapat dieksekusi. Adapun ilustrasi dari *build* dapat dilihat pada gambar di bawah ini.



Gambar 20 Ilustrasi *build* aplikasi

Dari ilustrasi di atas, dapat dilihat bahwa proses sebelum merilis sebuah aplikasi ke publik/pasar harus melakukan kompilasi atau membuat sebuah kode teks yang dibuat menjadi dapat dieksekusi oleh *platform* tujuan aplikasi. Perbedaan utama dari *build* dan *release* terletak pada pengujian perangkat lunak. Aplikasi yang masih dalam tahap *build* wajib memerlukan pengujian mendalam, akan tetapi aplikasi yang telah masuk ke tahap *release* sudah tidak perlu melakukan pengujian apapun.

1.5.2. *Deployment Definition*

Pada tahapan pengembangan sistem, implementasi sistem merupakan tahapan akhir dalam pembangunan sistem. Pada tahapan implementasi, perpindahan dari sistem lama ke sistem baru dibagi menjadi 3 macam yaitu *direct* (secara langsung), *parallel* (setengah-setengah) dan juga *phased* (pergantian ke sistem baru dibuat per modul sistem).

Deployment dalam *software* dan web berarti mendorong perubahan atau pembaruan dari satu lingkungan penyebaran ke lingkungan penyebaran lainnya. Saat menyiapkan situs web, kita akan selalu memiliki situs web langsung, yang disebut lingkungan langsung atau lingkungan produksi. *Deployment* adalah kegiatan yang bertujuan untuk menyebarkan aplikasi yang telah dikerjakan oleh para pengembang. Penyebarannya dapat melalui beragam cara tergantung dari jenis aplikasinya.

Deployment Activites meliputi melakukan tes sistem dan stres, melakukan tes penerimaan, mengonversi data yang ada, membangun materi pelatihan/melakukan pelatihan, mengonfigurasi dan mengatur lingkungan produksi, dan menyebarkan solusi. *Testing* adalah aktivitas utama implementasi dan penyebaran dan mencakup pengujian unit, uji integrasi, tes kegunaan, tes sistem/kinerja/stres, dan tes penerimaan.

Development plan program adalah *trade-off* (perdagangan) di antara sumber daya yang tersedia, waktu yang tersedia, dan keinginan untuk mendeteksi dan memperbaiki kesalahan sebelum penyebaran sistem.

DAFTAR PUSTAKA

Bear, S. (2022). *Negative Testing*. Retrieved from smartbear.com: https://smartbear.com/learn/automated-testing/negative-testing/

Berry, P. (2021, Agustus 20). *Apa Perbedaan Antara Build dan Release dalam Pengujian Perangkat Lunak*. Retrieved from Strephonsays: https://id.strephonsays.com/what-is-the-difference-between-build-and-release-in-software-testing

Geeks, G. F. (2021, November 24). *Equivalence Partitioning Method*. Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/equivalence-partitioning-method/

Geeks, G. F. (2022, Maret 27). *Software Testing - Boundary Value Analysis*. Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/software-testing-boundary-value-analysis/

Quipper. (2022). *Quality Assurance (QA) dan Quality Control (QC)*. Retrieved from quipper.com: https://campus.quipper.com/careers/quality-assurance-control

Synoptek. (2021, September 21). *Type of QA Testing*. Retrieved from synoptek.com: https://synoptek.com/insights/it-blogs/types-of-qa-testing-everything-you-need-to-know/

University, B. (2021, November 26). *Deployment Approaches (Direct, Parallel, Phased)*. Retrieved from sis.binus.ac.id: https://sis.binus.ac.id/2021/11/26/deployment-approaches-direct-parallel-phased/

University, S. (2022, Maret 12). *Memahami Quality Assurance*. Retrieved from sampoernauniversity.ac.id: https://www.sampoernauniversity.ac.id/id/quality-assurance-pengertian-peran-dan-tanggung-jawab