**RESUME PERBAIKAN BUKU QA**

* KELOMPOK

Dari hasil review kelompok ini, banyak penulisan yang masih belum sesuai. Berikut hasil perbaikan dari kelompok ini, yaitu:

**BAB VI  
(REGRESSION TESTING)**

1. Test Plans

Tujuan membuat *test plan* secara umum adalah untuk memudahkan developer untuk melakukan *testing*

Perbaikan:

untuk 🡪 dalam

testing 🡪 testing,

1. *Test Plan Identifier*

Tindormasiyang dijelaskan dapat berupa sekilas mengenai subjek *testing*, nama orang yang bertanggung jawab terhadap *testing*, penyusun *test plan*, tanggal dibuat *test plan* dan tanggal revisi, dll.

Perbaikan:

dan 🡪 ,

1. *Feature to be tested*

Penjelasan dan daftar-daftar fitur yang akan ditest di pada saat pelaksanaan *testing* dimulai.

Perbaikan:

di 🡪 (hapus)

1. *Approach/Test Strategy*

Bagian *Approach* adalah bagian yang digunakan untuk memberi deskripsi mengenai cara yang dilakukan untuk melaksanakan *testing* dan disertakan dengan penjelasan mengenai cara yang digunakan.

Perbaikan:

yang digunakan 🡪 penggunaannya

1. *Suspension Criteria*

Berisi tentang spesifikasi kriteria-kriteria yang dapat digunakan untuk menghentikan sementara kegiatan testing dan testing tersebut dapat dilanjutkan di waktu lain.

Perbaikan:

testing 🡪 testing,

1. What is Regressoon Testing?

Pengujian regresi tidak lain adalah seleksi penuh atau sebagian dari kasus uji yang sudah dieksekusi yang dieksekusi ulang untuk memastikan fungsionalitas yang ada berfungsi dengan baik.

Perbaikan:

yang 🡪 , kemudian

berfungsi 🡪 apakah sudah berfungsi

Pengujian ini dilakukan untuk memastikan bahwa perubahan kode baru tidak memiliki efek samping pada fungsionalitas yang ada. Ini memastikan bahwa kode lama masih berfungsi setelah perubahan kode baru selesai.

Perbaikan:

ini 🡪 hal ini

baru 🡪 terbaru

Kebutuhan pengujian regresi terutama timbul setiap kali ada kebutuhan untuk mengubah kode dan kita perlu menguji apakah kode dimodifikasi mempengaruhi bagian lain dari aplikasi perangkat lunak atau tidak.

Perbaikan:

dimodifikasi 🡪 yang dimodifikasi

Setelah *bug* diidentifikasi, perubahan yang diperlukan dibuat untuk memperbaikinya, kemudian pengujian regresi dilakukan dengan memilik kasus uji yang relevan dari rangkaian pengujian yang mencakup bagian kode yang dimodifikasi dan terpengaruh.

Perbaikan:

perubahan yang 🡪 (dihapus)

Dibuat 🡪 perubahan

Memperbaikunya 🡪 memperbaiki perangkat lunak

Prioritas kasus uji tergantung pada dampak bisnis, fungsi penting & sering digunakan.

Perbaikan:

Sering 🡪 yang sering

Uji regresi efektif dapat dilakukan dengan memilih kasus uji berikut

Perbaikan:

Berikut 🡪 sebagai berikut

1. Positive Testing and Negative Testing
2. Positive testing

Pengujian ini berupa menguji inputan nilai yang valid.

Perbaikan:

berupa 🡪 pengujian untuk

1. Negative testing

Menginput nilai selain dari angka seperti nilai A-Z/a-z pada kolom *input number* maka aplikasi akan memberikan berupa *alert*/pemberitahuan bahwa inputan tersebut *invalid*/salah.

Perbaikan:

menginput 🡪 jika user menginputkan

1. Boundary Value Analysis

Pengujian positif pengujian berupa data input berada dalam batas nilai batas.

Perbaikan:

pengujian 🡪 merupakan pengujian

berada 🡪 yang berada

pengujian negative adalah pengujian berupa data input berada di luar batas nilai batas.

Perbaikan:

berada 🡪 yang berada

setiap partisi memiliki nilai maksimum dan minimum dan nilai maksimum dan minimum nini adalah nilai batas dari sebuah partisi.

Perbaikan:

Dan 🡪 ,

Minimum 🡪 nilai minimum

Jika sebuah sistem hanya menerima nilai inpitan untuk umur hanya dari 18 sampai 56. Oleh karena itu pengujian batas sebagaimana tabel 6.1.

Perbaikan:

Hanya 🡪 mulai

pengujian 🡪 , pengujian

1. Equivalence Partitioning
2. Pembahasan What is The Case

*Equivalence partitioning* juga dikenal sebagai *Equivalence Class Partitioning (ECP)*. Ini adalah teknik pengujian perangkat lunak atau pengujian *black-box* yang membagi domain *input* ke dalam kelas data, dan dengan bantuan kelas data ini, kasus uji dapat diturunkan.

Perbaikan:

ini 🡪 ECP

dan 🡪 (hapus)

**BAB VII  
WEB ARCHITECTURE**

1. Modern Web Architecture Explained

Yaitu proses pada bagian *user* yang disebut *front-end* dan proses pada bagian yang tidak dapat dilihat oleh user disebut *back-end*.

Perbaikan:

disebut 🡪 yang disebut