Unpad
Rooted in Culture,
Driven by Impact

#515
QS WORLD RANKING
UNIVERSITY 2026

Evaluasi Perangkat Lunak

Dr. Mira Suryani, S.Pd., M.Kom – PPL I

Outline

- Pentingnya Evaluasi dalam Pengembangan Software
- Mengenal Evaluasi Internal dan Eksternal
- Macam-macam Evaluasi Internal
- Macam-macam Evaluasi Eksternal

#515
QS WORLD RANKING
UNIVERSITY 2026

Pentingnya Evaluasi dalam Pengembangan Software

Apa itu Evaluasi Software?

Evaluasi software adalah proses sistematis untuk menilai kualitas, efektivitas, kinerja, dan kesesuaian perangkat lunak terhadap kebutuhan pengguna maupun spesifikasi yang telah ditetapkan.

Evaluasi ini dilakukan untuk memastikan bahwa software:

- berfungsi dengan benar,
- memenuhi kebutuhan pengguna,
- memiliki kualitas teknis yang baik,
- serta layak digunakan dalam konteks operasional tertentu.

Dalam disiplin **Software Engineering**, evaluasi software mencakup berbagai dimensi, mulai dari aspek teknis hingga pengalaman pengguna.

Ruang Lingkup Evaluasi Software

Evaluasi software dapat mencakup:

Aspek	Contoh
Functional quality	Fitur berjalan sesuai requirement
Usability	Kemudahan penggunaan
Performance	Kecepatan dan efisiensi
Reliability	Stabilitas sistem
Security	Keamanan data dan akses
Maintainability	Kemudahan pengembangan
Compatibility	Dukungan lintas platform
User Experience	Kepuasan pengguna

Evaluasi Internal

Evaluasi internal software adalah proses penilaian kualitas perangkat lunak berdasarkan struktur internal sistem, tanpa perlu menjalankan software dari perspektif pengguna akhir. Fokusnya berada pada bagaimana software dibangun, bukan hanya bagaimana software terlihat atau digunakan.

Evaluasi ini biasanya **dilakukan oleh** developer, software engineer, system architect, atau evaluator teknis.

Dalam model kualitas ISO/IEC 25010, kualitas internal berkaitan dengan atribut yang dapat dianalisis dari:

- source code,
- desain sistem,
- arsitektur,
- database,
- dokumentasi,
- dan struktur implementasi.

Evaluasi Eksternal

Evaluasi eksternal software adalah proses penilaian kualitas perangkat lunak berdasarkan perilaku dan performa sistem saat dijalankan, tanpa melihat langsung kode sumber atau struktur internalnya.

Fokusnya adalah bagaimana software bekerja dari sudut pandang pengguna, penguji, atau lingkungan operasional.

Konsep ini berasal dari model kualitas perangkat lunak seperti ISO/IEC 25010, yang membedakan kualitas menjadi:

- kualitas internal (internal quality),
- kualitas eksternal (external quality),
- dan quality in use.

Perbedaan Evaluasi Internal dan Eksternal

Evaluasi Internal	Evaluasi Eksternal
Fokus pada kode dan struktur internal	Fokus pada perilaku software saat dijalankan
Dilakukan oleh developer	Bisa melibatkan pengguna
Contoh: code quality, complexity	Contoh: usability, performance
Tidak harus menjalankan sistem	Sistem harus dijalankan

#515
QS WORLD RANKING
UNIVERSITY 2026

Macam-macam Testing Internal

1. Code Review

Pemeriksaan kode oleh rekan satu tim sebelum digabung ke repositori utama.

Tujuan: menemukan bug, meningkatkan kualitas kode, menjaga konsistensi, dan berbagi pengetahuan.

Alat bantu: GitHub Pull Request, GitLab Merge Request, Gerrit, Review Board

Praktik terbaik:

- Fokus pada satu hal per review
- Sopan dan konstruktif
- Gunakan checklist code review

Contoh Code Review:

```
# Sebelum:
def calculate_area(l, w):
    return l * w

# Review Komentar:
# "Gunakan nama parameter yang lebih deskriptif."

# Sesudah:
def calculate_area(length, width):
    return length * width
```

Komentar fokus pada clarity, readability, dan maintainability.

Checklist Code Review

Aspek	Pertanyaan
Keterbacaan	Apakah kode mudah dibaca dan dipahami?
Penamaan	Apakah nama variabel, fungsi, dan class deskriptif?
Duplikasi	Apakah ada bagian kode yang duplikat tanpa alasan kuat?
Struktur Logika	Apakah alur logika jelas dan mudah diikuti?
Nilai Konstanta	Apakah ada hard-coded value yang seharusnya dikonstanta?
Gaya Penulisan	Apakah kode mengikuti style guide yang disepakati?
Pengujian	Apakah mencakup unit test atau tes otomatis lain?
Static Analysis	Sudah dicek dengan SonarQube/pylint?
Dokumentasi	Apakah dokumentasi telah diperbarui?
Pengujian Lokal	Apakah kode diuji secara lokal dan bebas dari error?

2. Unit Testing

Pengujian otomatis unit terkecil program (fungsi/metode).

- Tujuan: memastikan unit berfungsi dengan benar secara terisolasi.
- Ciri: cepat, tidak tergantung modul lain, terotomasi.
- Framework: JUnit, pytest, NUnit, Jest.

python

```python
# File: calculator.py
def tambah(a, b):
    return a + b

# File: test_calculator.py
import unittest
from calculator import tambah

class TestCalculator(unittest.TestCase):
    def test_tambah(self):
        self.assertEqual(tambah(2, 3), 5)
        self.assertEqual(tambah(-1, 1), 0)

if __name__ == '__main__':
    unittest.main()
```

Contoh Unit Testing (Sederhana)

2. Unit Testing

Pengujian otomatis unit terkecil program (fungsi/metode).

- Tujuan: memastikan unit berfungsi dengan benar secara terisolasi.
- Ciri: cepat, tidak tergantung modul lain, terotomasi.
- Framework: JUnit, pytest, NUnit, Jest.

python

```python
# File: calculator.py
def tambah(a, b):
    return a + b

# File: test_calculator.py
import unittest
from calculator import tambah

class TestCalculator(unittest.TestCase):
    def test_tambah(self):
        self.assertEqual(tambah(2, 3), 5)
        self.assertEqual(tambah(-1, 1), 0)

if __name__ == '__main__':
    unittest.main()
```

Contoh Unit Testing (Sederhana)

Studi Kasus Unit Testing + CI

Proyek: Sistem Pemesanan Online

- Validasi input pengguna diuji dengan unit test
- Diintegrasikan ke GitHub Actions

```yaml
name: Python Unit Test
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.9'
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run Tests
        run: python -m unittest discover
```

3. Static Code Analysis

Analisis kode tanpa dijalankan

Tujuan: deteksi bug, konsistensi, kompleksitas

Tools: SonarQube, pylint, flake8, ESLint, Checkstyle

Contoh Output:

```
calculator.py:1:0: C0114: Missing module docstring
calculator.py:2:0: C0103: Variable name "l" doesn't conform to snake_case
```

Static Code Analysis – CI Integration

```yaml
name: Static Code Analysis
on: [push, pull_request]
jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.9'
      - name: Install linter
        run: pip install pylint
      - name: Run pylint
        run: pylint calculator.py
```

4. Black-box Testing

Black-box testing adalah metode pengujian perangkat lunak yang berfokus pada **fungsi dan output sistem** tanpa melihat atau mengetahui struktur internal kode program.

Penguji hanya memperhatikan:
- input yang diberikan,
- proses dari sudut pandang pengguna,
- dan output yang dihasilkan.

Karena itu disebut “black box” — bagian internal sistem dianggap seperti kotak hitam yang tidak perlu diketahui.

Tujuan dan Karakteristik Black-box Testing

Black-box testing **bertujuan** untuk memastikan bahwa:

- fitur bekerja sesuai requirement,
- sistem merespons input dengan benar,
- validasi berjalan baik,
- dan tidak ada kesalahan fungsi dari sisi pengguna.

Karakteristik	Penjelasan
Tidak melihat source code	Fokus pada perilaku sistem
Berbasis requirement	Mengacu pada spesifikasi fungsi
Fokus input-output	Memeriksa hasil sistem
Perspektif pengguna	Meniru penggunaan nyata
Dapat dilakukan non-programmer	Tidak wajib memahami coding

Jenis:

- Functional Testing
- Boundary Value Testing
- Equivalence Partitioning
- Decision Table Testing

BT 1: Functional Testing

Skenario Aplikasi:
Formulir login pada sistem pembelajaran daring.

Spesifikasi Fungsi:
- Jika username dan password benar → masuk ke dashboard.
- Jika salah satu atau keduanya salah → tampilkan pesan “Login gagal”.

BT 1: Functional Testing

Kasus Uji 1: Input Valid

- **Input:**
  - Username: `mahasiswa01`
  - Password: `Belajar123`
- **Ekspektasi:** Pengguna diarahkan ke halaman dashboard.
- **Status:** ✔ Lulus

Kasus Uji 3: Username Kosong

- **Input:**
  - Username: *(kosong)*
  - Password: `Belajar123`
- **Ekspektasi:** Muncul pesan “Harap masukkan username”.
- **Status:** ✔ Lulus

Kasus Uji 2: Password Salah

- **Input:**
  - Username: `mahasiswa01`
  - Password: `salahpass`
- **Ekspektasi:** Muncul pesan “Login gagal”.
- **Status:** ✔ Lulus

Kasus Uji 4: Form Kosong

- **Input:**
  - Username: *(kosong)*
  - Password: *(kosong)*
- **Ekspektasi:** Muncul pesan “Harap isi semua field”.
- **Status:** ✔ Lulus

BT 2: Decision Table Testing

Decision Table Testing adalah teknik pengujian software yang digunakan untuk menguji berbagai kombinasi kondisi dan keputusan dalam sistem.

Skenario:
Sistem toko online memiliki fitur pemberian diskon berdasarkan dua kondisi:

1. Status pelanggan sebagai member atau bukan member.
2. Total belanja pelanggan apakah lebih dari Rp500.000 atau tidak.

Pengujian dilakukan untuk memastikan bahwa sistem memberikan persentase diskon yang sesuai untuk setiap kombinasi kondisi tersebut.

Reference Decision Table

Member	>500rb	Diskon
Ya	Ya	20%
Ya	Tidak	10%
Tidak	Ya	5%
Tidak	Tidak	0%

BT 2: Decision Table Testing

Skenario Testing (*Test Case*)

Test Case ID	Status Member	Total Belanja	Kondisi yang Diharapkan	Expected Result
TC-01	Ya	> 500rb	Member dan belanja di atas 500rb	Sistem memberikan diskon 20%
TC-02	Ya	≤ 500rb	Member tetapi belanja tidak lebih dari 500rb	Sistem memberikan diskon 10%
TC-03	Tidak	> 500rb	Bukan member tetapi belanja di atas 500rb	Sistem memberikan diskon 5%
TC-04	Tidak	≤ 500rb	Bukan member dan belanja tidak lebih dari 500rb	Sistem tidak memberikan diskon (0%)

BT 3: Boundary Value Testing

Boundary Value Testing (Boundary Value Analysis/BVA) adalah teknik pengujian software yang berfokus pada pengujian nilai batas minimum dan maksimum dari suatu input.

Kenapa Boundary Value Testing Penting?
Banyak bug terjadi pada:

- batas minimum,
- batas maksimum,
- transisi valid ↔ invalid.

Contoh kesalahan umum:

- program memakai > padahal harusnya >=,
- input batas tidak terbaca benar,
- validasi off-by-one error.

BT 3: Boundary Value Testing

Contoh pada sistem diskon (seperti BT2)

Aturan:
Diskon 20% jika belanja > Rp500.000

Maka boundary test yang penting:

Total Belanja	Expected Result
Rp499.999	Tidak masuk diskon 20%
Rp500.000	Tidak masuk diskon 20%
Rp500.001	Masuk diskon 20%

Tujuannya memastikan operator “lebih dari” (>) berjalan benar.

BT 3: Boundary Value Testing

Karakteristik Boundary Value Testing

Karakteristik	Penjelasan
Fokus pada nilai batas	Minimum dan maksimum
Menguji area rawan error	Sekitar batas validasi
Bagian dari black-box testing	Tidak melihat kode
Cocok untuk input numerik	Umur, nilai, jumlah, panjang karakter

Nilai yang Biasanya Diuji

Jika rentang valid adalah:

$$1 \leq x \leq 100$$

Maka biasanya diuji:

- 0 → di bawah batas,
- 1 → batas minimum,
- 2 → tepat di atas minimum,
- 99 → tepat di bawah maksimum,
- 100 → batas maksimum,
- 101 → di atas maksimum.

Perbedaan
Boundary Value & Decision Table Testing

Boundary Value Testing	Decision Table Testing
Fokus nilai batas	Fokus kombinasi kondisi
Menguji input numerik/range	Menguji aturan logika
Contoh: nilai 0, 1, 100	Contoh: member/non-member
Cari error di batas	Cari error aturan keputusan

BT 4: Equivalence Partitioning

Equivalence Partitioning adalah teknik pengujian software yang membagi data input ke dalam beberapa kelompok (partition/class) yang dianggap memiliki perilaku sama.

Tujuannya adalah:
- mengurangi jumlah test case,
- tetapi tetap mewakili seluruh kemungkinan input.

Konsep Dasar
Daripada menguji semua kemungkinan input, kita cukup mengambil satu atau beberapa perwakilan dari setiap kelompok input.

Input dibagi menjadi:
- kelas valid,
- dan kelas tidak valid.

Contoh Equivalence Partitioning

Kasus: Sistem input nilai ujian dengan ketentuan:

- Nilai valid: **0–100**
- Nilai tidak valid: < 0 atau > 100

Tujuan: Menguji representasi tiap kelompok nilai (partisi).

Partisi Data	Contoh Nilai	Status yang Diharapkan
Valid (0–100)	75	Diterima
Invalid (kurang dari 0)	-5	Ditolak
Invalid (lebih dari 100)	105	Ditolak

#515
QS WORLD RANKING
UNIVERSITY 2026

Macam-macam Testing Eksternal

Jenis Eksternal Testing

Jenis Testing	Fokus
Functional	Fitur bekerja benar
Usability	Mudah digunakan
UX	Pengalaman pengguna
Performance	Cepat dan efisien
Compatibility	Jalan di banyak platform
Security	Aman
Reliability	Stabil
Acceptance	Sesuai kebutuhan pengguna
Accessibility	Bisa digunakan semua orang
Regression	Update tidak merusak fitur lama

1. Usability Testing dengan SUS

- System Usability Scale adalah metode evaluasi usability yang digunakan untuk mengukur tingkat kemudahan penggunaan suatu sistem atau aplikasi berdasarkan persepsi pengguna.

- SUS dikembangkan oleh John Brooke pada tahun 1986 dan menjadi salah satu instrumen usability testing yang paling populer karena sederhana, cepat digunakan, dan memiliki reliabilitas tinggi.

SUS terdiri dari 10 pernyataan dengan skala Likert 1–5, mulai dari “sangat tidak setuju” hingga “sangat setuju”. Pernyataan tersebut mencakup aspek seperti kemudahan penggunaan, kompleksitas sistem, konsistensi, dan kepercayaan diri pengguna saat menggunakan sistem.

Hasil SUS menghasilkan skor antara 0–100, di mana semakin tinggi skor menunjukkan tingkat usability yang semakin baik. Secara umum:
- skor di atas 68 dianggap baik,
- skor di bawah 68 menunjukkan usability masih perlu ditingkatkan.

1. Usability Testing dengan SUS

- 10 pertanyaan, skala 1–5
- Untuk item ganjil: skor = nilai - 1
- Untuk item genap: skor = 5 - nilai
- Total skor × 2.5 → nilai 0–100

No	Pernyataan	Skala Likert (1-5)
1	Saya berpikir akan menggunakan sistem ini lagi.	
2	Saya merasa sistem ini rumit untuk digunakan.	
3	Saya merasa sistem ini mudah untuk digunakan.	
4	Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini.	
5	Saya merasa fitur-fitur sistem ini berjalan dengan semestinya.	
6	Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) pada sistem ini.	
7	Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat.	
8	Saya merasa sistem ini membingungkan.	
9	Saya merasa tidak ada hambatan dalam menggunakan sistem ini.	
10	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini.	

Keterangan Skala Likert:

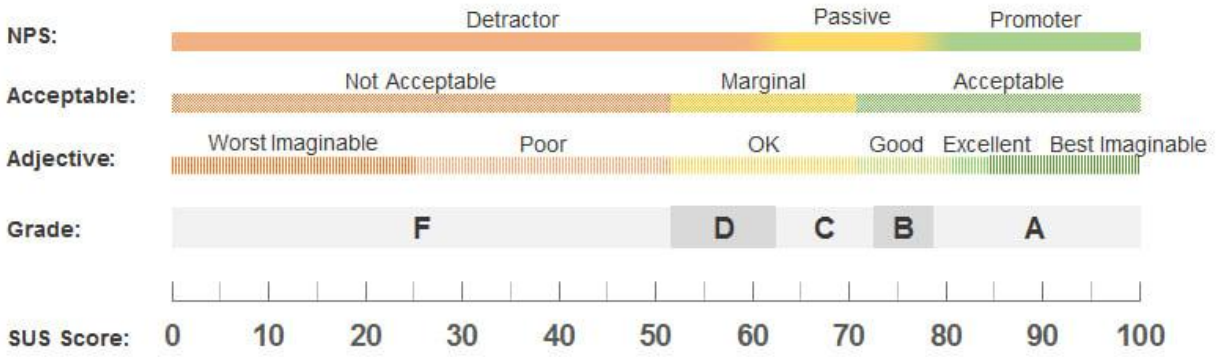
1 = Sangat tidak setuju | 2 = Tidak setuju | 3 = Cukup setuju | 4 = Setuju | 5 = Sangat setuju

Cek Excel Perhitungan

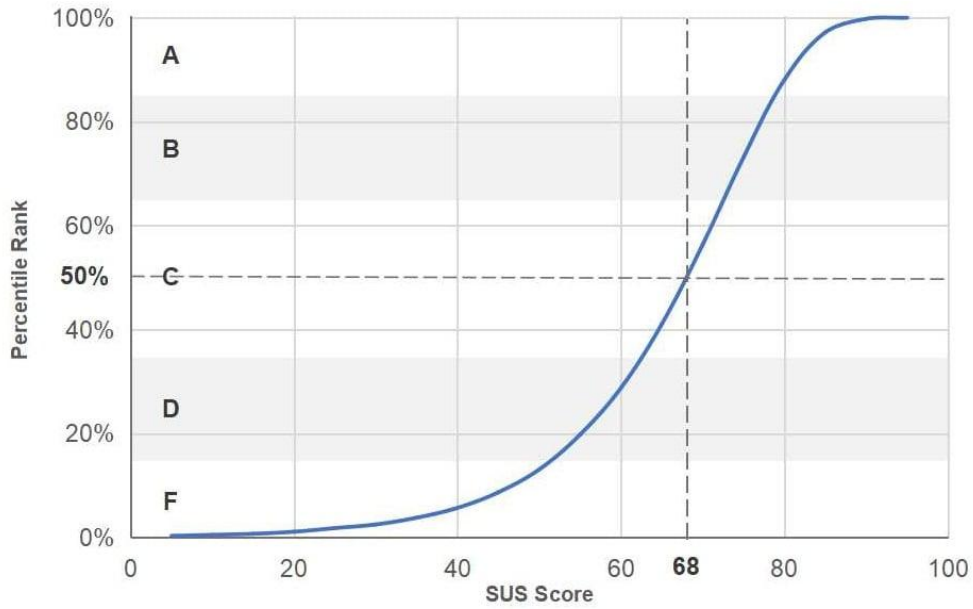
1. Usability Testing dengan SUS

Contoh Jawaban:
1=4, 2=2, 3=4, 4=2, 5=5, 6=2, 7=4, 8=1, 9=5, 10=2
→ SUS = (17+16)×2.5 = **82.5** → Good to Excellent

Interpretasi SUS Score



Interpretasi SUS Score using Percentile



2. UX Testing dengan UEQ

- Instrumen untuk mengevaluasi kualitas pengalaman pengguna secara komprehensif.
- Menggunakan pasangan kata berlawanan dalam skala bipolar (misalnya: menyenangkan – tidak menyenangkan).
- Setiap pernyataan diukur dalam skala 7 poin.
- Terdiri dari 6 skala utama:
  - **Attractiveness**: keseluruhan kesan pengguna
  - **Perspicuity**: kemudahan untuk memahami dan belajar
  - **Efficiency**: kecepatan dan efektivitas
  - **Dependability**: kepercayaan terhadap sistem
  - **Stimulation**: daya tarik dan motivasi penggunaan
  - **Novelty**: kebaruan dan kreativitas antarmuka

For More Details, Go to Website - https://www.ueq-online.org/

3. UX Testing dengan UEQ+

- UEQ+ adalah ekstensi modular dari Kuesioner Pengalaman Pengguna (UEQ). UEQ+ terdiri dari daftar skala UX yang dapat dikombinasikan untuk membangun kuesioner yang dioptimalkan untuk pertanyaan penelitian tertentu.
- Dengan demikian, UEQ+ adalah kerangka kerja untuk membangun kuesioner yang dioptimalkan untuk skenario evaluasi khusus.
- Ada 7 Skala dengan konsep **plug and play**

For More Details, Go to Website - https://ueqplus.ueq-research.org/

3. UX Testing dengan UEQ+

- UEQ+ adalah ekstensi modular dari Kuesioner Pengalaman Pengguna (UEQ). UEQ+ terdiri dari daftar skala UX yang dapat dikombinasikan untuk membangun kuesioner yang dioptimalkan untuk pertanyaan penelitian tertentu.
- Dengan demikian, UEQ+ adalah kerangka kerja untuk membangun kuesioner yang dioptimalkan untuk skenario evaluasi khusus.
- Ada 7 kategori dengan konsep **plug and play**

For More Details, Go to Website - https://ueqplus.ueq-research.org/

References

Brooke, J. (1996). *SUS: A quick and dirty usability scale*. In P. W. Jordan, B. Thomas, I. L. McClelland, & B. Weerdmeester (Eds.), *Usability evaluation in industry* (pp. 189–194). Taylor & Francis.
PDF SUS Original

International Organization for Standardization. (2011). *ISO/IEC 25010:2011 systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. ISO.
ISO/IEC 25010 Official Page

International Organization for Standardization. (2011). *ISO/IEC 25010:2011*.
ISO/IEC 25010 PDF Preview

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw Hill.

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing* (3rd ed.). John Wiley & Sons.

Beizer, B. (1995). *Black-box testing: Techniques for functional testing of software and systems*. John Wiley & Sons.

Patton, R. (2005). *Software testing* (2nd ed.). SAMS Publishing.

Sauro, J. (2011). *Measuring usability with the System Usability Scale (SUS)*. MeasuringU.
MeasuringU SUS Guide

Santoso, H. B., Schrepp, M., Isal, R. Y. K., Utomo, A. Y., & Priyogi, B. (2016). Measuring user experience of the student-centered e-learning environment. *Journal of Educators Online, 13*(1), 58–79.

Schrepp, M., Hinderks, A., & Thomaschewski, J. (2017). Design and evaluation of a short version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence, 4*(6), 103–108.

Schrepp, M., Hinderks, A., & Thomaschewski, J. (2019). Construction of a benchmark for the User Experience Questionnaire (UEQ). *International Journal of Interactive Multimedia and Artificial Intelligence, 4*(4), 40–44.

Enda, D. (2025). Pengukuran kualitas perangkat lunak dengan standar ISO/IEC 25010. *Jurnal Ekonomi dan Komputer (JEKIN)*.
Artikel ISO 25010 Indonesia

Susanti, E. (2023). Penilaian kualitas sistem informasi menggunakan ISO/IEC 25010. *JUTISI*.
Jurnal Penilaian ISO 25010

Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.

Unpad

Rooted in Culture,
Driven by Impact

#515
QS WORLD RANKING
UNIVERSITY 2026

Thank you

Terima Kasih

Universitas Padjadjaran

Jl. Ir. Soekarno Km. 21 Jatinangor, Sumedang 45363

Jl. Dipati Ukur No. 35, Bandung 40132

www.unpad.ac.id

Unpad

Rooted in Culture,
Driven by Impact

#515
QS WORLD RANKING
UNIVERSITY 2026