1. *Quality assurance* (*QA*) adalah proses sistematis untuk memastikan bahwa perangkat lunak/ produk telah sesuai dengan standar yang ditentukan oleh perusahaan, serta sesuai dengan kebutuhan user. *QA* berfokus pada pengujian dan evaluasi proses pengembangan perangkat lunak untuk mencegah tejadinya *bugs/error* pada produk. Sehingga produk siap digunakan oleh *end user* tanpa adanya masalah yang signifikan. Sedangkan peran utama seorang *QA Engineer* adalah:

    - Memahami kebutuhan pengguna, dan memastikan bahwa perangkat lunak sesuai dengan standar yang telah ditetapkan
    - Menyusun *test plan* dan *test cases*
    - Melakukan pengujian manual atau otomatis berdasarkan *test case* yang telah disusun
    - Melaporkan *bug* atau *error* yang ditemukan selama pengujian dengan membuat *bug report*

2. Pengujian Fungsional: pengujian untuk memastikan bahwa fitur atau fungsi yang dikembangkan sesuai dengan spesifikasi atau kebutuhan. Tujuannya adalah untuk memastikan fitur bekerja sesuai dengan yang diharapkan. Pengujian ini biasanya dilakukan berdasarkan *test case* yang telah dirancang. Contohnya adalah verifikasi f*orm login*, seperti memvalidasi input *username* dan *password*.

    Pengujian Non fungsional: pengujian untuk mengevaluasi aspek-aspek kualitas perangkat lunak seperti kinerja, keamanan, keandalan, dan tampilan antarmuka pengguna (*UI*). Tujuannya adalah untuk memastikan perangkat lunak memenuhi harapan dalam hal performa, pengalaman pengguna, dan memastikan perangkat lunak aman dari potensi ancaman atau serangan. Contohnya adalah melakukan uji kecepatan aplikasi untuk mengukur waktu respons aplikasi saat diakses oleh beberapa *user* secara bersamaan.

3. Cara saya untuk merencanakan dan melakukan pengujian manual untuk aplikasi web e-commerce adalah sebagai berikut:

    1. Memahami kebutuhan dan tujuan pengujian dengan mempelajari dokumen spesifikasi atau kebutuhan pengguna yang diberikan
    2. Menyusun rencana pengujian (*test plan*) dengan menentukan modul atau fitur utama yang harus diuji, seperti halaman pendaftaran, halaman pembayaran, dan halaman riwayat pesanan, juga menentukan lingkup pengujian (fungsional atau mencakup non-fungsional)
    3. Menyusun *test case* berdasarkan modul yang telah ditetapkan. Pembuatan *test case* lebih terstruktur jika mengelompokkan *test case* berdasarkan masing-masing modulnya
    4. Melakukan pengujian berdasarkan *test case* yang telah dibuat. Pengujian dapat dilakukan dengan memprioritaskan fitur-fitur utama atau fitur yang berdampak langsung pada pengalaman pengguna.

5. Membuat dokumentasi hasil pengujian. Jika terdapat *error/bug*, maka akan membuat *bug report*, termsuk langkah-langkah reproduksi, hasil yang diharapkan, dan hasil aktual, serta tambahkan lampiran bukti (opsional). Dokumentasi tersebut, selanjutnya akan dikirim/ diinformasikan kepada tim pengembang untuk dilakukan evaluasi.
6. Setelah tim pengembang selesai memperbaiki bug, QA dapat melakukan pengujian ulang (*regression testing*).

4. *Regression testing* atau uji regresi adalah salah satu jenis pengujian perangkat lunak yang dilakukan setelah adanya perubahan dalam kode. *Regression testing* bertujuan untuk memastikan bahwa perubahan dalam kode seperti penambahan fitur baru, perbaikan *bug*, atau pembaruan sistem tidak menyebabkan gangguan/ kerusakan pada bagian lain yang sudah ada sebelumnya.

   Alasan pentingnya *regression testing*:

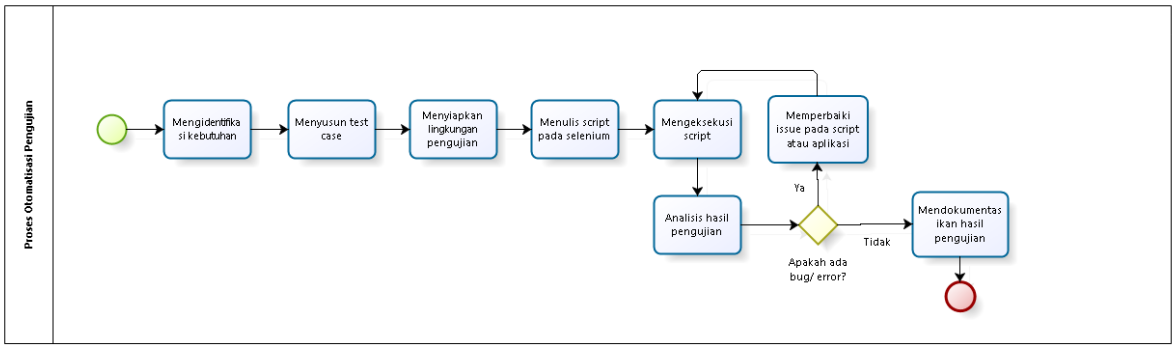
   - Mengurangi resiko kerusakan bagian lain setelah perubahan kode
   - Menjamin kepuasan *user* dengan memastikan bahwa tidak ada bagian produk yang malfungsi
   - Mengidentifikasi *bug* tersembunyi yang mungkin muncul dari perubahan kode

5. Pengujian otomatis: pengujian yang dilakukan dengan menggunakan alat otomatisasi seperti selenium atau katalon untuk menjalankan pengujian secara terprogram. Pengujian ini biasanya menggunakan skrip untuk mengeksekusi skenario pengujian.

   Pengujian manual: pengujian yang dilakukan oleh tester secara langsung tanpa menggunakan alat otomatisasi. Tester berinteraksi secara langsung dengan aplikasi untuk mengevaluasi fitur, tampilan, atau fungsi berdasarkan test case yang sudah dibuat.

   Pengujian otomatis dapat dilakukan ketika pengujian fitur yang terus menerus diuji pada setiap pembaruan aplikasi, sehingga dapat menghemat waktu dan tenaga. Namun, pengujian manual tetap diperlukan untuk mengevaluasi interaksi pengguna (*UI/UX*), dan untuk menemukan *bug* yang tidak terduga.

6. Berikut merupakan gambaran proses saat mengotomatisasi pengujian menggunakan selenium:



7. Kerangka kerja pengujian atau *testing framework* adalah serangkaian pedoman, aturan atau alat yang digunakan untuk membantu pengembangan dan merancang skenario pengujian perangkat lunak secara otomatis, terstruktur, dan efisien. Pedoman ini dapat mencakup standar pengkodean, metode penanganan data uji, mengatur eksekusi pengujian, menulis skrip pengujian, dan menghasilkan laporan hasil pengujian. *Testing framework* dapat membantu dalam:

    - Meningkatkan efisiensi dan produktivitas, karena dapat membuat proses pengujian lebih cepat dengan mengintegrasikan alat seperti laporan otomatis, dan pengumpulan data
    - Standarisasi pengujian, karena semua pengujian diatur dalam format yang konsisten, sehingga mudah dimengerti oleh orang lain
    - Mengurangi pekerjaan manual untuk melakukan pengujian skenario secara langsung, seperti memasukkan data, dan menekan tombol
    - Cakupan pengujian maksimum, dengan mempermudah pengujian berbagai aspek aplikasi, terutama dalam menguji berbagai kombinasi data dan skenario yang kompleks
    - Meningkatkan akurasi hasil pengujian, karena mengurangi kemungkinan *human error*

8. *Bug tracking system* adalah sistem atau perangkat lunak yang digunakan untuk mencatat, melacak, dan mengelola bug atau masalah yang ditemukan selama pengembangan perangkat lunak. Sistem ini dapat membantu tim pengembang untuk memantau status bug, priorotas penyelesaian, dan menganalisa laporan kesalahan (*bug report*). Contoh alat dari *Bug tracking system* adalah seperti JIRA, Trello, Github, dan Bugzilla.

9. Skenario uji fungsional aplikasi pemesanan tiket pesawat online

1.	Melakukan proses login/registrasi pada aplikasi pemesanan tiket
2.	Memilih menu pemesanan tiket pesawat (sistem menampilkan halaman pemesanan pesawat)
3.	Mencari penerbangan yang sesuai (masukkan data-data penerbangan, seperti tempat asal dan tujuan, tanggal keberangkatan, jumlah penumpang, dll)
4.	Memilih penerbangan (sistem menampilkan halaman detail penerbangan yang dipilih)
5.	Mengisi data penumpang dengan valid (data penumpang tersimpan, dan diarahkan ke halaman pembayaran)
6.	Melakukan pembayaran (sistem menampilkan status pembayaran (gagal/berhasil), dan jika berhasil, maka tiket penerbangan dikirim melalui email)
7.	Mengecek tiket pesawat yang sudah dipesan (tiket menampilkan informasi seperti tanggal penerbangan, tanggal keberangkatan, tujuan pergi, dll)

10. Berikut merupakan beberapa uji keamanan yang akan saya lakukan:

- Uji autentikasi: memastikan bahwa hanya user yang valid yang dapat mengakses aplikasi (*username* dan *password* valid, dan telah terdaftar di *database*)
- Uji hak akses *user*: memastikan hanya *user* tertentu yang dapat mengakses fitur-fitur admin. Sehingga, *user* selain admin tidak dapat mengakses fitur tersebut
- Uji enkripsi data: memastikan semua data yang dikirimkan antara klien dan server terenkripsi saat disimpan di *database*, seperti kata sandi, atau nomor rekening
- Uji keamanan kata sandi: memastikan kata sandi *user* memenuhi standar keamanan (kata sandi dengan minimum karakter, menggunakan huruf kapital dan kecil, serta menggunakan simbol)
- Uji manajemen sesi: memastikan sesi pengguna berakhir setelah user telah logout, dan harus login kembali jika ingin mengakses aplikasi

11. Berikut merupakan test case terkait update fitur pesanan saya pada aplikasi mobile shopee:

a. Edit pembayaran, jika pesanan belum dibayar

ID:	EP-001
Modul:	Edit pembayaran
Pre-condition:	-user sudah memiliki akun shopee yang aktif -user memiliki pesanan dengan status “belum bayar”

Scenario	Priority type	Severity type	Action	Data test	Variable test	Expected result	Actual result
Verifikasi user dapat mengedit metode pembayaran, jika user belum membayar pesanan	High	High	Klik tombol “Edit Pembayaran” di Tab belum bayar	Status pesanan	Belum melakukan pembayaran	Tombol “Edit pembayaran” muncul di Tab belum bayar, dan user dapat mengakses halaman untuk memilih metode pembayaran yang baru	Tombol “Edit pembayaran” muncul di Tab belum bayar, dan user dapat mengakses halaman untuk memilih metode pembayaran yang baru

b. Edit pembayaran, jika pembayaran pesanan sebelumnya gagal

ID:	EP-002
Modul:	Edit pembayaran
Pre-condition:	-user sudah memiliki akun shopee yang aktif -user memiliki pesanan dengan status “gagal-bayar”

Scenario	Priority type	Severity type	Action	Data test	Variable test	Expected result	Actual result
Verifikasi user dapat mengedit metode pembayaran, jika user memiliki pesanan dengan status “gagal-bayar”	High	High	Klik tombol “Edit Pembayaran” di Tab belum bayar	Status peembayaran	Gagal saat pembayaran	Tombol “Edit pembayaran” muncul di Tab belum bayar, dan user dapat mengakses halaman untuk memilih metode pembayaran yang baru	Tombol “Edit pembayaran” muncul di Tab belum bayar, dan user dapat mengakses halaman untuk memilih metode pembayaran yang baru