Adnan Hadiyana

1. **Quality Assurance (QA)** adalah proses sistematis yang digunakan untuk memastikan bahwa produk perangkat lunak memenuhi standar kualitas tertentu dan memenuhi persyaratan yang ditetapkan. QA berfokus pada pencegahan cacat dan memastikan bahwa proses pengembangan dan produk akhir berkualitas tinggi.
   Peran Utama Seorang QA Engineer dalam Siklus Pengembangan Perangkat Lunak
   Pengembangan Rencana Pengujian:QA Engineer bertanggung jawab untuk merencanakan dan mendokumentasikan rencana pengujian, yang mencakup ruang lingkup pengujian, jenis pengujian yang akan dilakukan, dan strategi pelaksanaan pengujian.
   Analisis Persyaratan:Memahami dan menganalisis dokumen spesifikasi dan persyaratan untuk merumuskan kasus uji yang sesuai. QA Engineer harus memastikan bahwa semua fungsionalitas yang diinginkan telah ditangkap dan dapat diuji.
   Pembuatan Kasus Uji:Menulis dan mendokumentasikan kasus uji berdasarkan persyaratan fungsional dan non-fungsional. Kasus uji ini dirancang untuk memverifikasi bahwa sistem berfungsi seperti yang diharapkan.
   Pelaksanaan Pengujian:Melakukan pengujian manual dan otomatis untuk memverifikasi bahwa produk perangkat lunak memenuhi spesifikasi. Ini termasuk pengujian fungsional, pengujian regresi, pengujian keamanan, dan pengujian kinerja.
   Identifikasi dan Pelaporan Bug:Mendeteksi bug dan masalah dalam perangkat lunak, mencatat detailnya, dan melaporkannya kepada tim pengembang. QA Engineer harus memberikan informasi yang cukup untuk memudahkan pengembang dalam memperbaiki masalah.
   Retesting dan Verifikasi:Setelah pengembang memperbaiki bug, QA Engineer melakukan retesting untuk memastikan bahwa masalah telah diperbaiki dan tidak ada fungsi lain yang terpengaruh.
   Pengujian Regresi:Melakukan pengujian regresi untuk memastikan bahwa perubahan atau perbaikan yang dilakukan tidak merusak fungsionalitas yang sudah ada.
   Kolaborasi dengan Tim:Bekerja sama dengan tim pengembang, manajer produk, dan pemangku kepentingan lainnya untuk memastikan semua pihak memahami status pengujian dan kualitas produk.
   Peningkatan Proses:Terlibat dalam analisis proses dan memberikan umpan balik untuk meningkatkan kualitas pengembangan dan pengujian, serta untuk mencegah masalah di masa depan.
   Dokumentasi dan Pelaporan:Membuat laporan pengujian yang merangkum hasil, termasuk jumlah kasus uji yang berhasil dan gagal, serta analisis masalah. Dokumentasi ini penting untuk pengambilan keputusan di masa depan.

2. Pengujian Fungsional: Pengujian fungsional adalah jenis pengujian yang dilakukan untuk memverifikasi bahwa aplikasi atau sistem berfungsi sesuai dengan spesifikasi dan persyaratan yang ditentukan. Fokus utamanya adalah pada "apa" yang dilakukan oleh aplikasi.
   Pengujian Non-Fungsional: Pengujian non-fungsional adalah jenis pengujian yang dilakukan untuk mengevaluasi aspek lain dari aplikasi yang tidak terkait langsung dengan fungsionalitas, seperti kinerja, keamanan, dan pengalaman pengguna. Fokus utamanya adalah pada "bagaimana" aplikasi berfungsi.

3. Berikut adalah langkah-langkah utama dalam proses ini:

Perencanaan Pengujian

Identifikasi Tujuan Pengujian:Tentukan tujuan pengujian, seperti memastikan fungsionalitas dasar, keamanan, dan pengalaman pengguna.

Analisis Persyaratan:Tinjau dokumen persyaratan untuk memahami fitur dan fungsionalitas aplikasi e-commerce. Ini mencakup pengelolaan produk, keranjang belanja, pembayaran, dan pengelolaan pengguna.

Buat Rencana Pengujian:Susun rencana pengujian yang mencakup ruang lingkup pengujian, strategi, dan sumber daya yang diperlukan. Tentukan jenis pengujian yang akan dilakukan (fungsional, keamanan, usability, dll.).

Desain Kasus Uji

Identifikasi Kasus Uji:

Buat daftar kasus uji berdasarkan persyaratan fungsional dan non-fungsional. Kasus uji dapat mencakup:Pencarian produk,Menambah dan menghapus item dari keranjang,Proses checkout,Pembayaran,Pendaftaran dan login pengguna dan Pengelolaan akun pengguna

Tulis Kasus Uji:

Dokumentasikan kasus uji dengan detail, termasuk:Tujuan kasus uji, Langkah-langkah yang harus diikuti, Data uji yang diperlukan, Hasil yang diharapkan

Persiapan Lingkungan Pengujian

Siapkan Lingkungan:Pastikan lingkungan pengujian sudah disiapkan, termasuk server, database, dan akses ke aplikasi.

Persiapkan Data Uji:Buat data uji yang diperlukan untuk menjalankan kasus uji, termasuk akun pengguna, produk, dan detail pembayaran.

Eksekusi Pengujian

Jalankan Kasus Uji:Lakukan pengujian sesuai dengan kasus uji yang telah dibuat. Catat langkah-langkah yang diambil dan hasilnya.

Catat Hasil:Dokumentasikan hasil pengujian, mencatat apakah setiap langkah berhasil atau gagal. Jika ada bug atau masalah, beri detail dan langkah-langkah untuk mereproduksinya.

Pelaporan dan Analisis

Buat Laporan Pengujian:Buat laporan pengujian yang merangkum hasil, termasuk jumlah kasus uji yang berhasil, gagal, dan status bug yang ditemukan.

Analisis Masalah:Tinjau bug dan masalah yang ditemukan. Diskusikan dengan tim pengembangan untuk menentukan penyebab dan tindakan perbaikan.

Retesting dan Regression Testing

Retest:Setelah pengembang memperbaiki bug, lakukan pengujian ulang pada kasus uji yang terpengaruh untuk memastikan masalah telah diperbaiki.

Uji Regresi:Jalankan pengujian regresi untuk memastikan bahwa perubahan yang dilakukan tidak memengaruhi fungsionalitas lain dalam aplikasi.

Penutupan Pengujian

Evaluasi Proses:Tinjau proses pengujian dan hasilnya. Identifikasi apa yang berhasil dan apa yang perlu ditingkatkan di masa mendatang.

Dokumentasi Akhir:Pastikan semua dokumentasi pengujian, termasuk laporan, catatan kasus uji, dan data uji, disimpan dengan baik untuk referensi di masa depan.

Umpan Balik dan Iterasi

Dapatkan Umpan Balik:Dapatkan umpan balik dari tim pengembangan dan pemangku kepentingan lainnya tentang hasil pengujian dan perbaikan yang perlu dilakukan.
Rencanakan Pengujian Berikutnya:Berdasarkan hasil dan umpan balik, rencanakan siklus pengujian berikutnya untuk fitur baru atau perbaikan yang direncanakan.

4. Uji regresi adalah jenis pengujian perangkat lunak yang dilakukan untuk memastikan bahwa perubahan atau modifikasi dalam kode (seperti perbaikan bug, penambahan fitur, atau perubahan lainnya) tidak merusak fungsionalitas yang sudah ada. Tujuan utama dari uji regresi adalah untuk memverifikasi bahwa perangkat lunak masih berfungsi seperti yang diharapkan setelah dilakukan perubahan.
Mengapa Uji Regresi Penting?
Menjamin Stabilitas Perangkat Lunak:Uji regresi membantu memastikan bahwa fungsionalitas yang ada tetap stabil setelah adanya perubahan. Hal ini penting untuk menjaga kualitas perangkat lunak.
Mendeteksi Bug Baru:Setiap kali ada perubahan dalam kode, ada risiko bahwa bug baru dapat muncul, termasuk yang mungkin tidak terkait dengan perubahan tersebut. Uji regresi dapat membantu mendeteksi bug baru yang mungkin tidak terduga.

5. Berikut adalah aliran kerja atau proses yang umum diikuti untuk mengotomatisasi pengujian:
1. Perencanaan dan Persiapan
Identifikasi Kasus Uji: Tentukan fitur dan fungsi aplikasi yang akan diuji secara otomatis. Pilih kasus uji yang sering digunakan dan memiliki nilai bisnis tinggi.
Analisis Kelayakan: Evaluasi apakah kasus uji tersebut cocok untuk otomatisasi (misalnya, stabil, repetitif, dan tidak terlalu sering berubah).
2. Persiapan Lingkungan Pengujian
Instalasi Alat: Instal Selenium dan semua alat pendukung yang diperlukan (seperti WebDriver, IDE, pustaka tambahan).
Konfigurasi Lingkungan: Siapkan lingkungan pengujian, termasuk browser yang akan digunakan (Chrome, Firefox, dll.), dan pastikan semua dependensi terpasang dengan benar.
3. Pengembangan Skrip Pengujian
Pengkodean Skrip: Tulis skrip pengujian menggunakan bahasa pemrograman yang didukung (seperti Java, Python, C#, dll.).
Gunakan Selenium WebDriver untuk berinteraksi dengan elemen-elemen di halaman web (klik, masukkan teks, ambil nilai, dll.).
Penerapan Desain Pengujian: Gunakan pola desain seperti Page Object Model (POM) untuk mengorganisir kode dan memisahkan logika pengujian dari detail implementasi.
4. Pengujian Skrip
Pengujian Lokal: Jalankan skrip pengujian secara lokal untuk memastikan bahwa mereka berfungsi dengan baik dan tidak ada kesalahan.
Debugging: Perbaiki masalah atau bug yang muncul selama pengujian lokal.
5. Integrasi dengan CI/CD
Pengaturan Alat CI/CD: Integrasikan skrip pengujian dengan sistem CI/CD (seperti Jenkins, GitLab CI, atau CircleCI) untuk menjalankan pengujian otomatis setiap kali ada perubahan kode.
Konfigurasi Pipeline: Buat pipeline yang menjalankan pengujian secara otomatis pada commit baru atau pull request.

6. Eksekusi Pengujian
Jadwalkan Pengujian: Jalankan pengujian secara terjadwal atau saat ada pemicu tertentu, seperti build baru.
Monitor Hasil: Pantau eksekusi pengujian dan catat hasilnya. Sistem CI/CD biasanya memberikan laporan tentang hasil pengujian.
7. Analisis Hasil
Tinjau Laporan Pengujian: Periksa laporan pengujian untuk melihat hasil, seperti jumlah tes yang berhasil, gagal, atau diabaikan.
Identifikasi Masalah: Analisis kegagalan pengujian untuk menentukan apakah ada bug di aplikasi atau masalah dengan skrip pengujian itu sendiri.
8. Pemeliharaan Skrip
Pembaruan Skrip: Sesuaikan skrip pengujian sesuai dengan perubahan pada aplikasi (misalnya, perubahan UI atau fungsionalitas).
Refactoring: Secara berkala refactor kode pengujian untuk meningkatkan kualitas dan keterbacaan.
9. Dokumentasi
Dokumentasikan Proses: Buat dokumentasi untuk setiap langkah, termasuk pengaturan lingkungan, struktur skrip, dan hasil pengujian.
Pelatihan Tim: Jika perlu, latih anggota tim lain untuk memahami dan menggunakan kerangka kerja pengujian yang telah dibuat.
10. Ulangi Proses:Iterasi Berkelanjutan:
Ulangi proses ini untuk menambah lebih banyak kasus uji dan meningkatkan cakupan pengujian seiring perkembangan aplikasi.

6. Kerangka kerja pengujian (testing framework) dalam pengujian otomatis adalah suatu struktur yang menyediakan dasar dan alat untuk merancang, mengembangkan, dan menjalankan pengujian otomatis. Kerangka kerja ini menyatukan berbagai komponen pengujian, termasuk alat, pustaka, dan prosedur, untuk memfasilitasi proses pengujian yang lebih sistematis dan efisien.
Komponen Umum dalam Kerangka Kerja Pengujian:
Aturan dan Standar: Definisi tentang bagaimana pengujian harus dilakukan, termasuk penamaan file, pengorganisasian kode, dan praktik terbaik lainnya.
Alat Pengujian: Penggunaan alat otomatis seperti Selenium, JUnit, TestNG, atau pytest, yang digunakan untuk menjalankan pengujian.
Pustaka dan Fungsi Ulang: Kode dan fungsi yang dapat digunakan kembali untuk mengurangi duplikasi dan meningkatkan efisiensi dalam penulisan pengujian.
Laporan dan Analisis: Mekanisme untuk menghasilkan laporan pengujian dan analisis hasil, termasuk ringkasan kesalahan dan statistik keberhasilan.
Integrasi dengan CI/CD: Kemampuan untuk mengintegrasikan pengujian otomatis dalam alur kerja Continuous Integration/Continuous Deployment, sehingga pengujian dapat dijalankan secara otomatis pada setiap perubahan kode.

7. Kerangka kerja pengujian (testing framework) dalam pengujian otomatis adalah sekumpulan alat, pustaka, dan aturan yang memberikan struktur dan pendekatan standar untuk menulis, mengelola, dan menjalankan pengujian otomatis pada perangkat lunak.

8. Bug tracking system" adalah alat atau perangkat lunak yang digunakan untuk melacak, mengelola, dan mendokumentasikan bug atau kesalahan dalam perangkat lunak. Sistem ini membantu tim pengembang dan penguji dalam mengidentifikasi masalah,

menetapkan prioritas, dan memastikan bahwa bug diperbaiki sebelum produk dirilis.contoh nya JIRA

9. **Pencarian Penerbangan**
   Tujuan: Memastikan pengguna dapat mencari penerbangan yang tersedia.
   Langkah:
   Buka aplikasi pemesanan tiket pesawat.
   Masukkan bandara keberangkatan dan kedatangan.
   Pilih tanggal keberangkatan dan kembali (jika berlaku).
   Klik tombol "Cari".
   Ekspektasi:
   Aplikasi menampilkan daftar penerbangan yang sesuai dengan kriteria pencarian.
   Setiap penerbangan menunjukkan informasi seperti waktu keberangkatan, waktu kedatangan, durasi penerbangan, dan harga.
10. Menguji keamanan aplikasi perbankan online adalah langkah kritis untuk melindungi data sensitif pengguna dan mencegah potensi serangan. Berikut adalah beberapa contoh uji keamanan yang dapat dilakukan:
    Uji Penetrasi (Penetration Testing)
    Tujuan: Mengidentifikasi kerentanan yang dapat dimanfaatkan oleh penyerang.
    Langkah: Melakukan simulasi serangan dengan mencoba mengeksploitasi kerentanan yang mungkin ada, seperti: SQL Injection: Menguji input pengguna untuk melihat apakah aplikasi rentan terhadap injeksi SQL.

11. Test Case: Ubah Pembayaran pada Fitur Pesanan Saya

Test Case ID**: TC_UbahPembayaran_001**
Deskripsi**: Menguji fungsi 'Ubah Pembayaran' pada tab "Belum Bayar" untuk memastikan tombol muncul sesuai dengan kondisi yang ditetapkan.**
Prasyarat**: Pengguna telah masuk ke aplikasi dan memiliki pesanan yang memenuhi syarat.**

Test Variables

- **Order Status**:
  - Belum Dibayar
  - Gagal Bayar
  - Sudah Dibayar
- **Tombol Ubah Pembayaran**:
  - Tersedia
  - Tidak Tersedia

Test Data

No	Order Status	Jenis Pembayaran	Status Pembayaran	Tombol Ubah Pembayaran
1	Belum Dibayar	Kartu Kredit	Belum Bayar	Tersedia
2	Gagal Bayar	E-Wallet	Gagal-Bayar	Tersedia

No	Order Status	Jenis Pembayaran	Status Pembayaran	Tombol Ubah Pembayaran
3	Belum Dibayar	Transfer Bank	Belum Bayar	Tersedia
4	Sudah Dibayar	Kartu Kredit	Sukses	Tidak Tersedia
5	Gagal Bayar	Kartu Kredit	Gagal-Bayar	Tersedia
6	Sudah Dibayar	E-Wallet	Sukses	Tidak Tersedia

Langkah Pengujian

1. **Setup**: Pastikan pengguna sudah masuk ke aplikasi dan berada di tab "Pesanan Saya" - "Belum Bayar".
2. **Uji Kasus 1**:
    - **Input**: Pilih order dengan status Belum Dibayar.
    - **Ekspektasi**: Tombol Ubah Pembayaran ditampilkan.
3. **Uji Kasus 2**:
    - **Input**: Pilih order dengan status Gagal Bayar.
    - **Ekspektasi**: Tombol Ubah Pembayaran ditampilkan.
4. **Uji Kasus 3**:
    - **Input**: Pilih order dengan status Sudah Dibayar.
    - **Ekspektasi**: Tombol Ubah Pembayaran tidak ditampilkan.
5. **Uji Kasus 4**:
    - **Input**: Pilih order dengan status Gagal Bayar tetapi tidak diizinkan untuk Belum Dibayar.
    - **Ekspektasi**: Tombol Ubah Pembayaran ditampilkan.

Hasil yang Diharapkan

- Untuk order dengan status Belum Dibayar, tombol Ubah Pembayaran harus muncul.
- Untuk order dengan status Gagal Bayar, tombol Ubah Pembayaran harus muncul.
- Untuk order dengan status Sudah Dibayar, tombol Ubah Pembayaran tidak boleh muncul.