3 Alasan Dunia Tidak Butuh Software Testing

KAPITA SELEKTA

Disusun Oleh

1204037 Muhammad Sapwan Suhadi

Program Studi Diploma IV Teknik Informatika

Universitas Logistik Dan Bisnis Internasional

2023

1. Rangkuman

Dunia tidak menggunakan software testing karena:

1. Alasan waktu dan biaya pengembangan yang menjadi lebih lama akibat proses pengujian yang memakan waktu dan biaya tambahan.
2. Keterbatasan sumber daya yang dimiliki, seperti keterbatasan SDM atau peralatan pengujian yang tidak memadai.
3. Keyakinan berlebihan pada pengalaman proyek sebelumnya yang membuat tim merasa tidak perlu melakukan pengujian secara terstruktur dan sistematis.

Namun, penggunaan software testing sangat penting untuk memastikan kualitas dan keamanan perangkat lunak yang dihasilkan serta menghindari kerugian yang lebih besar di kemudian hari. Oleh karena itu, tim pengembang sebaiknya mempertimbangkan dan mengalokasikan waktu dan sumber daya yang memadai untuk proses pengujian perangkat lunak.

Penggunaan software testing sangat penting dalam pengembangan perangkat lunak karena dapat membantu menemukan bug atau masalah yang mungkin terjadi pada perangkat lunak sebelum diperkenalkan ke pasar atau digunakan oleh pengguna. Namun, terkadang tim pengembang tidak menggunakan software testing karena beberapa alasan.

Salah satu alasan yang sering digunakan adalah karena pengujian dapat memakan waktu dan biaya tambahan. Tim pengembang terkadang memiliki tekanan untuk menyelesaikan proyek dalam waktu yang singkat dengan biaya yang minimal. Proses pengujian yang memakan waktu dan biaya tambahan dapat menjadi penghalang dalam mencapai tujuan tersebut.

Selain itu, keterbatasan sumber daya seperti keterbatasan SDM atau peralatan pengujian yang tidak memadai juga dapat menjadi alasan tidak digunakannya software testing. Pengembangan perangkat lunak membutuhkan sumber daya yang memadai untuk dapat menghasilkan produk yang berkualitas. Namun, terkadang keterbatasan sumber daya menjadi penghalang dalam melakukan proses pengujian secara terstruktur dan sistematis.

Selain faktor waktu dan sumber daya, keyakinan berlebihan pada pengalaman proyek sebelumnya juga dapat menjadi alasan tim pengembang tidak menggunakan software testing. Terkadang tim pengembang merasa bahwa karena telah berhasil menyelesaikan proyek sebelumnya tanpa menggunakan software testing, maka tidak perlu lagi melakukannya pada proyek yang baru.

Namun, tidak menggunakan software testing dapat menyebabkan risiko kegagalan pada perangkat lunak, yang pada akhirnya dapat merugikan pengguna atau perusahaan. Perangkat lunak yang tidak diuji dengan baik dapat menghasilkan bug dan masalah yang tidak terdeteksi, yang pada akhirnya dapat mengakibatkan kerugian yang lebih besar.

Oleh karena itu, sebaiknya tim pengembang mempertimbangkan dan mengalokasikan waktu dan sumber daya yang memadai untuk proses pengujian perangkat lunak. Dengan melakukan software testing yang baik dan terstruktur, maka dapat membantu memastikan kualitas dan keamanan perangkat lunak yang dihasilkan.

2. Pemahaman 7 prinsip testing, Functional testing, Non Functional testing, Structural testing dan Testing related to Change:

1. Prinsip-prinsip Testing:
    a. Testing harus dimulai sedini mungkin dalam siklus pengembangan perangkat lunak.
    b. Tidak mungkin menguji keseluruhan perangkat lunak, jadi pengujian harus didasarkan pada risiko.
    c. Pengujian harus mengikuti rencana yang terdokumentasi dengan baik dan sesuai dengan kebutuhan bisnis.
    d. Pengujian harus menyeluruh dan mencakup berbagai jenis pengujian.
    e. Pengujian harus dilakukan oleh orang yang berbeda dari yang membuat perangkat lunak.
    f. Hasil pengujian harus didokumentasikan dan dikomunikasikan dengan baik.
2. Functional Testing:
    a. Pengujian fungsional harus mencakup semua fitur dan fungsionalitas yang dijelaskan dalam spesifikasi perangkat lunak.
    b. Pengujian fungsional harus mengikuti skenario penggunaan yang realistis dan mencakup pengujian positif dan negatif.
    c. Pengujian fungsional harus dilakukan dengan menggunakan data yang valid dan relevan.
    d. Hasil pengujian fungsional harus dianalisis dengan seksama dan segala cacat harus didokumentasikan dengan jelas.
3. Non-Functional Testing:
    a. Pengujian non-fungsional mencakup semua aspek non-fungsional perangkat lunak, seperti keamanan, kinerja, dan skalabilitas.
    b. Pengujian non-fungsional harus mengikuti persyaratan yang jelas dan terukur.
    c. Pengujian non-fungsional harus dilakukan menggunakan skenario yang realistis dan dengan beban yang sesuai.
    d. Hasil pengujian non-fungsional harus dianalisis dengan seksama dan segala cacat harus didokumentasikan dengan jelas.
4. Structural Testing:
    a. Pengujian struktural harus mencakup semua bagian kode sumber yang dapat diuji.
    b. Pengujian struktural harus dilakukan dengan menggunakan teknik pengujian yang sesuai, seperti pengujian path dan pengujian mutasi.
    c. Hasil pengujian struktural harus dianalisis dengan seksama dan segala cacat harus didokumentasikan dengan jelas.
5. Testing Related to Change:
    a. Pengujian yang terkait dengan perubahan harus dilakukan setiap kali ada perubahan pada perangkat lunak.
    b. Pengujian yang terkait dengan perubahan harus mencakup semua bagian perangkat lunak yang terpengaruh oleh perubahan tersebut.
    c. Pengujian yang terkait dengan perubahan harus dilakukan sebelum perubahan diterapkan ke lingkungan produksi.
    d. Hasil pengujian yang terkait dengan perubahan harus dianalisis dengan seksama dan segala cacat harus didokumentasikan dengan jelas.

3. Contoh Functional testing, Non Functional testing, Structural testing dan Testing related to Change

1. Contoh Functional Testing:
    a. Pengujian fungsi login pada aplikasi web untuk memastikan bahwa pengguna dapat masuk ke akun mereka dengan benar.
    b. Pengujian fungsi pembayaran pada aplikasi belanja online untuk memastikan bahwa transaksi dapat diselesaikan dengan sukses.
    c. Pengujian fungsi pencarian pada aplikasi pencarian untuk memastikan bahwa hasil pencarian relevan dan akurat.
2. Contoh Non-Functional Testing:
    a. Pengujian kinerja pada aplikasi e-commerce untuk memastikan bahwa waktu muat halaman cukup cepat dan dapat menangani lalu lintas yang tinggi.
    b. Pengujian keamanan pada aplikasi perbankan untuk memastikan bahwa data sensitif terlindungi dengan baik.
    c. Pengujian skalabilitas pada aplikasi media sosial untuk memastikan bahwa aplikasi dapat menangani peningkatan pengguna yang besar.
3. Contoh Structural Testing:
    a. Pengujian path pada kode sumber untuk memastikan bahwa semua cabang kode telah diuji.
    b. Pengujian mutasi pada kode sumber untuk memastikan bahwa perubahan kecil dalam kode dapat diidentifikasi.
4. Contoh Testing Related to Change:
    a. Pengujian regresi pada aplikasi web setelah perubahan dilakukan untuk memastikan bahwa fitur yang telah diuji sebelumnya masih berfungsi dengan benar.
    b. Pengujian integrasi pada sistem manajemen inventaris setelah penambahan modul baru untuk memastikan bahwa modul tersebut terintegrasi dengan benar.