

RESUME PERBAIKAN BUKU QA

- **KELOMPOK DARA MELISA**

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

I. Bab 2

a) Pembahasan Security Testing

1) *Assessment*, yaitu analisis keamanan untuk persyaratan dan **periksa** kasus penyalahgunaan.

Perbaikan:

periksa → pemeriksaan

4) *Testing*, yaitu pengujian *black box*, pengujian statis dan dinamis **dan** pengujian *white box*, dan pemindaian kerentanan.

Perbaikan:

dan → ,

b) Pembahasan Monkey Testing (Ad-hoc)

Setelah bug ditemukan, maka akan dilakukan tangkapan layar **yang mana** cara dalam menangkap layar **tergantung pada** sistem yang digunakan.

Perbaikan:

yang mana → dengan

tergantung → sesuai dengan

II. Bab 3

a) Pembahasan How To report Bugs

1. *Type*

Pilih jenis masalah *bug*, **yang mana** tergantung pada proyek yang telah disiapkan di JIRA. Jenis masalah dapat berisi nilai berikut: *bug*, *story*, *epic*, *task* dan sebagainya untuk melaporkan kerusakan.

Perbaikan:

yang mana → pilihan bug

2. *Summary*

Menulis ringkasan singkat tentang apa dan dimana **yang salah**. Pesan yang dapat ditulis di *summary* yaitu [Authorize.net] Verification is missing.

Perbaikan:

yang salah → terjadinya kesalahan

3. *Description*

Deskripsi ini adalah bidang yang paling penting ketika membuat laporan bug, yang mana deskripsi ini sedikit lebih deskriptif **dari dari** ringkasan (summary).

Perbaikan:

dari dari → daripada

4. *Screenshot*

Screenshot jendela Authorize.Net **untuk** lebih terpercaya dan aman. Setelah itu, masukkan file image di attachment.

Perbaikan:

untuk → dilakukan agar

b) Pembahasan Bug Triage

Pada *bug* yang memiliki tingkat resiko kegagalan yang **besar** akan lebih diprioritaskan perbaikannya daripada *bug* yang tingkat resiko **kegagalannya** kecil.

Perbaikan:

Penambahan **koma** didepan kata besar dan penambahan kata **lebih** di depan kata keagalannya

Namun, hal **ini** tidak menjamin perbaikan secara total pada perangkat lunak

Perbaikan:

ini → tersebut juga

c) Pembahasan Severity and Priority

Sedangkan, *priority* dapat **didefinisikan** rentetan bug

Perbaikan:

Ada penambahan kata sebagai didepan kata didefinisikan

Kedua istilah tersebut saling melengkapi satu sama **lainnya**

Perbaikan:

lainnya → lain

pada *bug* dapat lebih terorganisir **dan** tepat

Perbaikan:

dan → dengan

III. Bab 4

a) Pembahasan What is The Case

Kasus uji dapat dijelaskan sebagai serangkaian daftar tes atau pengujian yang harus dieksekusi **atau** dijalankan

Perbaikan:

atau → ,

Hasil dari **pengujian** menggunakan kasus uji

Perbaikan:

Adanya penambahan kata **dengan** didepan kata pengujian

b) Pembahasan How to Write Test Cases

Isi template laporan hasil uji **berupa** kolom modul proses yang akan diuji

Perbaikan:

berupa → berisi

c) Pembahasan Smoke Testing

Ini terdiri dari serangkaian

Perbaikan

Ini → yang

Smoke Testing berarti memverifikasi bahwa fitur-fitur **penting** berfungsi dan tidak ada *showstoppers*

Perbaikan:

Ada penambahan kata **pada perangkat lunak** didepan kata penting

Ini adalah tes regresi

Perbaikan:

Ini adalah → pengujian ini termasuk ke dalam tes regresi

fungsionalitas **utama**. **Ini** adalah tes sederhana yang menunjukkan

utama. **Ini** → **tanda titik** dihilangkan dan kata **ini** diganti dengan kalimat **atau disebut dengan**

untuk **pengujian**. **Ini** membantu

Perbaikan:

utama. **Ini** → tanda titik diganti dengan **tanda koma** dan kata ini diganti dengan kalimat **hal ini akan**

d) Pembahasan How to Write Smoke Suite

Kwidos **Ini** adalah platform yang **menghubungkan** kontraktor dan konsumen

Perbaikan:

Kata **ini** dihilangkan dan ada penambahan kata **antara** di depan kata menghubungkan

Setelah tombol registrasinya **di-klik** maka

Perbaikan:

ada penambahan **tanda koma** didepan kata di-klik

Ketika proses pendaftaran akun berhasil **dilakukan**, akan

Perbaikan:

Ada penambahan kata **maka** didepan kata lakukan

IV. Bab 5

a) Pembahasan How to Test on Mobile Devices

Selain emulator, **juga terdapat** layanan *cloud* khusus

Perbaikan:

Kata **juga** di belakang terdapat dipindahkan kedepan terdapat

Klik kanan pada area mana saja di *website* tersebut kemudian klik Inspeksi **(F12)**.

Perbaikan:

(F12) → ada penambahan kata **atau klik** (F12) **pada keyboard**

Tampilan *mobile* hanya muncul pada *website* **yang** fitur Inspeksi

Perbaikan:

yang → jika

b) Pembahasan Modern Browsers

terdapat Chrome, Firefox, Opera, Maxton dan **banyak lainnya**

Perbaikan:

banyak lainnya → sebagainya

melaikan *browser* yang digunakan pelanggan

Perbaikan:

melaikan → melainkan

c) Pembahasan How to Test Android Apps Without A Device

Android Studio memiliki *emulator* yang sudah terpasang **dalam Android Studio**

Perbaikan:

dalam Android Studio → didalamnya

d) Pembahasan How to Test Using Cloud Sevices

Jadi seperti menggunakan perangkat nyata yang sebenarnya terletak disuatu tempat yang lokasi dari perangkat itu **sendiri** dapat diketahui oleh penggunanya

Perbaikan:

Ada penambahan **tanda koma** didepan kata **jadi** dan kata **sendiri** dihilangkan
Itu tentunya jauh lebih mahal dan bisa saja itu berkisar antara \$120 per bulan atau mungkin lebih

Perbaikan:

Itu → Hal tersebut

V. Bab 6

A. 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 → ,

2) *Feature to be tested*

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

Perbaikan:

di → (hapus)

3) *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

4) *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,

B. What is Regression 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 memilih kasus uji yang relevan dari rangkaian pengujian yang mencakup bagian kode yang dimodifikasi dan terpengaruh.

Perbaikan:

perubahan yang → (dihapus)

Dibuat → perubahan

Memperbaikinya → 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

C. Positive Testing and Negative Testing

a) Positive testing

Pengujian ini **berupa** menguji inputan nilai yang valid.

Perbaikan:

berupa → pengujian untuk

b) 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

c) 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** ini adalah nilai batas dari sebuah partisi.

Perbaikan:

Dan → ,

Minimum → nilai minimum

Jika sebuah sistem hanya menerima nilai inputan untuk umur **hanya** dari 18 sampai 56.

Oleh karena itu **pengujian** batas sebagaimana tabel 6.1.

Perbaiki:

Hanya → mulai

pengujian → , pengujian

d) Equivalence Partitioning

a) 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.

Perbaiki:

ini → ECP

dan → (hapus)

VI. Bab 7

A. 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*.

Perbaiki:

disebut → yang disebut

B. How to Debug Front-End or Back-End Issues?

a) Adapun jika membuat *inspect elemen jika* ingin membuat tombol *login bisa di klik*, dengan cara hapus *disable pada elemen button*.

Perbaiki :

membuat *inspect elemen jika* → dihilangkan

bisa di klik, → anda dapat membuatnya pada *inspect element*

dengan cara hapus *disable pada elemen button*. → kemudian pergi ke *element button* dan hapus *disable pada element button*.

b) Dari gambar 7.10 dapat diketahui bahwa proses validasi terhadap kolom *login masing* kurang bagus, dikarenakan harus melakukan proses secara *front-end* dan *back-end*. Alangkah lebih bagus menggunakan proses validasi seperti menggunakan *Ajax* atau pengecekan *javascript* terlebih dahulu sebelum melakukan proses *login*.

Perbaiki :

masing, → masih terlihat

Alangkah lebih bagus → Untuk memperbaikinya kita dapat

VII. Bab 8

a) What is Software Build?

Kemudian tim penguji perangkat lunak akan memeriksa aplikasi tersebut. Jika terdapat beberapa bug dan **jika** tidak memenuhi persyaratan, maka tim **pengujian** perangkat lunak menolak pengembangan aplikasi tersebut.

Perbaikan :

Kemudian, → Selanjutnya

Jika → dihilangkan

Pengujian → penguji

- **KELOMPOK ILHAM AFIF**

Dari hasil review kelompok ini, banyak penulisan yang masih belum sesuai.

Berikut hasil perbaikan dari kelompok ini, yaitu:

I. Kata pengantar

- a. segala puji bagi allah, tuhan yang maha esa atas ramhat dan karunia-nya, sehingga kami dapat menyelesaikan buku ajar ini.

⇒ Kami seharusnya diganti dengan penulis

- b. Buku ajar kami yang berjudul “software testing simple

⇒ Software testing simple digantikan ke tulisan atalic

1. Testing definition

- a. Error, software, error dan bug digantikan ke tulisan italic

2. Software defect

- a. Pada judul ini semua kata bug digantikan ke tulisan italic

3. User interface testing

- a. Kata design digantikan ke tulisan italic

4. Usability testing

- a. New away digantikan ke tulisan italic

- b. Kata dikomputer digantikan dengan kata di computer

5. How to report bugs

- a. Kata jira digantikan dengan kata JIRA

- b. Kata verification is missing digantikan ke tulisan italic italic

- c. File image di attachment digantikan ke tulisan italic

6. Severity and priority

7. Bug digantikan ke tulisan italic

II. Bab 4

- a) What is Test Case

Perbaikan:

What is Test Case → What is Test Case ?

- b) Why Test Cases is Important

Perbaikan:

Why Test Cases is Important → Why Test Cases is Important ?

- c) How to Write Test Cases

Perbaikan:

How to Write Test Cases → How to Write Test Cases ?

Isi **template** laporan hasil uji berupa kolom modul proses yang akan diuji, kasus uji, langkah-langkah proses terkait kasus uji, deskripsi atau penjelasan yang lebih merinci terkait kasus uji (hal-hal yang akan diuji), status pengujian, hasil uji yang diharapkan dan kolom hasil uji yang didapatkan.

Perbaikan:

template → *template*

Salah satu perangkat lunak yang populer digunakan untuk pengujian perangkat lunak adalah **zephyr**.

Perbaikan:

zephyr → Zephyr

d) Smoke Testing

Ini membantu menentukan apakah build tersebut cacat sehingga pengujian lebih lanjut akan membuang-buang waktu dan sumber daya.

Perbaikan:

build → *build*

e) How to Write Smoke Suite?

Kwidos **ini** adalah platform yang menghubungkan kontraktor dan konsumen.

Perbaikan:

ini → seharusnya kata “**ini**” dihilangkan

Berikut penjelasan tentang pengujian *smoke suite* yang dilakukan pada aplikasi kwidos.

Perbaikan:

kwidos → Kwidos

1. Halaman *Homepage / Layout*

- a. *Homepage* merupakan halaman berada yang pasti dijumpai oleh pengunjung pada situs **kwidos**

Perbaikan:

kwidos → Kwidos

Tampilan **kwidos** sangatlah umum dan tidak ada yang istimewa

Perbaikan:

kwidos → Kwidos

- b. Hal terpenting pada halaman beranda pada situs **kwidos** adalah **banner**/spanduk yang harus ditampilkan

Perbaikan:

kwidos → Kwidos

banner → *banner*

- c. Selanjutnya pengisian uji pada **smoke suite**.

Perbaikan:

Smoke suite → *smoke suite*

2. *Create a consumer account*

- a. Pengujian yang akan dilakukan adalah pembuatan akun sebagai **consumer**

Perbaikan:

consumer → consumer

Gambar 4.8 **Form** pendaftaran akun consumer pada situs Kwidos

Perbaikan:

Form → *Form*

2.3 Ketika proses pendaftaran akun **behasil** dilakukan, akan muncul pesan bahwa pembuatan akun telah sukses seperti gambar 4.9

Perbaikan:

behasil → berhasil

III. Bab 5

a) Pembahasan How to test on Mobile Devices?

Solusi dari permasalahan tersebut adalah menggunakan **simulator** dan **emulator** pengguna, salah satunya melalui *browser* Chrome.

Perbaikan:

simulator → *simulator*

emulator → *emulator*

Selain **emulator**, juga terdapat layanan *cloud* khusus, pengguna bisa membuat akun di sana, dan menggunakan komputer dasar melalui *browser*, pengguna dapat menavigasi ke situs web tersebut dan hanya memilih perangkat yang dibutuhkan seperti sistem operasi atau memilih versi.

Perbaikan:

emulator → *emulator*

b) Pembahasan How to use Chrome Mobile View?

Tampilan *website mobile* ditujukan untuk pengguna dengan perangkat *mobile* (*smartphone* & *tablet*), sementara *website desktop* adalah untuk perangkat PC (komputer desktop & **laptop**).

Perbaikan:

&laptop → & *laptop*

c) How to Test Android Apps Without A Device

Perbaikan:

How to test Android Appas Without A Device → How to test Android Appas Without A Device?

Dengan menggunakan *emulator* **andorid**, pengembang dapat melihat perangkat android seperti aslinya.

Perbaikan:

andorid → android

Begitupula halnya untuk tim penguji, tim penguji juga dapat memanfaatkan *emulator* untuk melakukan pengujian pada suatu perangkat lunak yang telah selesai dikembangkan.

Perbaikan:

Begitupula → Begitu pula

d) How to Test on IOS Without A Device?

Emulator yang paling populer untuk pengujian perangkat lunak berbasis IOS adalah X code

Perbaikan:

code → *code*

e) How to Test Using Cloud Services

Perbaikan:

How to Test Using Cloud Services → How to Test Using Cloud Services ?

f) What is Cross-Browser testing

Perbaikan:

What is Cross-Browser testing → What is Cross-Browser testing ?

IV. Bab 6

a) Pembahasan Positive Testing

Memasukkan nilai dengan 99999 pada kolom tersebut dapat diterima/*valid* oleh sistem dikarenakan kolom tersebut bertipe **Number**.

Perbaikan:

Number → *Number*

V. Bab 7

a) Pembahasan Modern Web Architecture Explained

Pengujian web arsitektur terbagi atas dua uji, yaitu secara **Front-end** dan **Back-end**. Pengujian secara **Front-end** meliputi uji aplikasi dari bagian *user* yang melakukan interaksi secara langsung.

Perbaikan:

Front-end → *front-end*

Back-end → *back-end*

Pada gambar 1.7 dapat diketahui bahwa *user* adalah pada bagian *client*, dan *Server Side* adalah bagian yang tidak diketahui oleh *user* atau biasa disebut dengan *Back-end* aplikasi. Pada bagian *Server Side*, *user* tidak akan mengetahui proses logika dari aplikasi, *database*, dan hal sensitif lainnya.

Perbaikan:

Server Side → *server side*

Back-end → *back-end*

b) Pembahasan Bug Triage

Dari gambar 7.2 dapat diketahui bahwa *Back-end* adalah sebuah hal yang dilakukan untuk membuat *Front-end* sesuai dengan yang diinginkan. Seperti pada gambar 7.2, *developer* ingin menampilkan singa mengaum, tentunya pada segi *back-end* atau belakang panggung menunjukkan bahwa singa tersebut diikat dan hanya menampilkan kepalanya saja.

Perbaikan:

Back-end → *back-end*

Front-end → *front-end*

c) How to Debug Front-End or Back-End Issues?

Dari gambar 7.10 dapat diketahui bahwa proses validasi terhadap kolom *login* masing kurang bagus, dikarenakan harus melakukan proses secara *front-end* dan *back-end*. Alangkah lebih bagus menggunakan proses validasi seperti menggunakan Ajax atau pengecekan *javascript* terlebih dahulu sebelum melakukan proses *login*.

Perbaikan:

javascript → javascript (tidak ada perbaikan karena javascript tidak termasuk bahasa asing).

VI. Bab 8

a) Pembahasan Sanity Testing

Tujuannya adalah "bukan" untuk memverifikasi secara menyeluruh fungsionalitas baru tetapi untuk menentukan bahwa pengembang telah menerapkan beberapa rasionalitas saat memproduksi perangkat lunak.

Perbaikan:

Tujuannya adalah untuk menentukan bahwa pengembang telah menerapkan beberapa rasionalitas saat memproduksi perangkat lunak.

VII. Bab 9

a) Pembahasan Peran dan Tanggung Jawab Quality Assurance

9. Melakukan penyusunan terkait perencanaan **Prosedur Operasi Standar** (SOP) proses produksi terhadap produk dan layanan.

Perbaikan:

Prosedur Operasi Standar → Standar Operasional Prosedur

b) Pembahasan Education Requirements for Quality Assurances

7. Memiliki pola pikir pengujian, pengujian keamanan, otomasi hingga UAT (**User Acceptance Testing**)

Perbaikan:

User Acceptance Testing → *User Acceptance Testing*

Dalam menjalankan proses atau tugasnya, seorang QA membutuhkan dukungan peralatan atau **tools** yang kompatibel sesuai dengan pekerjaannya.

Perbaikan:

tools → *tools*

Misalnya *Software Quality Assurance* atau *QA Engineer*, maka harus bisa menguasai SDLC (*Software Development Life Cycle*), bahasa pemrograman seperti (**Java, JavaScript, Python, C++** dll), *testing tool, manual testing* hingga *automation testing*.

Perbaikan:

Java, JavaScript, Python, C++ → Java, JavaScript, Python, C++ (tidak ada perbaikan karena tidak termasuk bahasa asing).

c) Pembahasan Quality Assurance Jobs

Banyak *platform* yang dapat digunakan untuk melamar pekerjaan sebagai *quality assurance*, diantaranya **LinkedIn, Glassdoor, Indeed, dan Monster**.

Perbaikan:

LinkedIn, Glassdoor, Indeed, dan Monster → LinkedIn, Glassdoor, Indeed, dan Monster (tidak ada perbaikan karena tidak termasuk bahasa asing).

KELOMPOK WALI

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

I. Bab 3

c) How to Report Bug

- 2) *Summary*, menulis ringkasan singkat tentang apa dan dimana **yang salah**.

Yang salah → permasalahan terjadi

- 3) *Description*, deskripsi ini adalah bidang yang paling penting ketika membuat laporan bug, yang mana deskripsi ini sedikit lebih deskriptif dari **dari** ringkasan (*summary*)

Dari → dihilangkan

II. Bab V

- 1) *Cross-platform testing* mengharuskan aplikasi berjalan **sama baiknya** (**sebaiknya sempurna**)

Sama baiknya (sebaiknya sempurna) → dengan baik

5. How to Use Chrome Mobile View?

- 1) Klik kanan **pada area mana saja di website tersebut** kemudian klik inspeksi (F12)
Pada area mana saja di website tersebut → di sembarang tempat pada website tersebut

III. BAB IV

A. Test Plans

Pembuatan *test plan* dapat dibuat dengan mengikuti *template* pembuatan *test plan*, namun tidak selalu harus mengacu kepada *template test*.

Pembuatan → dihilangkan

4. Features to be tested

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

Daftar, di → dihilangkan

5. Features Not to be Tested

Menjelaskan mengenai fitur **fitur** apa saja yang ada di dalam objek *testing*.

Fitur → dihilangkan

6. Item Pass/Fail Criteria

Kriteria yang dimaksud **merangkup** kriteria yang benar serta kriteria yang salah.

Merangkup → mencakup

B. *What is Deployment?*

- Pada tahapan implementasi, perpindahan dari sistem lama ke sistem baru dibagi menjadi 3 macam yaitu *direct*(secara langsung), *parallel*(setengah-setengah) dan juga *phased*(pergantian ke sistem baru dibuat per modul sistem).

3 macam → 3 jenis

- *Deployment Activities* meliputi **melakukan** tes sistem dan **stres**, **melakukan** tes penerimaan

Melakukan tes sistem → kata melakukan dihilangkan

Tes sistem dan **stres** → tes sistem dan tes stres

Melakukan tes penerimaan → kata melakukan dihilangkan

- *Deployment plan* **program** adalah *trade-off*(perdagangan) di antara sumber daya yang tersedia

Deployment plan **program** → kata program dihilangkan

C. *What is Sanity Testing?*

Tujuannya adalah “bukan” untuk memverifikasi secara menyeluruh fungsionalitas baru tetapi untuk menentukan bahwa pengembangan telah menerapkan beberapa rasionalitas saat memproduksi perangkat lunak. Contoh *sanity testing*:

Dalam proyek *e-commerce*, modul utama adalah halaman *login*, halaman beranda, halaman profil pengguna, pendaftaran pengguna, dll.

Testing: → titik dua(:) dihilangkan