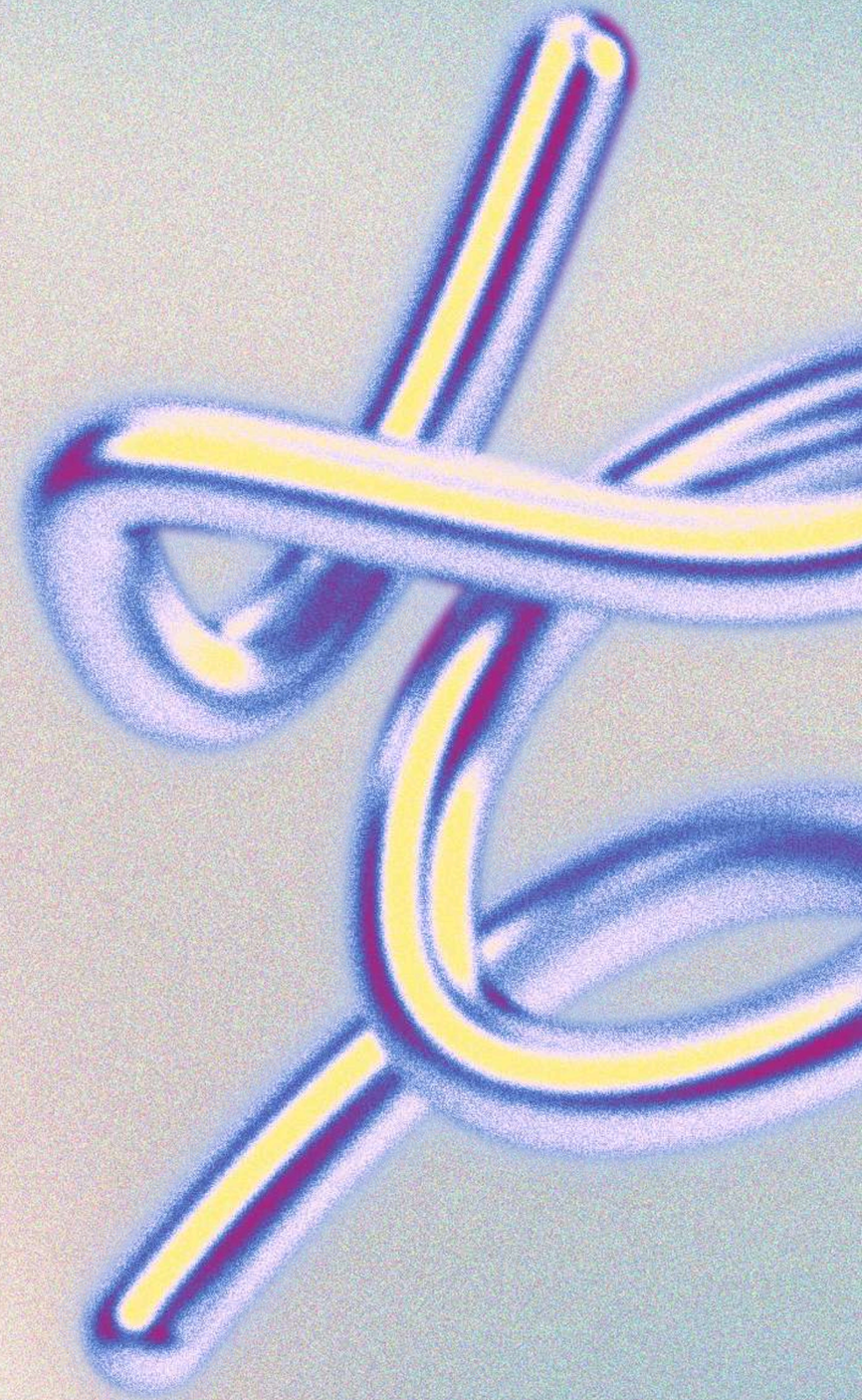
Software Testing and Quality Assurance

Saldan Rama (H071221071)

Functional Testing

Functional Testing adalah jenis pengujian perangkat lunak yang memeriksa apakah aplikasi, situs web, atau sistem berfungsi sesuai dengan yang dimaksudkan. Pengujian ini memastikan bahwa perangkat lunak melakukan fungsi yang diharapkan dan memenuhi kebutuhan pengguna. Fokus utama dari Functional Testing adalah pada kebenaran fungsi aplikasi, bukan pada aspek lain seperti kinerja atau keamanan (BrowserStack, n.d.).



Langkah/Proses Functional Testing

1. Identifikasi Fungsionalitas: Tentukan fungsi mana dari produk yang perlu diuji, termasuk pengujian fungsi utama, pesan, kondisi kesalahan, dan/atau kegunaan produk.
2. Buat Data Input: Siapkan data input untuk fungsionalitas yang akan diuji sesuai dengan persyaratan yang ditentukan.
3. Tentukan Parameter Output: Tentukan parameter output yang dapat diterima sesuai dengan persyaratan yang ditentukan.
4. Eksekusi Kasus Uji: Jalankan kasus uji yang telah disiapkan.
5. Bandingkan Output Aktual dengan Output yang Diharapkan: Periksa apakah output aktual dari pengujian sesuai dengan nilai output yang telah ditentukan. Ini akan menunjukkan apakah sistem berfungsi seperti yang diharapkan (BrowserStack, n.d.).

CONTOH KASUS

Sebuah portal HRMS online di mana pengguna dapat masuk menggunakan akun pengguna dan kata sandi mereka. Halaman login memiliki dua kolom teks untuk username dan password. Selain itu, terdapat dua tombol – Login dan Cancel. Ketika login berhasil, halaman login akan mengarahkan pengguna ke halaman utama HRMS. Tombol Cancel akan membatalkan proses login.

Spesifikasi:

1. **Kolom User ID:**
    a. Harus memiliki minimal 6 karakter dan maksimal 10 karakter.
    b. Hanya diperbolehkan angka (0-9), huruf (a-z, A-Z), dan karakter khusus (underscore, titik, dan hyphen).
    c. Tidak boleh kosong.
    d. Harus dimulai dengan angka atau huruf.
    e. Tidak boleh dimulai atau mengandung karakter khusus selain yang diperbolehkan.
2. Kolom Password:
    a. Harus memiliki minimal 6 karakter dan maksimal 8 karakter.
    b. Dapat berisi angka (0-9), huruf (a-z, A-Z), dan semua jenis karakter khusus.
    c. Tidak boleh kosong.

User Interface Testing

User Interface Testing (UIT) adalah proses evaluasi antarmuka pengguna dari aplikasi atau sistem untuk memastikan bahwa desainnya memenuhi kebutuhan pengguna dan berfungsi dengan baik. UIT bertujuan untuk mengidentifikasi masalah usability yang mungkin dihadapi oleh pengguna saat berinteraksi dengan aplikasi.

Langkah/Proses User Interface Testing

1. Identifikasi Pengguna: Menentukan siapa yang akan menjadi pengguna dalam pengujian.
2. Analisis Kebutuhan Pengguna: Mengumpulkan informasi tentang apa yang dibutuhkan dan diharapkan oleh pengguna.
3. Desain Prototipe: Membuat model antarmuka yang akan diuji.
4. Pengujian Usability: Melakukan pengujian dengan pengguna nyata untuk mengumpulkan data tentang pengalaman mereka.
5. Evaluasi Hasil: Menganalisis data yang dikumpulkan untuk mengidentifikasi masalah dan area perbaikan.
6. Implementasi Perbaikan: Menggunakan umpan balik untuk memperbaiki desain antarmuka.

CONTOH KASUS

Salah satu contoh penerapan UIT adalah penelitian yang dilakukan pada aplikasi E-Anjal menggunakan metode Black Box. Penelitian ini bertujuan untuk menguji kesesuaian antara desain antarmuka dengan model yang telah ditentukan, serta mendeteksi kesalahan dalam rancangan antarmuka.

Dalam studi ini, pengujian dilakukan dengan memasukkan berbagai input ke dalam aplikasi dan memeriksa apakah output yang dihasilkan sesuai dengan harapan. Hasil dari penelitian ini menunjukkan bahwa tidak ada kesalahan signifikan pada rancangan aplikasi, sehingga antarmuka dianggap memenuhi standar desain yang telah ditetapkan.

Contoh lain adalah penggunaan Usability Testing pada website resmi Badan Narkotika Nasional (BNN). Penelitian ini menggunakan pendekatan HCD untuk meningkatkan efektivitas dan efisiensi antarmuka website. Hasil menunjukkan peningkatan usability dari 96% menjadi 100% setelah implementasi perbaikan berdasarkan umpan balik pengguna

Load Testing

Load Testing adalah metode pengujian yang digunakan untuk mengevaluasi kinerja sistem dengan mengukur responnya dibawah berbagai kondisi beban. Tujuan utama dari load testing adalah untuk menentukan seberapa banyak beban yang dapat ditangani oleh sistem sebelum mengalami penurunan kinerja atau kegagalan.

Langkah/Proses Load Testing

1. Perencanaan Load Testing:
    - Tentukan tujuan pengujian dan skenario beban yang akan diuji.
    - Siapkan infrastruktur dan alat yang diperlukan untuk melakukan pengujian
2. Persiapan Load Test:
    - Siapkan data dan skrip uji beban.
    - Atur pengaturan seperti cache browser dan cookie untuk memastikan hasil yang akurat
3. Pelaksanaan Load Test:
    - Jalankan skrip uji beban dan kumpulkan data kinerja aplikasi.
    - Monitor hasil selama pengujian untuk mencatat masalah yang muncul
4. Analisis dan Optimasi:
    - Setelah pengujian selesai, analisis data untuk memahami performa aplikasi.
    - Identifikasi masalah dan lakukan optimasi berdasarkan hasil analisis

CONTOH KASUS

Salah satu contoh penerapan load testing adalah pada sistem informasi pertanian menggunakan Apache JMeter. Dalam penelitian ini, load testing dilakukan untuk mengukur waktu loading dan penggunaan memori proses ketika beberapa pengguna mengakses sistem secara bersamaan. Target pengujian ditetapkan agar waktu loading tidak lebih dari 3 detik dan penggunaan memori tidak melebihi 400MB. Hasil pengujian menunjukkan bahwa kedua target tersebut dapat terpenuhi, sehingga aplikasi dinyatakan siap untuk digunakan dalam lingkungan nyata [3]

Stress Testing

Stress Testing adalah metode pengujian yang dirancang untuk mengevaluasi kinerja sistem atau aplikasi di bawah kondisi ekstrem. Tujuan utamanya adalah untuk menentukan seberapa tangguh sistem ketika dihadapkan pada beban yang melebihi kapasitas normal, seperti lonjakan lalu lintas pengguna atau masalah teknis yang tidak terduga.

Langkah/Proses Stress Testing

1. Definisikan Tujuan dan Kriteria Keberhasilan: Tentukan apa yang ingin diuji dan hasil yang dianggap berhasil, seperti waktu respons maksimum atau tingkat transaksi yang dapat ditangani.
2. Identifikasi Skenario Stress: Pilih jenis situasi ekstrem yang ingin diuji, seperti jumlah pengguna yang sangat besar atau volume data yang tinggi.
3. Persiapkan Lingkungan Uji: Siapkan infrastruktur dan alat yang diperlukan untuk melakukan pengujian.
4. Jalankan Pengujian Awal: Lakukan pengujian dengan skenario stres yang telah dikonfigurasi dan amati perilaku sistem.
5. Analisis Hasil: Setelah pengujian selesai, analisis data untuk mengidentifikasi penurunan kinerja atau masalah lainnya.
6. Optimasi dan Peningkatan: Jika ada masalah, lakukan perbaikan dan optimasi pada sistem.
7. Ulangi Pengujian: Setelah perbaikan dilakukan, ulangi pengujian untuk memastikan bahwa masalah telah teratasi.
8. Dokumentasi dan Pelaporan: Catat semua hasil pengujian dan langkah-langkah yang diambil untuk perbaikan

CONTOH KASUS

Salah satu contoh penerapan stress testing adalah pada situs e-commerce selama periode promosi besar. Dalam kasus ini, tim pengembang melakukan stress testing untuk mensimulasikan lonjakan trafik pengguna saat penjualan berlangsung. Hasil dari pengujian ini membantu tim memahami batas kapasitas aplikasi dan memastikan bahwa sistem dapat menangani beban tinggi tanpa mengalami crash atau penurunan performa

End-to-End Testing

End-to-End Testing (E2E Testing) adalah metode pengujian perangkat lunak yang bertujuan untuk memverifikasi alur kerja sistem dari awal hingga akhir. Pengujian ini dilakukan untuk memastikan bahwa semua komponen dalam aplikasi berfungsi dengan baik dan terintegrasi secara efektif, serta memberikan pengalaman pengguna yang memuaskan

Langkah-langkah dalam End-to-End Testing

1. Perencanaan Pengujian: Tentukan tujuan pengujian dan skenario yang akan diuji berdasarkan kebutuhan pengguna.
2. Persiapan Lingkungan Uji: Siapkan lingkungan yang mencerminkan kondisi produksi, termasuk perangkat keras dan perangkat lunak yang diperlukan.
3. Desain Kasus Uji: Buat kasus uji yang mencakup semua langkah interaksi pengguna dengan aplikasi, mulai dari login hingga penyelesaian tugas.
4. Eksekusi Pengujian: Jalankan kasus uji dan catat hasilnya. Pastikan untuk mengamati setiap langkah untuk mendeteksi masalah.
5. Analisis Hasil: Tinjau hasil pengujian untuk mengidentifikasi bug atau masalah performa dan lakukan perbaikan jika diperlukan.
6. Dokumentasi: Catat semua temuan dan perbaikan yang dilakukan selama proses pengujian untuk referensi di masa mendatang.

CONTOH KASUS

Salah satu contoh kasus yang relevan untuk End-to-End Testing dapat ditemukan dalam penelitian yang dilakukan oleh Gu et al. (2023) berjudul "Acto: Automatic End-to-End Testing for Operation Correctness of Cloud System Management". Penelitian ini berfokus pada pengujian otomatis untuk operator sistem manajemen cloud, khususnya yang digunakan dalam platform seperti Kubernetes.

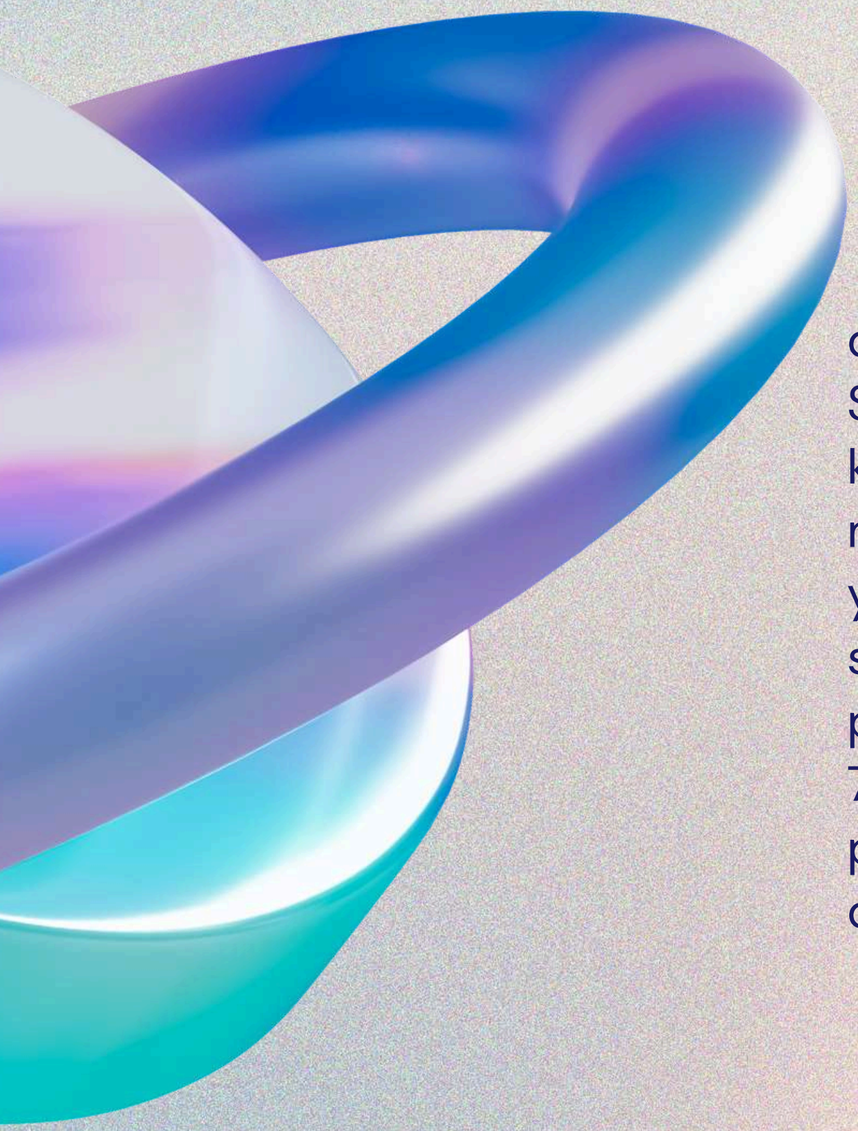
Penelitian ini menunjukkan pentingnya E2E testing dalam konteks pengelolaan cloud dan bagaimana teknik otomatis seperti Acto dapat meningkatkan kualitas perangkat lunak dengan mendeteksi bug yang mungkin tidak terlihat melalui metode pengujian tradisional lainnya. Dengan menerapkan E2E testing secara efektif, tim pengembang dapat memastikan bahwa sistem manajemen cloud berfungsi dengan baik dan dapat diandalkan dalam situasi nyata.

Behavior-Driven Testing

Behavior-Driven Testing (BDT) adalah metode pengujian perangkat lunak yang berfokus pada perilaku pengguna dan interaksi mereka dengan sistem. BDT bertujuan untuk memastikan bahwa aplikasi memenuhi kebutuhan pengguna dengan cara yang dapat dipahami oleh semua pemangku kepentingan, termasuk pengembang, penguji, dan pihak bisnis.

Langkah-langkah dalam Behavior-Driven Testing

1. Identifikasi Skenario Pengujian: Tentukan perilaku sistem yang ingin diuji berdasarkan kebutuhan pengguna.
2. Tulis Kasus Uji Menggunakan Gherkin: Gunakan sintaks Gherkin untuk menulis skenario pengujian dalam format "Given [kondisi], When [aksi], Then [hasil]".
3. Kolaborasi dengan Pemangku Kepentingan: Libatkan semua pihak terkait untuk memastikan bahwa skenario mencerminkan kebutuhan dan harapan pengguna.
4. Implementasi dan Eksekusi Pengujian: Implementasikan skenario sebagai tes otomatis menggunakan alat BDT seperti Cucumber atau SpecFlow.
5. Analisis Hasil: Tinjau hasil pengujian untuk memastikan bahwa sistem berperilaku sesuai harapan. Jika ada masalah, lakukan perbaikan yang diperlukan.



CONTOH KASUS

Salah satu contoh penerapan Behavior-Driven Testing (BDT) dapat ditemukan dalam penelitian yang dilakukan pada aplikasi Web Sentratekstil. Dalam kasus ini, tim pengembang melakukan elisitasi kebutuhan pengguna melalui wawancara dan kuesioner untuk menentukan kesesuaian antara kebutuhan pengguna dan aplikasi yang dibangun. Setelah itu, mereka menerapkan BDT untuk membuat skenario pengujian berdasarkan pernyataan kebutuhan. Hasil dari pengujian ini menunjukkan tingkat kepuasan pengguna sebesar 78,9%, yang menandakan bahwa aplikasi tersebut memenuhi harapan pengguna dan membantu mengidentifikasi beberapa bug yang perlu diperbaiki sebelum peluncuran

Acceptance Testing

Acceptance Testing adalah tahap penting dalam pengembangan perangkat lunak yang bertujuan untuk memastikan bahwa sistem memenuhi kebutuhan dan harapan pengguna sebelum diluncurkan. Pengujian ini dilakukan oleh pengguna akhir atau klien untuk memvalidasi bahwa aplikasi berfungsi sesuai dengan spesifikasi yang telah ditetapkan.

Langkah-langkah dalam Behavior-Driven Testing

1. Perencanaan: Menentukan kriteria penerimaan dan merencanakan skenario pengujian.
2. Desain Kasus Uji: Membuat kasus uji berdasarkan kebutuhan pengguna.
3. Pelaksanaan Pengujian: Melakukan pengujian sesuai dengan skenario yang telah direncanakan.
4. Dokumentasi Hasil: Mencatat hasil pengujian dan umpan balik dari pengguna.
5. Tindak Lanjut: Mengidentifikasi masalah yang ditemukan dan melakukan perbaikan jika diperlukan.

CONTOH KASUS

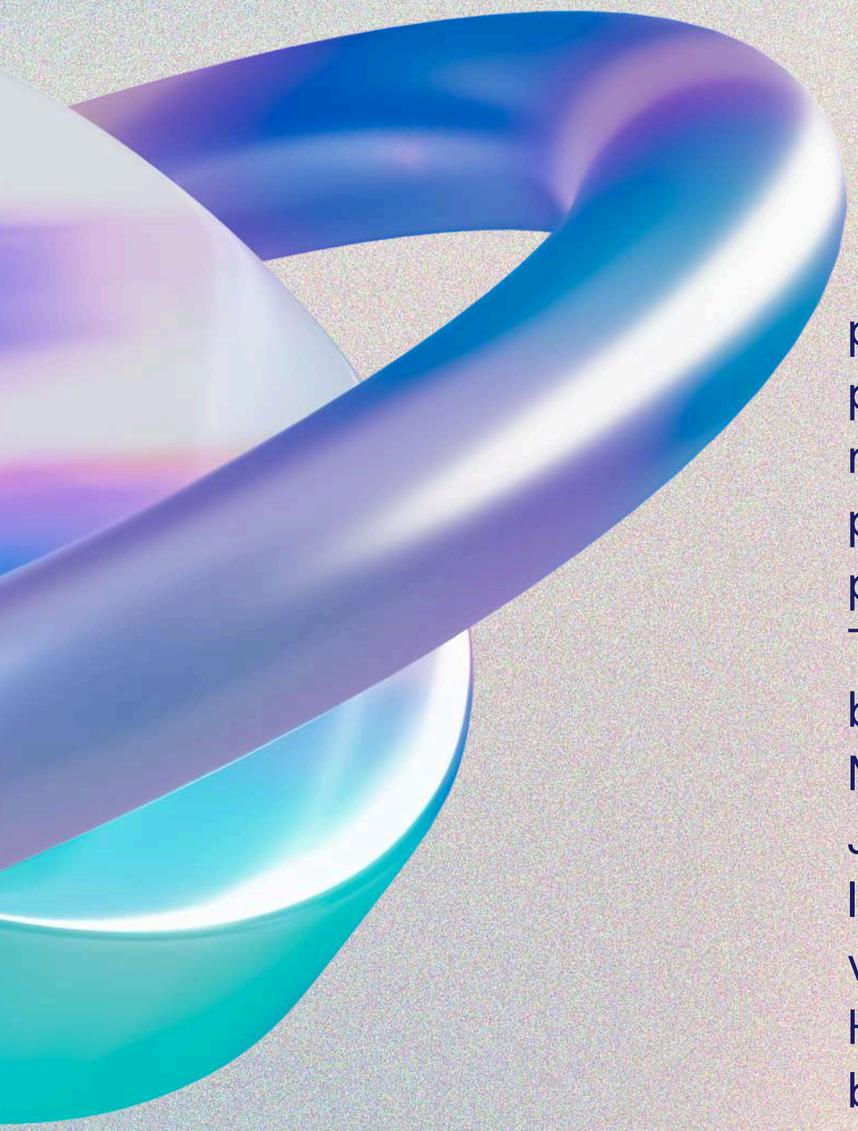
Salah satu contoh penerapan Acceptance Testing dapat ditemukan dalam penelitian mengenai sistem informasi pengelolaan bedah rumah di Dinas Perumahan Rakyat dan Kawasan Permukiman Kabupaten Jepara. Dalam kasus ini, tim pengembang melakukan User Acceptance Testing (UAT) untuk memastikan bahwa sistem dapat mempermudah proses pendataan dan penyeleksian rumah masyarakat yang berhak menerima bantuan. Hasil dari pengujian ini menunjukkan bahwa sistem mendapatkan nilai persentase sebesar 86,2%, yang menandakan bahwa sistem informasi tersebut sangat sesuai dan mudah digunakan oleh pengguna, sehingga meningkatkan efisiensi dalam pelaksanaan program bantuan bedah rumah

API Testing

API Testing adalah proses pengujian Application Programming Interface (API) untuk memastikan bahwa API berfungsi dengan baik, memenuhi spesifikasi yang diinginkan, dan dapat menangani berbagai skenario penggunaan. Pengujian ini penting karena API sering kali menjadi jembatan antara berbagai sistem dan aplikasi, sehingga kestabilan dan keandalannya sangat krusial.

Langkah-langkah dalam Behavior-Driven Testing

1. Perencanaan: Menentukan tujuan pengujian dan skenario yang akan diuji.
2. Desain Kasus Uji: Membuat kasus uji berdasarkan spesifikasi API.
3. Eksekusi Pengujian: Melakukan pengujian menggunakan alat seperti Postman, SoapUI, atau JMeter.
4. Analisis Hasil: Menganalisis hasil pengujian untuk mengidentifikasi masalah atau bug.
5. Dokumentasi Hasil: Mencatat hasil pengujian dan langkah-langkah yang diambil.



CONTOH KASUS

Salah satu contoh penerapan API Testing dapat ditemukan dalam penelitian oleh Ni Luh Ayu Sonia Ginasaria et al. (2021) mengenai pengujian Stress Testing pada sistem pelayanan berbasis API menggunakan Apache JMeter. Dalam penelitian ini, tim melakukan pengujian untuk menganalisis ketahanan sistem ketika dihadapkan pada beban tinggi.

Tujuan Penelitian: Menguji ketahanan sistem pelayanan laboratorium berbasis Android yang menggunakan API untuk mengakses data.

Metodologi: Pengujian dilakukan dengan menggunakan Apache JMeter untuk menguji beberapa endpoint API, termasuk laboratorium, layanan laboratorium, dan berita. Skenario pengujian mencakup variasi beban dengan 25, 50, dan 75 sampel.

Hasil Pengujian: Hasil menunjukkan bahwa meskipun ada peningkatan beban, sistem mampu mempertahankan waktu respons yang konstan dengan error rate 0%, menunjukkan bahwa sistem memiliki ketahanan yang baik terhadap beban tinggi.

Cross-Platform Testing

Cross-Platform Testing adalah proses pengujian perangkat lunak untuk memastikan bahwa aplikasi dapat berfungsi dengan baik di berbagai platform dan perangkat. Dengan meningkatnya penggunaan aplikasi di berbagai sistem operasi dan perangkat, penting untuk melakukan pengujian lintas platform untuk memastikan pengalaman pengguna yang konsisten.

Metode dalam Cross-Platform Testing

1. Manual Testing: Pengujian dilakukan secara manual dengan menguji aplikasi di setiap platform dan perangkat secara terpisah.
2. Automated Testing: Menggunakan alat otomatisasi untuk menjalankan skrip pengujian di berbagai platform. Alat seperti Selenium, Appium, atau TestComplete sering digunakan.
3. Cloud-Based Testing: Menggunakan layanan cloud untuk menguji aplikasi di berbagai perangkat dan platform tanpa perlu memiliki perangkat fisik.

CONTOH KASUS

Salah satu contoh penerapan Cross-Platform Testing dapat ditemukan dalam penelitian oleh Husni et al. (2021) yang berjudul "Pengujian Aplikasi Mobile Berbasis Android Menggunakan Metode Cross-Platform". Dalam penelitian ini, tim pengembang menguji aplikasi mobile yang dirancang untuk berjalan di Android dan iOS.

Tujuan Penelitian: Untuk mengevaluasi kinerja aplikasi mobile pada dua platform berbeda (Android dan iOS) dengan fokus pada fungsionalitas dan responsivitas.

Metodologi: Penelitian ini menggunakan pendekatan automated testing dengan alat Appium untuk menjalankan skrip pengujian pada kedua platform. Skenario pengujian mencakup fungsi dasar seperti pendaftaran pengguna, login, dan interaksi dengan antarmuka pengguna.

Hasil Pengujian: Hasil menunjukkan bahwa meskipun ada beberapa perbedaan dalam tampilan antarmuka, fungsionalitas utama aplikasi bekerja dengan baik di kedua platform. Namun, beberapa masalah kompatibilitas ditemukan pada versi tertentu dari iOS yang memerlukan perbaikan lebih lanjut.

Acceptance Testing-Driven Development

Acceptance Testing-Driven Development (ATDD) adalah metodologi pengembangan perangkat lunak yang berfokus pada kolaborasi antara pemangku kepentingan, termasuk pengembang, penguji, dan pemilik produk, untuk mendefinisikan kriteria penerimaan sebelum pengembangan dimulai. ATDD bertujuan untuk memastikan bahwa semua pihak memiliki pemahaman yang sama mengenai kebutuhan pengguna dan bagaimana sistem seharusnya berfungsi.

Proses ATDD

1. Pengumpulan Tim: Mengumpulkan semua pihak terkait, termasuk pengembang, penguji, dan pemilik produk untuk mendiskusikan kebutuhan dan tujuan proyek.
2. Definisi Kriteria Penerimaan: Bersama-sama menulis kriteria penerimaan yang jelas untuk setiap fitur yang akan dikembangkan.
3. Penulisan Tes Penerimaan: Mengubah kriteria penerimaan menjadi tes penerimaan yang dapat dieksekusi.
4. Pengembangan Fitur: Pengembang kemudian menulis kode untuk memenuhi kriteria penerimaan yang telah ditentukan.
5. Pengujian: Setelah pengembangan, tes penerimaan dijalankan untuk memastikan bahwa fitur berfungsi sesuai harapan.
6. Iterasi dan Ulangi: Proses diulang untuk fitur-fitur berikutnya.

CONTOH KASUS

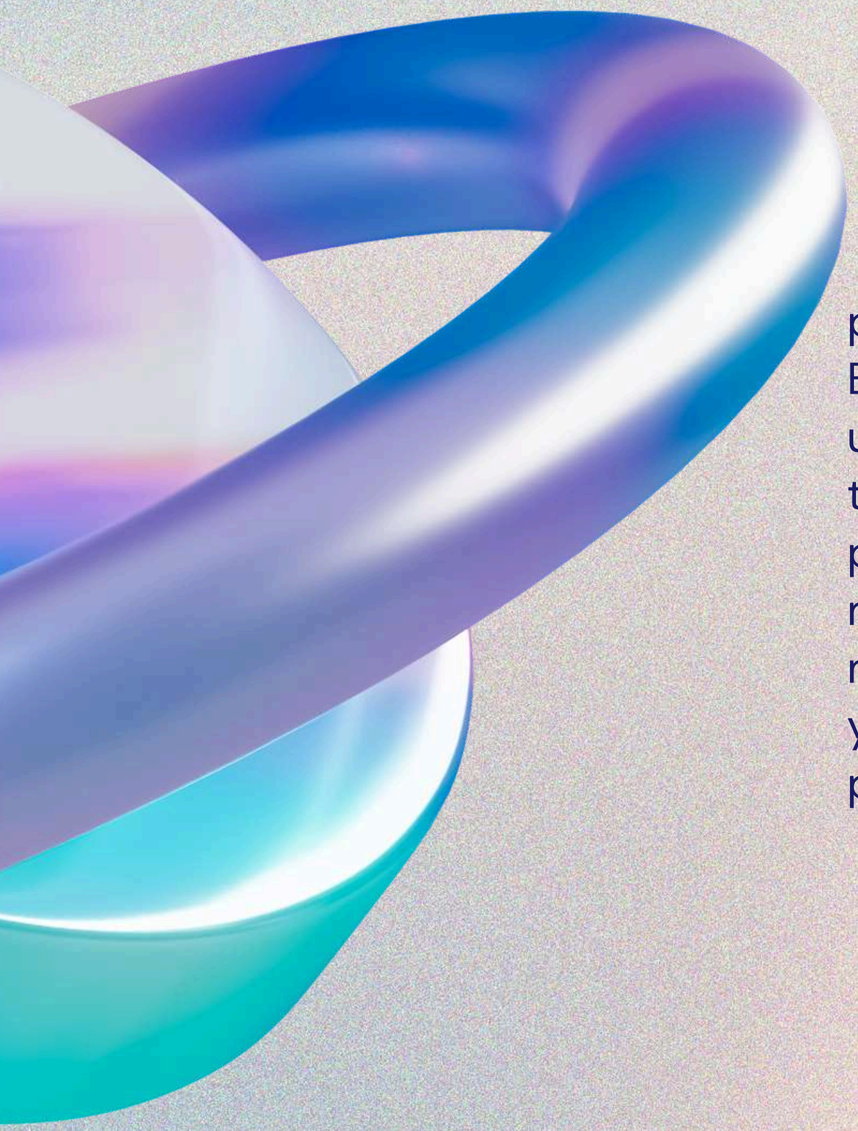
Salah satu contoh penerapan Acceptance Testing dapat ditemukan dalam penelitian yang dilakukan oleh Eko Suprapto mengenai pengujian User Acceptance Testing (UAT) pada sistem upgrade PABX menjadi IP PBX di BNI Kanwil Padang. Dalam kasus ini, tim pengembang melakukan UAT untuk memastikan bahwa sistem baru memenuhi kebutuhan pengguna dan berfungsi dengan baik dalam lingkungan yang diharapkan. Hasil dari pengujian ini menunjukkan bahwa upgrade sistem berhasil meningkatkan kualitas layanan komunikasi suara tanpa mengganggu komunikasi data yang sudah ada, dengan tingkat keberterimaan pengguna yang tinggi. Penelitian ini menekankan pentingnya melibatkan pengguna dalam proses pengujian untuk mendapatkan umpan balik yang konstruktif sebelum peluncuran sistem secara penuh.

Scalability Testing

Scalability Testing adalah metode pengujian non fungsional yang digunakan untuk mengevaluasi kemampuan suatu sistem, aplikasi, atau infrastruktur untuk menangani peningkatan beban atau volume yang lebih besar secara efisien tanpa mengalami penurunan signifikan dalam performa. Tujuannya adalah untuk memastikan bahwa sistem dapat ditingkatkan (scale up) atau dimeleksalkan (scale out) untuk menjaga kualitas layanan saat penggunaan atau kebutuhan meningkat.

Langkah-Langkah dalam Scalability Testing

1. Perencanaan dan Penentuan Tujuan: Tentukan skenario dan tujuan dari pengujian, seperti meningkatkan jumlah pengguna simultan, volume data yang diproses, atau transaksi per detik.
2. Pengaturan Lingkungan Uji: Siapkan lingkungan uji yang mencerminkan kondisi nyata, termasuk konfigurasi perangkat keras, jaringan, dan perangkat lunak yang relevan.
3. Eksekusi Pengujian: Jalankan pengujian dengan meningkatkan beban secara bertahap atau secara drastis untuk melihat bagaimana sistem menanggapi peningkatan beban. Pengujian dapat mencakup scaling up (menambah sumber daya pada sistem yang ada) atau scaling out (menambah node atau instance dalam sistem terdistribusi)32.
4. Pemantauan dan Pengukuran Performa: Kumpulkan data performa seperti waktu respons, throughput, penggunaan CPU dan memori, dan lain-lain selama pengujian untuk mengevaluasi performa sistem.
5. Analisis Hasil: Analisis hasil pengujian untuk menentukan apakah sistem dapat di-skala secara efektif dan mempertahankan performa yang diharapkan saat beban meningkat. Identifikasi bottleneck atau titik lemah yang dapat membatasi kemampuan scaling system.
6. Strategi Uji Skalabilitas: Strategi pengujian skalabilitas berbeda dalam hal jenis aplikasi yang diuji. Jika suatu aplikasi mengakses database, parameter pengujiannya adalah menguji ukuran database dalam kaitannya dengan jumlah pengguna dan seterusnya.



CONTOH KASUS

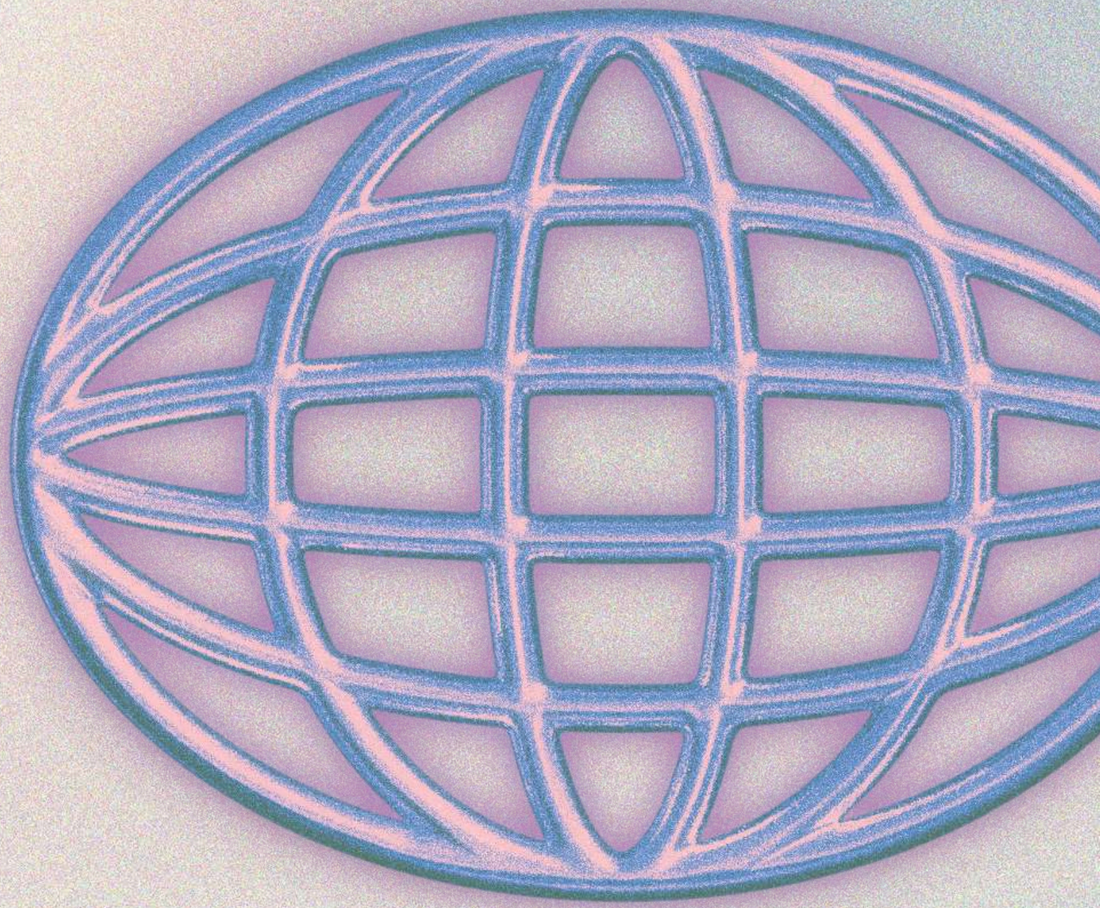
Salah satu contoh penerapan Scalability Testing dapat ditemukan dalam pengujian sistem e-commerce selama periode penjualan besar seperti Black Friday. Dalam kasus ini, tim pengembang melakukan scalability testing untuk memastikan bahwa platform dapat menangani lonjakan pengguna dan transaksi secara bersamaan. Pengujian dilakukan dengan mensimulasikan peningkatan jumlah pengguna secara bertahap hingga mencapai kapasitas maksimum yang diharapkan. Hasil dari pengujian ini membantu tim mengidentifikasi bottleneck dalam infrastruktur dan melakukan optimasi yang diperlukan, sehingga sistem tetap responsif dan stabil selama periode puncak penjualan .

Security Testing

Security Testing adalah proses pengujian yang dilakukan untuk mengidentifikasi kerentanan, celah, dan masalah keamanan dalam suatu sistem perangkat lunak atau aplikasi. Tujuan utama dari security testing adalah untuk memastikan bahwa data dan sumber daya sistem terlindungi dari ancaman dan serangan siber.

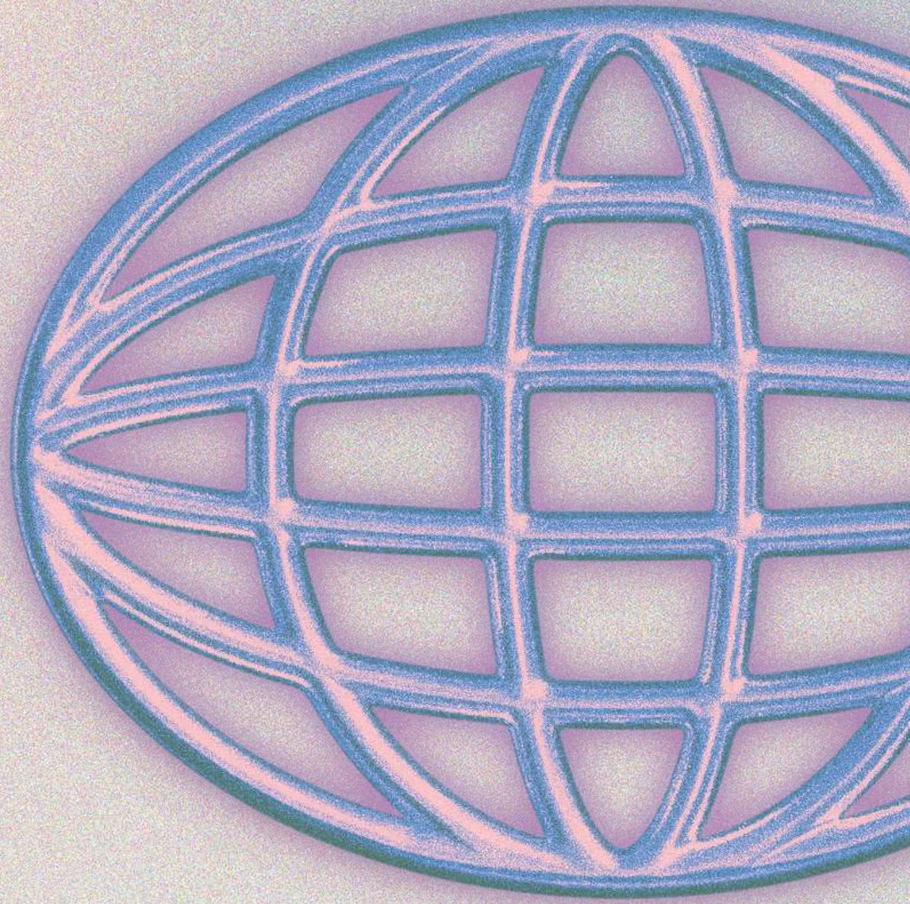
Tujuan Security Testing

1. Identifikasi Kerentanan: Menemukan celah keamanan yang dapat dieksploitasi oleh penyerang.
2. Perlindungan Data: Memastikan bahwa data sensitif dan informasi penting terlindungi dengan baik.
3. Kepatuhan: Memastikan bahwa aplikasi memenuhi standar keamanan yang ditetapkan oleh industri atau regulasi.



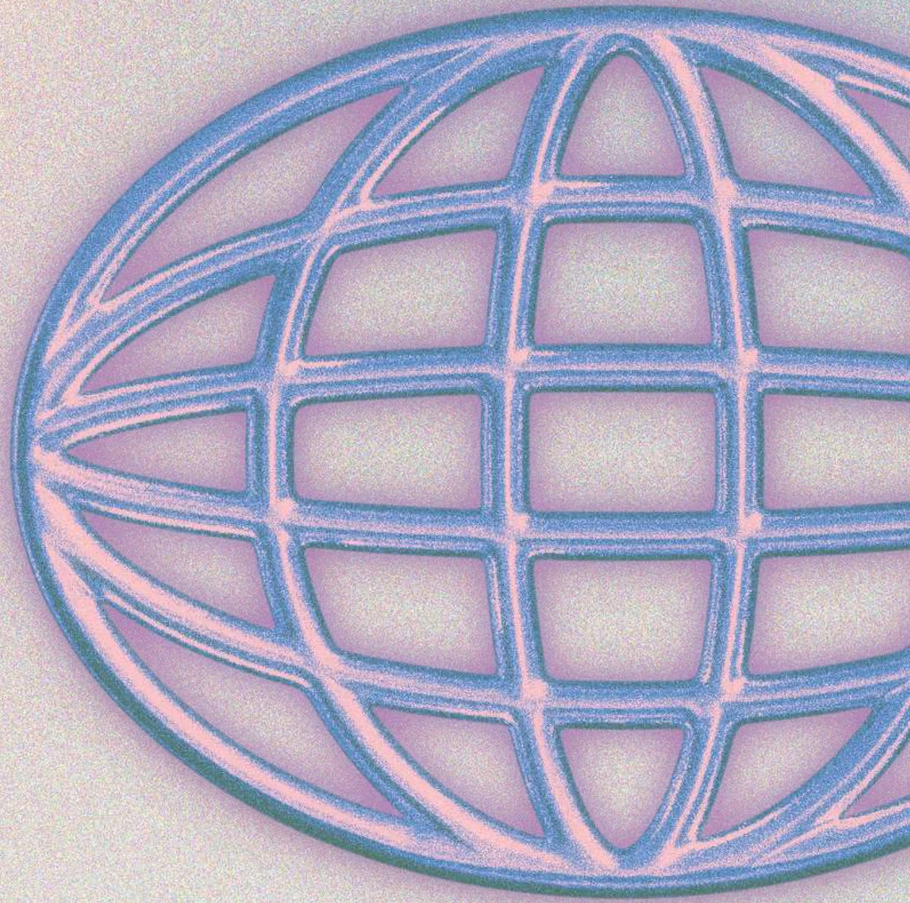
Jenis-jenis Security Testing

1. Penetration Testing: Simulasi serangan untuk mengevaluasi keamanan sistem dengan cara mencoba mengeksploitasi kerentanan yang ada.
2. Security Scanning: Proses pemindaian sistem untuk menemukan kerentanan atau modifikasi file yang tidak diinginkan, baik secara otomatis maupun manual.
3. Vulnerability Assessment: Penilaian menyeluruh terhadap kerentanan yang ada dalam sistem untuk menentukan risiko yang terkait.
4. Risk Assessment: Mengidentifikasi dan menganalisis risiko terhadap keamanan sistem serta mengklasifikasikan mereka berdasarkan tingkat keparahan.
5. Security Auditing: Evaluasi sistem keamanan secara menyeluruh untuk memastikan bahwa kebijakan dan prosedur keamanan diterapkan dengan benar.



Metodologi dalam Security Testing

1. Black Box Testing: Pengujian dilakukan tanpa pengetahuan tentang struktur internal sistem. Penguji fokus pada input dan output tanpa mengetahui cara kerja internalnya.
2. White Box Testing: Penguji memiliki akses penuh terhadap kode sumber dan arsitektur sistem, memungkinkan analisis mendalam terhadap potensi kerentanan.
3. Grey Box Testing: Kombinasi antara black box dan white box testing, di mana penguji memiliki pengetahuan terbatas tentang struktur internal tetapi tetap melakukan pengujian dari sudut pandang pengguna.



CONTOH KASUS

Salah satu contoh penerapan Security Testing dapat ditemukan dalam pengujian aplikasi perbankan yang baru diluncurkan. Dalam kasus ini, tim pengembang melakukan penetration testing untuk memastikan bahwa sistem aman dari serangan siber. Pengujian dilakukan dengan mensimulasikan berbagai jenis serangan, termasuk SQL Injection dan Cross-Site Scripting (XSS). Hasil dari pengujian ini membantu tim mengidentifikasi kerentanan kritis dan melakukan perbaikan yang diperlukan, sehingga melindungi data pengguna dan meningkatkan kepercayaan pelanggan sebelum aplikasi resmi digunakan.

CONTOH KASUS

Quick Start | Request | Response | Requester

Automated Scan

ZAP by Checkmarx

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: https://regmhs.unhas.ac.id/ Select...

Use traditional spider: ☑

Use ajax spider: If Modern with Firefox Headless

Attack | Stop

Progress: Manually stopped

History | Search | Alerts | Output | Spider | AJAX Spider | Active Scan | WebSockets

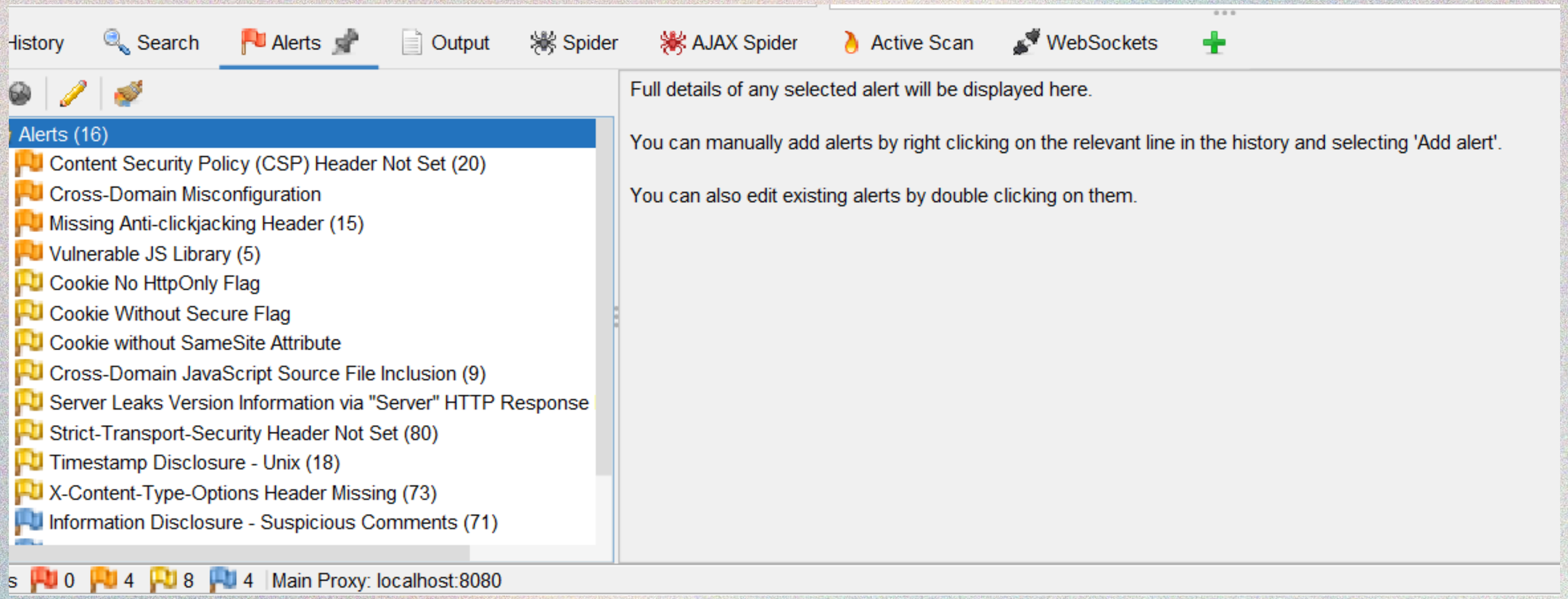
New Scan | Progress: 0: https://regmhs.unhas.ac.id/ | 0% | Current Scans: 0 | Num Requests: 5 | New Alerts: 0 | Export

Sent Messages | Filtered Messages

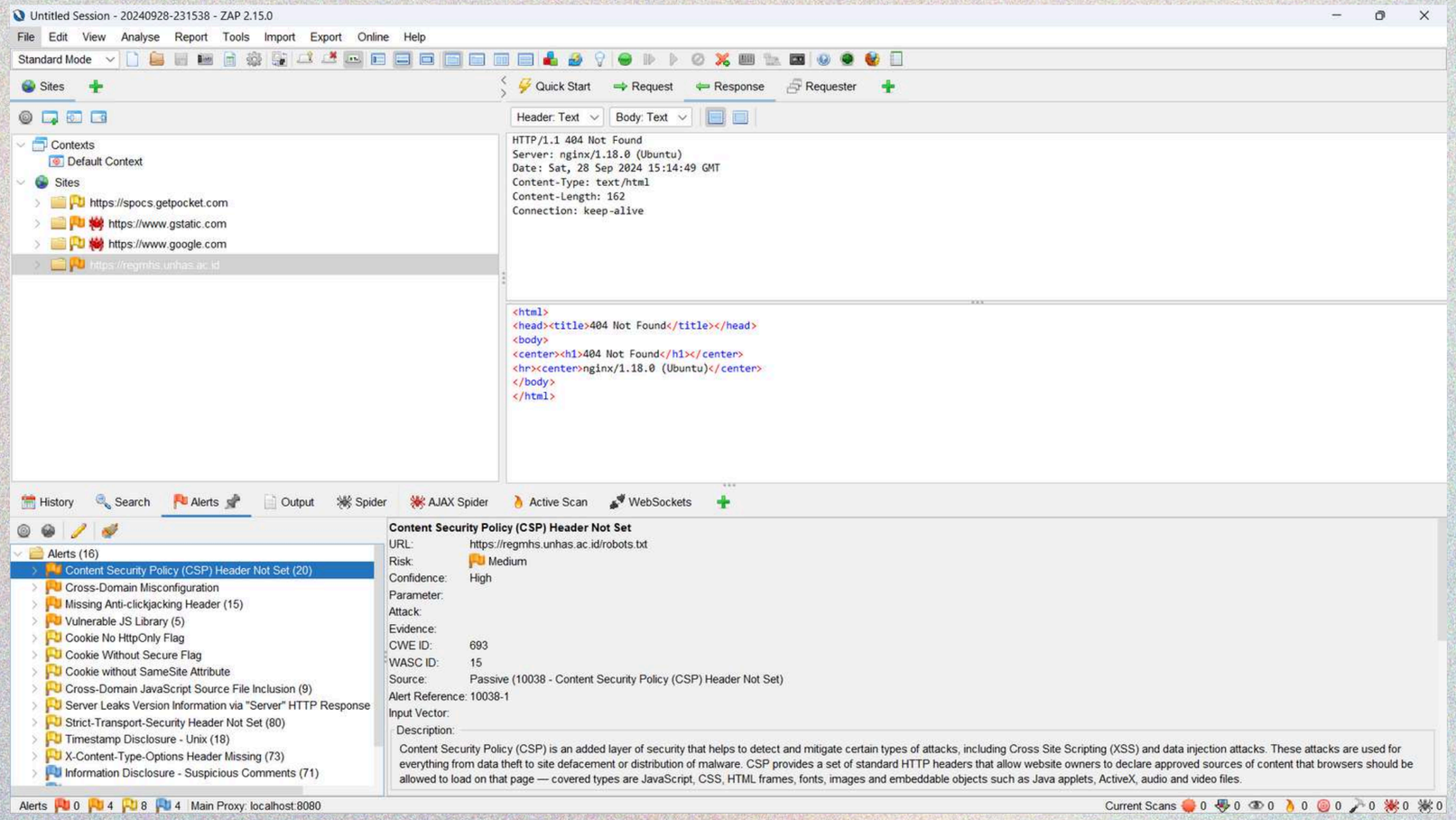
ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
2,286	9/28/24, 11:19:26 PM	9/28/24, 11:19:27 PM	GET	https://regmhs.unhas.ac.id/674002106876738939 5.php	404	Not Found	258 ms	164 bytes	162 bytes
2,290	9/28/24, 11:19:27 PM	9/28/24, 11:19:27 PM	GET	https://regmhs.unhas.ac.id/auth/9160236965139084819	404	Not Found	97 ms	164 bytes	162 bytes
2,292	9/28/24, 11:19:27 PM	9/28/24, 11:19:27 PM	GET	https://regmhs.unhas.ac.id/bower_components/820495913	404	Not Found	82 ms	164 bytes	162 bytes
2,301	9/28/24, 11:19:27 PM	9/28/24, 11:19:27 PM	GET	https://regmhs.unhas.ac.id/bower_components/moment/88	404	Not Found	85 ms	164 bytes	162 bytes
2,304	9/28/24, 11:19:27 PM	9/28/24, 11:19:27 PM	GET	https://regmhs.unhas.ac.id/bower_components/moment/mi	404	Not Found	91 ms	164 bytes	162 bytes

Alerts 0 4 8 4 | Main Proxy: localhost:8080 | Current Scans 0 0 0 0 0 0 0 0 0

CONTOH KASUS



CONTOH KASUS



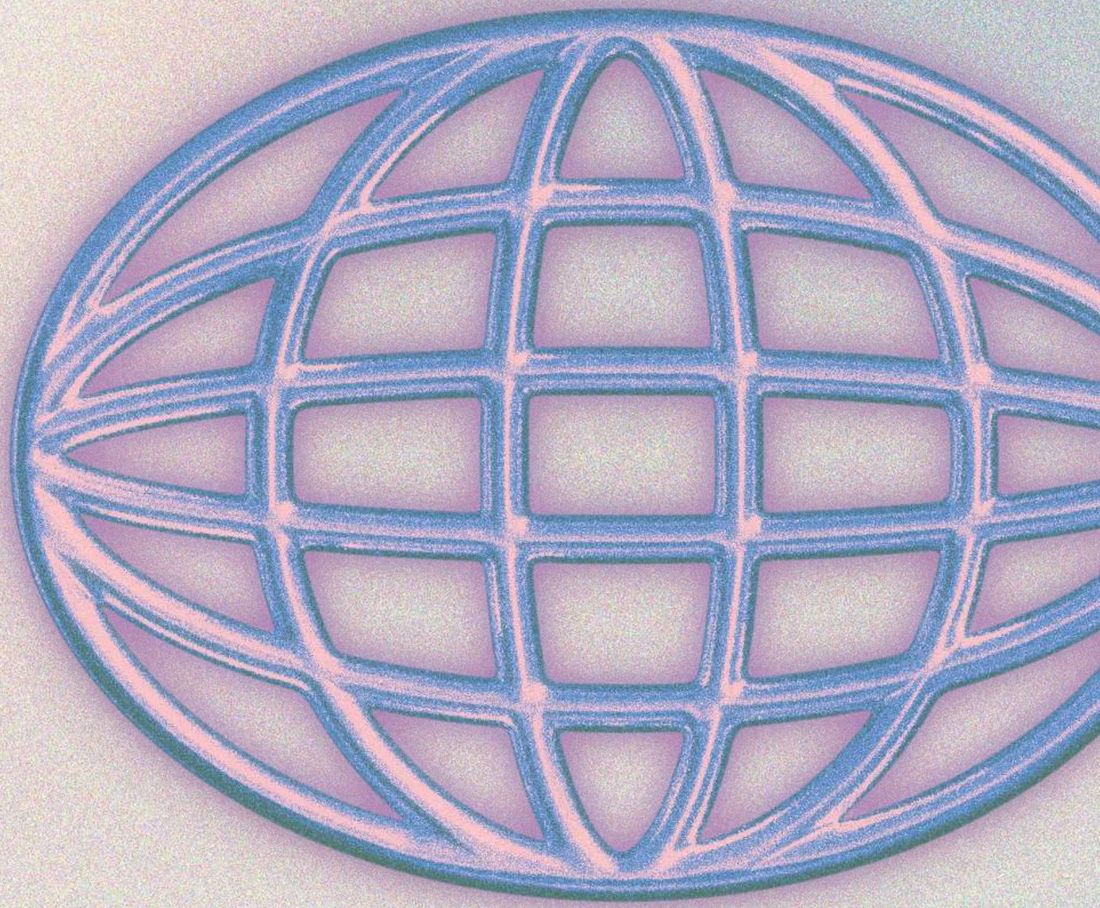
Kerentanan ini dikategorikan sebagai risiko Medium, dengan tingkat keyakinan (confidence) High, yang menunjukkan bahwa alat OWASP ZAP yakin bahwa masalah ini merupakan kerentanan nyata. Tidak adanya CSP pada aplikasi memungkinkan potensi serangan Cross-Site Scripting (XSS) dan injeksi data lainnya. Content Security Policy adalah mekanisme keamanan penting yang bertujuan untuk membatasi sumber konten yang dapat dimuat pada aplikasi, termasuk JavaScript, CSS, gambar, iframe, dan elemen lainnya. Dengan tidak adanya CSP header, aplikasi lebih rentan terhadap manipulasi konten oleh pihak ketiga

Database Testing

Database Testing adalah proses yang dilakukan untuk memastikan bahwa sistem basis data berfungsi dengan baik dan memenuhi kebutuhan pengguna. Ini mencakup validasi data, integritas, dan fungsionalitas database untuk memastikan bahwa data dapat disimpan, diambil, dan dikelola dengan benar

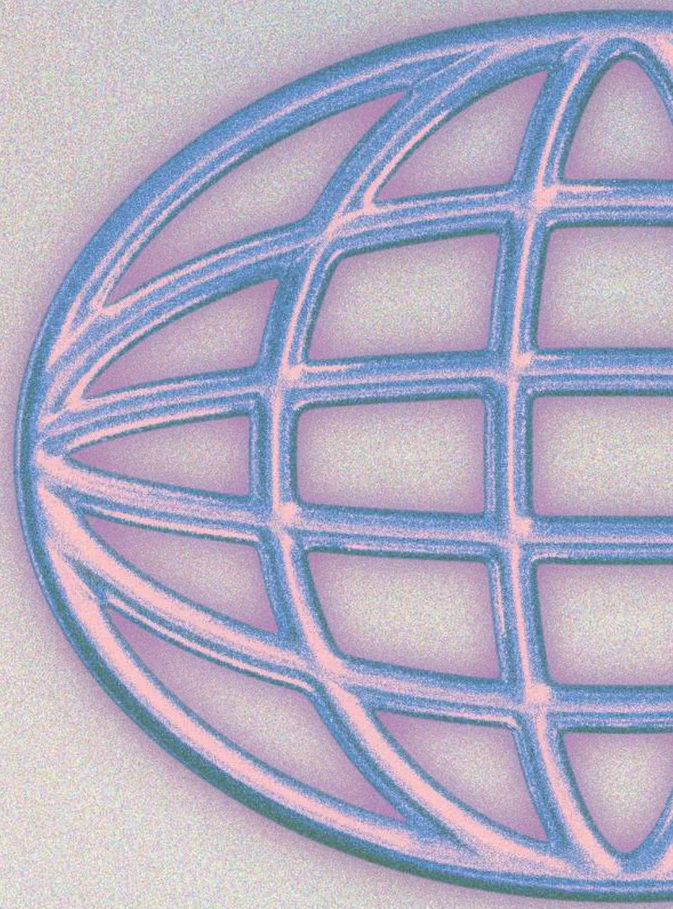
Tujuan Database Testing

1. Validasi Data: Memastikan bahwa data yang disimpan dalam database adalah akurat dan konsisten.
2. Integritas Data: Menjamin bahwa hubungan antar tabel dan data tetap terjaga.
3. Fungsionalitas: Memastikan bahwa semua fungsi dan fitur database bekerja sesuai dengan spesifikasi yang ditetapkan.



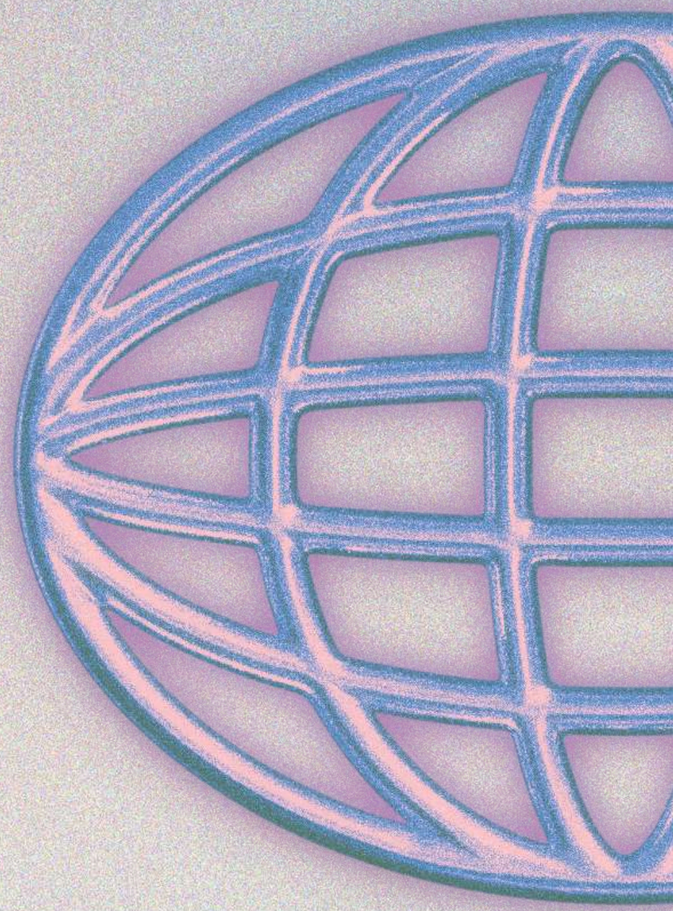
Jenis-jenis Database Testing

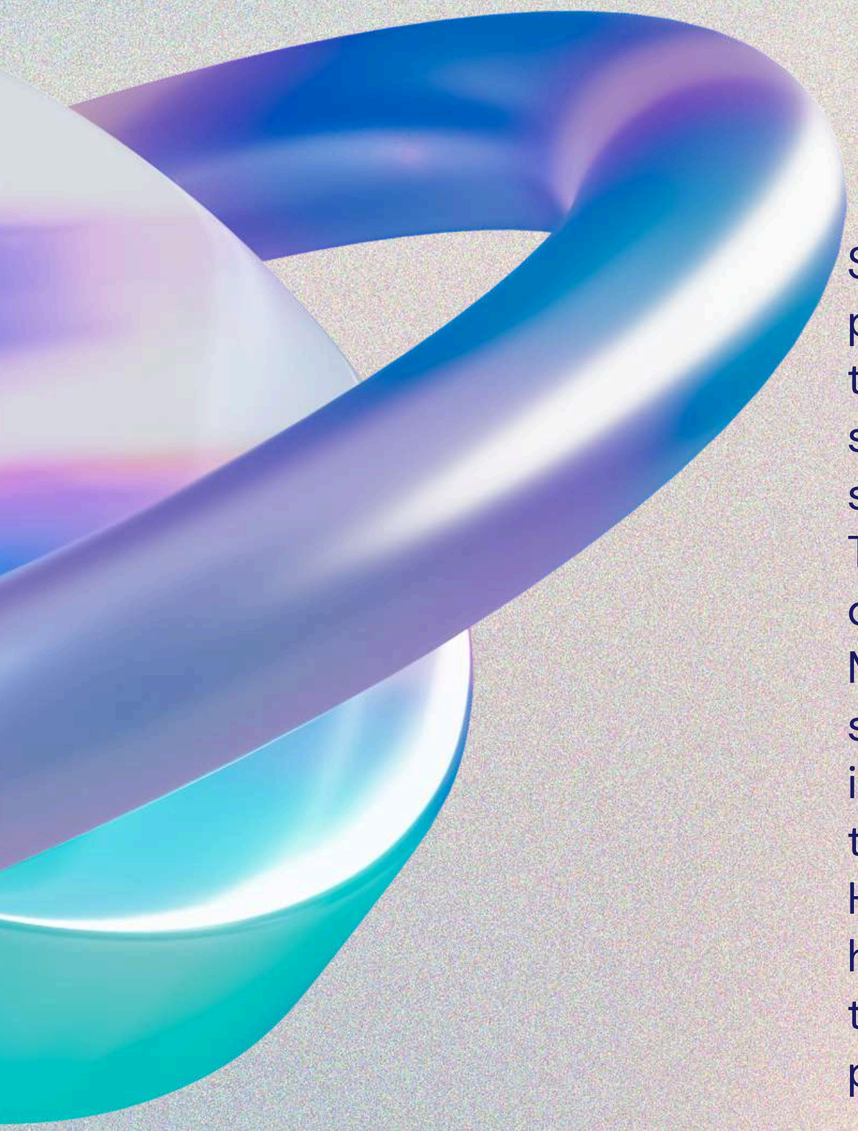
1. Functional Testing: Menguji fungsionalitas database dari perspektif pengguna akhir. Ini termasuk pengujian query SQL untuk memastikan bahwa operasi penyimpanan, pengambilan, pembaruan, dan penghapusan data berjalan dengan baik.
2. Structural Testing: Memverifikasi elemen-elemen dalam repositori data yang tidak dapat diakses langsung oleh pengguna. Ini mencakup pengujian skema database dan validasi struktur tabel.
3. Performance Testing: Menguji kemampuan database untuk menangani volume data yang besar dan transaksi yang tinggi sambil mempertahankan responsivitas.
4. Security Testing: Mengidentifikasi kerentanan dalam sistem basis data untuk melindungi data dari akses tidak sah atau modifikas.
5. Compliance Testing: Memastikan bahwa database mematuhi standar industri dan regulasi yang relevan seperti GDPR atau HIPAA.



Metodologi dalam Database Testing

1. Persiapan Lingkungan: Menyiapkan lingkungan pengujian yang mencakup konfigurasi database dan alat yang diperlukan.
2. Menentukan Kasus Uji: Mengidentifikasi skenario pengujian berdasarkan kebutuhan bisnis dan spesifikasi sistem.
3. Eksekusi Pengujian: Melakukan pengujian berdasarkan kasus uji yang telah ditentukan, termasuk menjalankan query SQL dan memvalidasi hasilnya.
4. Analisis Hasil: Menganalisis hasil pengujian untuk mengidentifikasi masalah atau ketidaksesuaian antara hasil aktual dan yang diharapkan.
5. Pelaporan Temuan: Mendokumentasikan hasil pengujian, termasuk masalah yang ditemukan dan rekomendasi perbaikan





CONTOH KASUS

Salah satu contoh penerapan Database Testing dapat ditemukan dalam pengujian sistem manajemen inventaris perusahaan ritel. Dalam kasus ini, tim pengembang melakukan functional testing untuk memastikan bahwa semua transaksi terkait produk, seperti penambahan stok, pengurangan stok, dan pencarian produk, berfungsi dengan baik.

Tujuan Penelitian: Untuk memastikan bahwa sistem manajemen inventaris dapat menyimpan dan mengelola data produk secara akurat.

Metodologi: Tim menggunakan query SQL untuk menguji berbagai fungsi seperti menambahkan produk baru ke dalam database, memperbarui informasi produk yang ada, serta mencari produk berdasarkan kriteria tertentu.

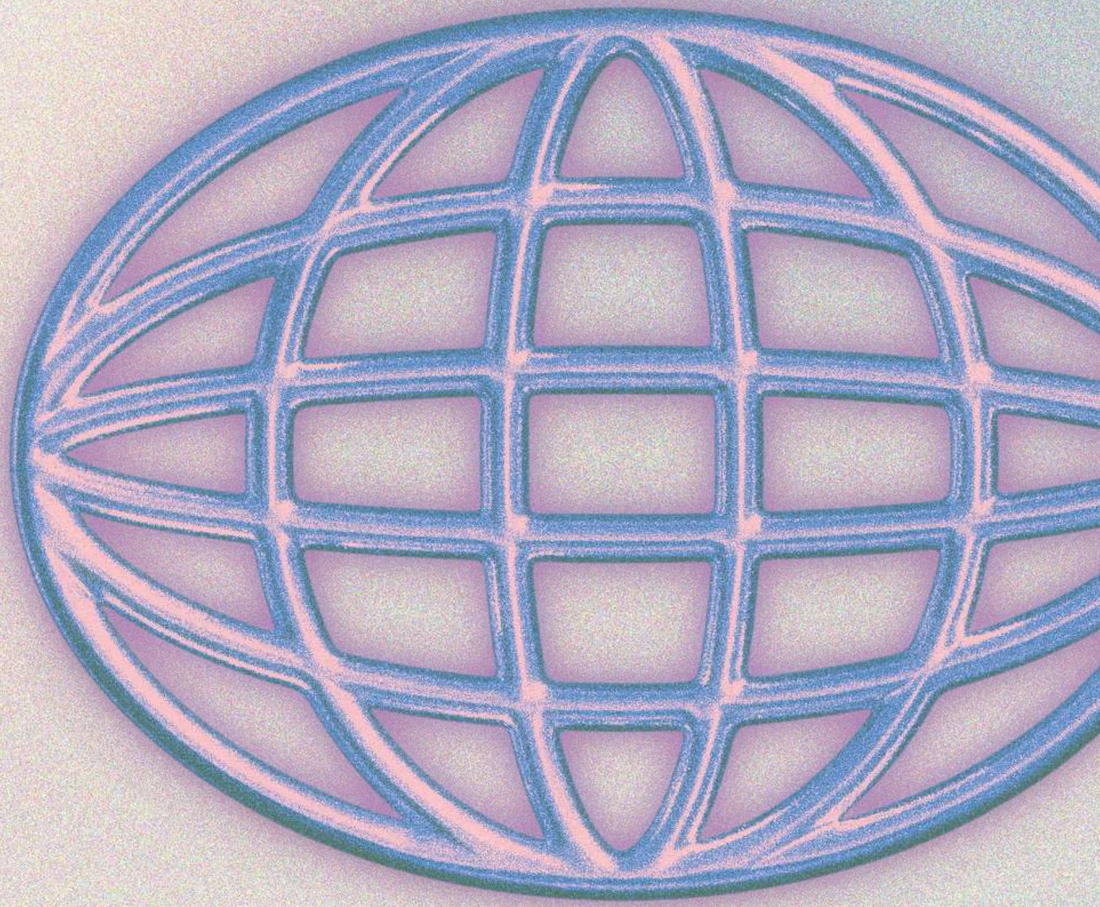
Hasil Pengujian: Hasil menunjukkan bahwa semua fungsi berjalan sesuai harapan tanpa adanya kesalahan, tetapi tim menemukan beberapa masalah terkait integritas referensial antara tabel produk dan tabel kategori yang perlu diperbaiki sebelum peluncuran sistem.

Penetration & Exploitation Testing

Penetration Testing (pentest) adalah metode pengujian keamanan yang dilakukan untuk mengidentifikasi dan mengeksploitasi kerentanan dalam sistem, aplikasi, atau jaringan. Tujuan utama dari penetration testing adalah untuk menilai keamanan sistem dengan cara mensimulasikan serangan yang mungkin dilakukan oleh penyerang yang berniat jahat.

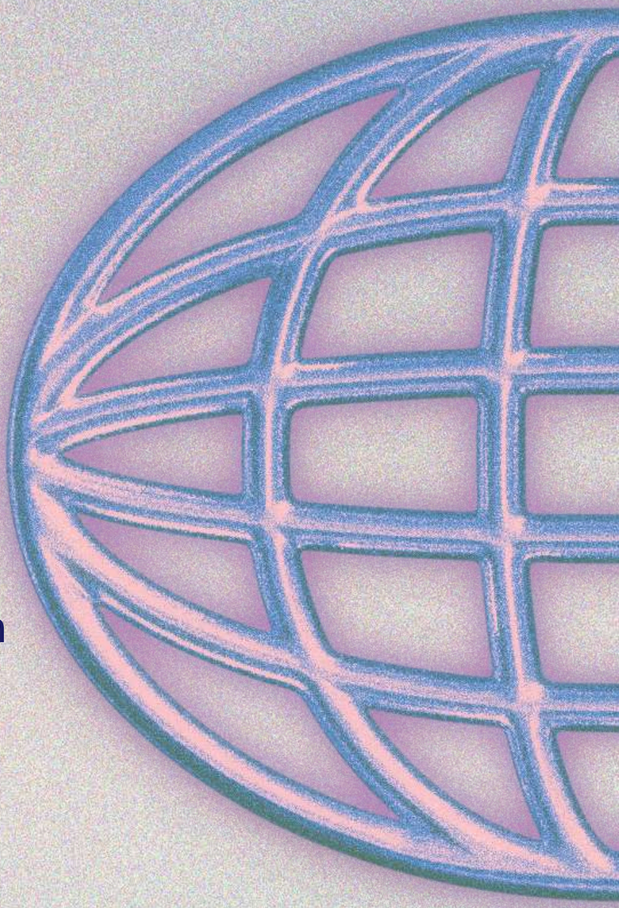
Tujuan Penetration Testing

1. Identifikasi Kerentanan: Menemukan celah keamanan yang dapat dieksploitasi oleh penyerang.
2. Evaluasi Keamanan: Menilai efektivitas kontrol keamanan yang ada dalam melindungi data dan sistem.
3. Peningkatan Keamanan: Memberikan rekomendasi untuk memperbaiki kerentanan yang ditemukan selama pengujian.



Proses Penetration Testing

1. Perencanaan: Tim penguji berkomunikasi dengan pemilik sistem untuk memahami lingkungan dan menentukan jangkauan serta tujuan dari pengujian.
2. Pengumpulan Informasi (Reconnaissance): Mengumpulkan informasi terkait sistem yang akan diuji, termasuk alamat IP, layanan yang berjalan, dan informasi publik lainnya.
3. Analisis Kerentanan: Menggunakan alat dan teknik untuk menganalisis kerentanan dalam sistem. Ini mencakup pemindaian untuk menemukan kelemahan umum seperti SQL Injection atau Cross-Site Scripting (XSS).
4. Eksploitasi: Setelah mengidentifikasi kerentanan, tim melakukan eksploitasi untuk membuktikan bahwa kerentanan tersebut dapat dieksploitasi. Ini mungkin melibatkan teknik seperti brute force atau privilege escalation.
5. Pelaporan Hasil: Menyusun laporan hasil pengujian yang mencakup temuan kerentanan, risiko terkait, dan rekomendasi perbaikan.
6. Perbaikan dan Verifikasi: Berdasarkan laporan hasil pengujian, perusahaan melakukan perbaikan untuk mengatasi kerentanan yang ditemukan.



CONTOH KASUS

Vulnerability Assessment and Penetration Testing on Student Service Center System

Dalam penelitian ini, penulis melakukan Vulnerability Assessment and Penetration Testing (VAPT) pada sistem Student Service Center di Universitas XYZ. Penelitian ini bertujuan untuk mengidentifikasi kerentanan keamanan dalam sistem yang menyimpan data sensitif dan pribadi mahasiswa.Hasil dari pengujian menunjukkan adanya beberapa kerentanan serius, termasuk:File Upload Functionality: Ditemukan risiko yang dapat dieksploitasi untuk serangan keamanan.

- Path Traversal Vulnerabilities: Kerentanan ini memungkinkan akses tidak sah ke direktori di luar root web.
- Unrestricted File Upload: Kerentanan ini memungkinkan penyerang meng-upload dan mengeksekusi skrip berbahaya di server web.

Penelitian ini menekankan pentingnya penerapan VAPT untuk mengidentifikasi dan mengatasi potensi ancaman sebelum dapat dimanfaatkan oleh penyerang.

References

[1] Yuni Widiastiwi, Ati Zaidiah, and Intan Hesti Indriana, “Pengujian Model Aplikasi User Interface E-Anjal Dengan Menggunakan Metode Black Box,” Informatik Jurnal Ilmu Komputer, vol. 16, no. 2, pp. 106–106, Aug. 2020, doi: https://doi.org/10.52958/iftk.v16i2.1980.

[2] Tasya Mina Alifia, Nugroho Purnomo Aji, A. A. Arsyad, and Lutfi Rahmatuti Maghfiroh, “Perbaikan User Interface Menggunakan Usability Testing dan Pendekatan Human-Centered Design,” Seminar Nasional Official Statistics, vol. 2021, no. 1, pp. 926–934, Nov. 2021, doi: https://doi.org/10.34123/semnasoffstat.v2021i1.760.

[3] Desy Intan Permatasari, “Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian,” Jurnal Sistem dan Teknologi Informasi (JustIN), vol. 8, no. 1, pp. 135–135, Jan. 2020, doi: https://doi.org/10.26418/justin.v8i1.34452.

[4] Alvino Dirgantara, “Stress Testing Si Penentu Ketahanan Sistem - ITGID | IT Governance Indonesia,” ITGID | IT Governance Indonesia, Apr. 13, 2020. https://itgid.org/insight/assessment/stress-testing-si-penentu-ketahanan-sistem/

[5] R. C. Sam, “Stress Testing: Pengujian Sistem untuk Strategi Bisnis,” Appsensi : Sistem Aplikasi HR, Software Payroll & Tarik Gaji, Mar. 30, 2023. https://appsensi.com/stress-testing/

[6] Jiawei Tyler Gu et al., “Acto: Automatic End-to-End Testing for Operation Correctness of Cloud System Management,” Oct. 2023, doi: https://doi.org/10.1145/3600006.3613161.

[7] Telkomuniversity.ac.id, 2024. https://repository.telkomuniversity.ac.id/pustaka/173042/analisis-implementasi-behavior-driven-development-bdd-melalui-kasus-uji-pada-aplikasi-web-sentratekstil.html

[8] M. A. Chamida, A. Susanto, and A. Latubessy, “ANALISA USER ACCEPTANCE TESTING TERHADAP SISTEM INFORMASI PENGELOLAAN BEDAH RUMAH DI DINAS PERUMAHAN RAKYAT DAN KAWASAN PERMUKIMAN KABUPATEN JEPARA,” Indonesian Journal of Technology, Informatics and Science (IJTIS), vol. 3, no. 1, pp. 36–41, Dec. 2021, doi: https://doi.org/10.24176/ijtis.v3i1.7531.

[9] N. Luh, A. Ginasari, K. Wibawa, N. Kadek, and A. Wirdiani, “Pengujian Stress Testing API Sistem Pelayanan dengan Apache JMeter,” vol. 2, no. 3, 2021, Accessed: Nov. 10, 2024. [Online]. Available: https://media.neliti.com/media/publications/351376-pengujian-stress-testing-api-sistem-pela-b5df7c9d.pdf

[10] B. Saputra and A. Stefanie, “Automation Testing Api, Android, dan Website Menggunakan Serenity Bdd Pada Software Sistem Manajemen Rumah Sakit”, jiwa, vol. 9, no. 10, pp. 114-126, May 2023.

[11] Suprapto, E. (2021). User Acceptance Testing (UAT) Refreshment PBX Outlet Site BNI Kanwil Padang. Jurnal Civronlit Unbari.

[12] “Database Testing: What to Test For in a Database,” The Agile Data (AD) Method - Strategies for effective data-oriented development, Apr. 05, 2023. https://agiledata.org/essays/whattotest.html

[13] Syarif Hidayatulloh and Desky Saptadiaji, “Penetration Testing pada Website Universitas ARS Menggunakan Open Web Application Security Project (OWASP),” Jurnal Algoritma, vol. 18, no. 1, pp. 77–86, Aug. 2021, doi: https://doi.org/10.33364/algoritma/v.18-1.827.

Isnaini, K. N., Asyari, M. H., Amrillah, S. F., & Suhartono, D. (2024). Vulnerability Assessment and Penetration Testing on Student Service Center System. ILKOM Jurnal Ilmiah, 16(2), 161-171

Thank You