TUGAS 02

1. Misalkan Anda adalah Quality Assurance Engineer yang bertugas untuk menguji dan memastikan kualitas sebuah aplikasi mobile perbankan yang baru dikembangkan oleh tim pengembangan perusahaan XYZ. Aplikasi ini bertujuan untuk memberikan layanan perbankan digital kepada nasabah, termasuk fitur-fitur seperti cek saldo, transfer dana, dan pembayaran tagihan. Jelaskan langkah-langkah dalam Software Testing Life Cycle (STLC) yang Anda terapkan dalam menguji dan memastikan kualitas aplikasi mobile perbankan ini!

Jawaban :

Langkah-langkah *Software Testing Life Cycle* (STLC) untuk menguji dan memastikan kualitas aplikasi *mobile* perbankan :

A. *Requirements Analysis*

I. Siapkan daftar pertanyaan dan bekerja sama dengan Client, Technical Manager dan Technical Lead, dll.

Untuk proses ini membutuhkan dokumen seperti SRS (*Software Reqiurement Spesification*) atau dokumen serupa yang berisi detail tentang aplikasi.

a. Cek Saldo
   - Bagaimana fitur 'Cek Saldo' harus bekerja?
   - Apakah perlu *login* akun dahulu untuk dapat melakukan fitur 'Cek Saldo'?
   - Bagaimana interaksi antara fitur 'Cek Saldo' dengan fitur 'Transfer Dana' dan 'Pembayaran Tagihan'? Saat melakukan 'Transfer Dana' dan 'Pembayaran Tagihan' bagaimana *update* saldo pada 'Cek Saldo'?
   - Apakah perlu keamanan seperti *password* transaksi untuk melakukan 'Cek Saldo'?
   - Bagaimana tampilan fitur 'Cek Saldo'? Apa informasi yang ditampilkan saat 'Cek Saldo'?
   - Saldo yang ditampilkan dalam mata uang apa?
   - Jenis Saldo yang ditampilkan apa saja? Apakah saldo tabungan saja?

b. Transfer Dana
   - Bagaimana fitur 'Transfer Dana' harus bekerja?
   - Apakah perlu *login* akun dahulu untuk dapat melakukan fitur 'Transfer Dana'?
   - Apakah perlu keamanan seperti *password* transaksi untuk melakukan 'Transfer Dana'?
   - 'Transfer Dana' yang dilakukan apakah antar apa saja? Antar Rekening sesame Bank atau Antar Bank juga bisa?
   - Syarat jika berhasil melakukan 'Transfer Dana'?
   - Apa saja *error* atau kegagalan saat melakukan 'Transfer Dana'?

- Apakah ada penyimpanan bukti ‘Transfer Dana’ dan *update* Saldo ketika fitur ‘Transfer Dana’ sukses?

c. Pembayaran Tagihan
   - Bagaimana fitur ‘Pembayaran Tagihan’ harus bekerja?
   - Apakah perlu *login* akun dahulu untuk dapat melakukan fitur ‘Pembayaran Tagihan’?
   - Apakah perlu keamanan seperti *password* transaksi untuk melakukan ‘Pembayaran Tagihan’?
   - Pada ‘Pembayaran Tagihan’ apa saja daftar tagihan yang tersedia?
   - Syarat jika berhasil melakukan ‘Pembayaran Tagihan’?
   - Apa saja *error* atau kegagalan saat melakukan ‘Pembayaran Tagihan’?
   - Apakah ada penyimpanan bukti ‘Pembayaran Tagihan’ dan *update* Saldo ketika fitur ‘Transfer Dana’ sukses?

II. Menentukan fokus pengujian dan prioritas pengujian.

Dari ketiga fitur (Cek Saldo, Transfer Dana, dan Pembayaran Tagihan) menentukan focus pengujian dan prioritas pengujiannya apa. Misalnya : agar dapat melakukan ‘Transfer Dana’ dan ‘Pembayaran’ harus dipastikan saldo mencukupi. Oleh karena itu, perlu dipastikan Saldo mencukupi dengan melakukan pengecekan pada fitur ‘Cek Saldo’.

III. Buat daftar rincian requirements yang dapat diuji

Rincikan setiap *requirement* yang akan diiuji dari aplikasi *mobile* perbankan.
Contoh:

- Melakukan Cek Saldo dengan memasukkan *password* transaksi yang benar dan yang salah
- Melakukan Transfer Dana untuk sesama Bank dan Bank yang berbeda.
- Melakukan Pembayaran Tagihan seperti Tagihan PDAM ataupun listrik PLN

IV. Siapkan tool automation yang akan digunakan dan report akhir

Jika ingin melakukan otomasi testing pada aplikasi perbankan, persiapkan tool *automation* yang akan digunakan

B. *Test Planning*

Setelah requirement diketahui maka selanjutnya membuat rencana pengujian atau test planning. Test planning akan membantu proses pengujian lebih terstruktur, terukur dan terarah.

Berikut hal-hal yang harus diperhatikan ketika menyusun test planning:

I. Menentukan berbagai tipe testing yang akan digunakan.

Pemilihan berbagai tipe pengujian yang akan digunakan dalam menguji aplikasi mobile perbankan harus didasarkan pada tujuan pengujian, cakupan pengujian, serta risiko dan kebutuhan kualitas aplikasi. Berikut beberapa tipe pengujian yang relevan untuk aplikasi mobile perbankan:

1. Pengujian Fungsional (Functional Testing):

   - **Uji Kasus Fungsional**: Menguji fitur-fitur utama aplikasi seperti cek saldo, transfer dana, pembayaran tagihan, dan lainnya sesuai dengan persyaratan fungsional.
   - **Uji Kasus Penggunaan (Usability Testing)**: Menguji seberapa mudah aplikasi digunakan oleh pengguna akhir dalam hal antarmuka pengguna, navigasi, dan pengalaman pengguna keseluruhan.

2. Pengujian Integrasi (Integration Testing):

   - **Pengujian Antarmuka (Interface Testing**): Menguji interaksi antara berbagai komponen aplikasi, termasuk koneksi ke sistem backend perbankan dan integrasi dengan layanan pihak ketiga seperti sistem pembayaran.

3. Pengujian Kinerja (Performance Testing):

   - **Uji Beban (Load Testing)**: Menguji kinerja aplikasi dalam situasi beban yang tinggi untuk memastikan bahwa aplikasi tetap responsif dan stabil.
   - **Uji Kinerja Aplikasi (Application Performance Testing)**: Menguji kinerja aplikasi dalam berbagai kondisi, termasuk saat koneksi internet yang lambat atau tidak stabil.

4. Pengujian Keamanan (Security Testing):

   - **Pengujian Keamanan Aplikasi (Application Security Testing)**: Menguji aplikasi untuk mengidentifikasi potensi kerentanan keamanan, termasuk perlindungan terhadap serangan seperti SQL injection atau cross-site scripting (XSS).
   - **Pengujian Otorisasi (Authorization Testing)**: Memastikan bahwa akses ke fitur-fitur tertentu terbatas pada pengguna yang berwenang.

5. Pengujian Kompatibilitas (Compatibility Testing):

   - **Uji Kompatibilitas Perangkat (Device Compatibility Testing)**: Menguji aplikasi pada berbagai perangkat mobile (smartphone dan tablet) dan sistem operasi yang berbeda.

- **Uji Kompatibilitas Browser (Browser Compatibility Testing)**: Jika ada versi web aplikasi, memastikan bahwa aplikasi berfungsi dengan baik di berbagai peramban web.

6. Pengujian Penerimaan Pengguna (User Acceptance Testing - UAT):

   - **Pengujian Penerimaan Pengguna (User Acceptance Testing)**: Melibatkan pengguna akhir atau pemegang kepentingan untuk menguji aplikasi dan memastikan bahwa itu memenuhi ekspektasi mereka sebelum peluncuran.

7. Pengujian Pemulihan Bencana (Disaster Recovery Testing):

   - **Pengujian Pemulihan Bencana (Disaster Recovery Testing)**: Memastikan bahwa sistem dan data perbankan dapat dipulihkan dengan cepat dalam situasi bencana atau kegagalan sistem.

8. Pengujian Pembaruan dan Pemeliharaan (Regression Testing):

   - **Pengujian Regresi (Regression Testing)**: Memeriksa apakah pembaruan, perbaikan, atau perubahan aplikasi tidak mengganggu fungsionalitas yang sudah ada sebelumnya.

9. Pengujian Aksesibilitas (Accessibility Testing):

   - **Pengujian Aksesibilitas (Accessibility Testing)**: Memastikan bahwa aplikasi dapat diakses dan digunakan dengan baik oleh individu dengan disabilitas, seperti penyandang cacat visual yang menggunakan pembaca layar.

II. Menentukan teknik pendekatan testing yang sesuai. Ada dua teknik pendekatan testing:

a. Proactive: pendekatan testing dimana proses tes dimulai sedini mungkin untuk menemukan bug.
b. Reactive: pendekatan testing dimana proses tes akan dimulai ketika implementasi secara keseluruhan selesai.

III. Pemilihan test tool yang sesuai kebutuhan. Contohnya spreadsheet untuk management test cases

IV. Estimasi waktu dan cost yang diperlukan.

a. Menentukan sumberdaya yang dibutuhkan dan tugas-tugas ketika testing.

V. Membuat dokumen test planning.

C. *Test Case Development*

Test case development adalah proses pembuatan test case yang akan diterapkan pada proses pengujian software. Hal yang perlu diperhatikan dalam pembuatan test case antara lain:

I. Informasi tentang Fitur yang Diuji:

- Ini adalah deskripsi umum tentang fitur atau fungsionalitas yang akan diuji. Informasi ini harus mencakup nama fitur, deskripsi singkat tentang apa yang fitur tersebut lakukan, dan konteksnya dalam aplikasi.
- Contoh: "Fitur: Transfer Dana Antar Rekening"

II. Acceptance Criteria (Kriteria Penerimaan):

- Kriteria penerimaan adalah kriteria atau kondisi yang harus dipenuhi agar fitur atau fungsi dianggap berhasil atau sesuai dengan harapan. Ini membantu memastikan bahwa hasil pengujian sejalan dengan harapan pengguna dan pemangku kepentingan.
- Contoh: "Kriteria Penerimaan: Transfer dana harus berhasil jika saldo mencukupi, dan saldo sumber rekening harus dikurangi sejumlah dana yang ditransfer."

III. Langkah-langkah (Steps):

- Ini adalah langkah-langkah yang harus diikuti oleh penguji untuk menguji fitur atau fungsi tersebut. Langkah-langkah ini harus dijelaskan dengan jelas, langkah demi langkah, sehingga penguji dapat mengikutinya dengan mudah.
- Contoh:

Langkah-langkah: 1. Buka aplikasi mobile perbankan. 2. Masuk ke akun pengguna. 3. Pilih menu "Transfer Dana." 4. Masukkan jumlah dana yang akan ditransfer dan nomor rekening tujuan. 5. Konfirmasi transfer. 6. Verifikasi bahwa transfer berhasil dan saldo sumber rekening telah dikurangi sesuai dengan jumlah yang ditransfer.

IV. Data yang Dibutuhkan atau Dihasilkan dari Test Case:

- Ini mencakup data yang diperlukan untuk menjalankan test case dan data yang diharapkan dihasilkan sebagai hasil dari test case.
- Data yang dibutuhkan bisa mencakup masukan pengguna, konfigurasi aplikasi, atau kondisi awal tertentu.
- Data yang dihasilkan bisa berupa pesan sukses/gagal, perubahan dalam basis data, atau perubahan dalam tampilan antarmuka pengguna.
- Contoh:
    - Data yang Dibutuhkan: Saldo awal sumber rekening, nomor rekening tujuan, jumlah dana yang akan ditransfer.
    - Data yang Dihasilkan: Pesan sukses transfer atau pesan kesalahan jika transfer gagal.

Perhatikan management test cases ketika membuat test case, hal ini akan sangat bermanfaat apabila dari proses inisiasi sudah memiliki format management yang baik. Contohnya buatlah database test cases di sebuah platform misalnya spreadsheet atau gunakan tool management test cases yang ada di internet, mereka akan membantu sekali untuk proses pelacakan, dokumentasi dan versioning test case.

D. *Test Environment Setup*

Pada tahap ini, penguji akan mempersiapkan segala sesuatu yang dibutuhkan untuk menjalankan smoke testing pada aplikasi perbankan mobile. Smoke testing adalah pengujian yang dilakukan untuk memastikan bahwa build atau versi terbaru dari aplikasi dapat berjalan tanpa adanya masalah besar atau "blocker". Berikut adalah langkah-langkah yang perlu diambil:

I. Hardware dan Perangkat Lunak:

- Pastikan bahwa perangkat keras (misalnya, komputer atau perangkat mobile) yang akan digunakan untuk pengujian siap digunakan dan dalam kondisi baik.
- Pastikan sistem operasi perangkat yang digunakan sesuai dengan persyaratan aplikasi mobile perbankan.

II. Koneksi Internet:

- Pastikan koneksi internet yang digunakan untuk pengujian stabil dan dapat diandalkan.
- Pastikan bahwa Anda memiliki akses internet yang cukup untuk mengunduh build atau versi terbaru dari aplikasi jika diperlukan.

III. VPN (Jika Diperlukan):

- Jika aplikasi perbankan mobile memerlukan akses melalui jaringan VPN, pastikan Anda terhubung ke VPN yang sesuai dengan lingkungan pengujian yang ditargetkan.

IV. Perangkat Mobile (Jika Diperlukan):

- Jika pengujian akan dilakukan pada perangkat mobile fisik, pastikan perangkat mobile tersebut tersedia dan dalam kondisi baik.
- Pastikan perangkat telah diatur untuk debugging USB jika diperlukan.

V. Software Aplikasi:

- Pastikan bahwa build atau versi terbaru dari aplikasi mobile perbankan sudah siap digunakan. Ini bisa berupa APK (untuk Android) atau IPA (untuk iOS) yang telah disiapkan oleh tim pengembangan.
- Instal aplikasi tersebut di perangkat yang akan digunakan untuk pengujian.

VI. Dokumentasi Smoke Test Result:

- Setelah semua persiapan selesai, jalankan smoke testing pada aplikasi mobile perbankan.
- Selama pengujian, catat hasil smoke test result. Ini harus mencakup informasi tentang apakah ada masalah besar atau "blocker" yang menghalangi aplikasi dari berfungsi dengan baik.
- Jika tidak ada masalah besar yang ditemukan, hasil smoke test harus mencatat bahwa build tersebut berhasil lulus smoke test.

Contoh Dokumentasi Smoke Test Result:

```
======================
Smoke Test Result
======================
- Tanggal Pengujian: 21 September 2023
- Penguji: Ercherio Galang Dameross Marpaung

Hasil Smoke Test:
- Aplikasi berhasil diunduh dan diinstal pada perangkat.
- Aplikasi dapat diluncurkan tanpa adanya crash atau error.
- Fungsionalitas utama seperti cek saldo, transfer dana, dan pembayaran tagihan dapat diakses dan digunakan tanpa masalah.
- Tidak ada masalah besar atau "blocker" yang ditemukan pada build ini.

Kesimpulan: Build telah berhasil lulus smoke test dan siap untuk dilanjutkan dengan pengujian lebih lanjut.

Tanda Tangan Penguji: _____
```

Dengan melakukan smoke testing dan mendokumentasikan hasilnya, Anda dapat memastikan bahwa build atau versi terbaru dari aplikasi mobile perbankan sudah lulus tahap awal pengujian dan siap untuk diuji lebih lanjut untuk mengidentifikasi masalah lebih lanjut atau perbaikan yang diperlukan sebelum peluncuran.

E. *Test Execution Phase*

Biasanya terdiri dari langkah-langkah berikut:

I. Eksekusi Pengujian Awal:

Pada tahap ini, tim penguji menjalankan rencana pengujian yang telah disiapkan sebelumnya. Mereka mengikuti langkah-langkah yang telah ditentukan dalam rencana pengujian untuk menguji berbagai fitur dan fungsionalitas aplikasi.
Selama pengujian, penguji mencatat setiap bug atau masalah yang ditemukan, serta detail tentang bagaimana masalah tersebut terjadi.

II. Pelaporan Bug:

Setelah menemukan bug atau masalah, penguji mengumpulkan informasi lengkap tentang bug tersebut, termasuk langkah-langkah untuk memreproduksi masalah, hasil yang diharapkan, dan hasil yang sebenarnya.
Bug tersebut dilaporkan ke dalam sistem manajemen bug atau alat pelaporan yang digunakan oleh tim pengembang. Bug biasanya diberi status, prioritas, dan penugasan kepada pengembang yang bertanggung jawab.

III. Perbaikan Bug:

Tim pengembang menerima laporan bug dan mulai menganalisis masalah yang dilaporkan.
Setelah bug ditemukan, pengembang memperbaikinya dengan melakukan perubahan pada kode sumber aplikasi.
Setelah perbaikan selesai, pengembang mengirimkan kode perbaikan untuk diuji ulang.

IV. Pengujian Bug Ulang (Regression Testing):

Tim penguji menerima kode perbaikan dari pengembang dan melakukan pengujian ulang pada area yang terkena dampak oleh bug yang diperbaiki.
Selain itu, penguji juga melakukan pengujian regresi untuk memastikan bahwa perbaikan tersebut tidak mempengaruhi fungsionalitas lain dalam aplikasi.
Jika bug tidak terjadi kembali dan tidak ada masalah baru yang muncul, bug dinyatakan selesai.

V. Siklus Berulang:

Proses pelaporan, perbaikan, dan pengujian bug berulang hingga seluruh bug yang telah diidentifikasi diperbaiki dan aplikasi dinyatakan layak untuk penggunaan.
Pada setiap iterasi, penguji akan mengulangi pengujian untuk memastikan bahwa bug yang telah diperbaiki tidak muncul kembali dan untuk mengidentifikasi masalah baru jika ada.

VI. Akhir Pengujian:

Setelah seluruh bug diperbaiki, dan aplikasi telah menjalani serangkaian pengujian yang sukses, aplikasi dianggap layak untuk digunakan oleh pengguna akhir.
Aplikasi siap untuk peluncuran, dan tim pengujian dapat memberikan rekomendasi kepada manajemen terkait kesiapan aplikasi.

F. *Test Cycle Closure*

Berikut beberapa poin yang perlu diperhatikan dalam tahap ini:

I. Pembuatan Laporan Pengujian:

- Laporan pengujian adalah dokumen yang berisi semua informasi penting tentang pengujian yang telah dilakukan. Ini mencakup ringkasan pengujian, temuan bug, hasil pengujian, dan metrik pengujian.
- Laporan pengujian harus disusun dengan rapi dan jelas agar dapat dengan mudah dipahami oleh pihak-pihak terkait, termasuk manajemen, tim pengembang, dan klien.

II. Pengumpulan Matriks:

- Matriks pengujian adalah data dan statistik yang menggambarkan hasil pengujian secara numerik. Ini termasuk jumlah bug yang ditemukan, tingkat keparahan, status bug, cakupan pengujian, dan lain-lain.
- Matriks pengujian membantu dalam mengevaluasi kualitas perangkat lunak dan efektivitas pengujian.

III. Analisis dan Retrospektif:

- Dalam tahap ini, tim pengujian dan tim pengembang dapat melakukan analisis bersama tentang hasil pengujian dan masalah-masalah yang muncul selama proses pengujian.
- Diskusi retrospektif dapat membantu dalam mengidentifikasi penyebab masalah, memahami pelajaran yang dapat dipetik, dan merencanakan perbaikan untuk pengujian di masa mendatang.

IV. Dokumen Referensi:

- Laporan pengujian dan matriks pengujian dapat dijadikan dokumen referensi yang berguna untuk klien atau pemangku kepentingan. Ini memberikan gambaran tentang bagaimana aplikasi telah diuji dan kualitasnya.

V. Rekomendasi Perbaikan:

- Berdasarkan hasil analisis dan retrospektif, tim pengujian dapat memberikan rekomendasi perbaikan untuk proses pengujian di masa mendatang. Ini termasuk

perubahan dalam strategi pengujian, peningkatan dalam alat pengujian, atau perbaikan dalam kolaborasi antara tim pengujian dan tim pengembang.

VI. Penutupan Pengujian:

- Setelah semua dokumen dan matriks telah disiapkan, tahap STLC dapat ditutup dengan menyampaikan hasil pengujian dan laporan kepada pihak yang berkepentingan.
- Ini juga merupakan kesempatan untuk memutuskan apakah perangkat lunak telah siap untuk peluncuran atau perlu pengujian lebih lanjut.

G. *Exit – Entry Criteria*

Exit dan Entry Criteria adalah dua konsep penting dalam manajemen pengujian perangkat lunak yang digunakan untuk mengatur kapan sebuah fase atau siklus pengujian harus dimulai (Entry Criteria) dan kapan harus dianggap selesai (Exit Criteria). Kedua kriteria ini membantu dalam pengelolaan dan pengendalian proses pengujian.

Fase	Entry Criteria	Exit Criteria
Requirement Analysis	• Requirements specification document • Acceptance criteria document • Technical Specification	• Requirement traceability matrix (RTM) • Automation Feasibility Document
Test Planning	• Requirement Documents • Requirement Traceability Matrix Document	• Test Plan document • Testing Techniques • Timeline • Cost • Risks • Resource allocation
Test Case Development	• Test Plan document • Timeline • Cost • Risks • Resource allocation	• Test Case Documentation/Test Script • Identified Test Data
Test Environment Setup	• Test strategy and test plan document • Test case document • Testing data	• Environment ready with test data set up • Smoke Test Report
Test Execution Phase	• Test strategy and test plan document. • Test case document. • Testing data. • Testing Environment • Testing Tool	• Test Case -Execution Result • RTM with execution status • Defect Report
Test Cycle Closure	• Test Execution Report, Defect Report	• Test Closure Report • Summary Report

2. Misalkan Anda adalah Quality Assurance Engineer. Anda diberikan tugas untuk melakukan tes sebuah program matematika sederhana yang akan menghitung akar pangkat dua dari suatu angka. Buatlah Behavior Driven Development (BDD) untuk menguji dan mengembangkan fungsi atau program ini.

   Hint : Perhatikan teori-teori matematika dalam proses perhitungan akar pangkat

   Jawaban :

   Behavior Driven Development (BDD) adalah pendekatan pengembangan perangkat lunak yang berfokus pada perilaku atau fungsi yang diinginkan dari perangkat lunak. Dalam kasus ini, kita akan mengembangkan dan menguji sebuah program matematika sederhana yang menghitung akar pangkat dua dari suatu angka.

   Fitur: Menghitung Akar Pangkat Dua
   Sebagai seorang pengguna program kalkulator
   Saya ingin dapat menghitung akar pangkat dua dari suatu angka
   Agar saya bisa dengan cepat mendapatkan hasilnya

   Scenario 1: Menghitung Akar Pangkat Dua dari Angka Positif
   Diberikan angka positif X
   Ketika saya menghitung akar pangkat dua dari X
   Maka hasilnya haruslah akar pangkat dua dari X

   Contoh:

   - **Diberikan** angka positif 4
   - **Ketika** saya menghitung akar pangkat dua dari 4
   - **Maka** hasilnya haruslah 2

   Scenario 2: Menghitung Akar Pangkat Dua dari Nol
   Diberikan angka 0
   Ketika saya menghitung akar pangkat dua dari 0
   Maka hasilnya haruslah 0

   Scenario 3: Menghitung Akar Pangkat Dua dari Angka Negatif
   Diberikan angka negatif -9
   Ketika saya menghitung akar pangkat dua dari -9
   Maka program harus memberikan pesan kesalahan bahwa akar pangkat dua dari angka negatif tidak valid

Scenario 4: Menghitung Akar Pangkat Dua dari Angka Pecahan
Diberikan angka pecahan 0.25
Ketika saya menghitung akar pangkat dua dari 0.25
Maka hasilnya haruslah akar pangkat dua dari 0.25

Contoh:

- **Diberikan** angka pecahan 0.25
- **Ketika** saya menghitung akar pangkat dua dari 0.25
- **Maka** hasilnya haruslah 0.5