Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 5 No. 2 (2026) pp: 22451-22458
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Automation Testing pada Website Konseling Mahasiswa (Sikosa) Menggunakan *Jest*

Muhammad Aidil[1], Muhammad Jibril[2], Fitri Yunita[3]
[1,2,3]Sistem Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Islam Indragiri
[1]muhammad.aidil09767@gmail.com, [2]jibril.unisi@gmail.com, [3]fitriyun@gmail.com

Abstrak

Sistem Konseling Mahasiswa (SiKosa) merupakan platform berbasis web yang dirancang untuk mendukung layanan konseling daring bagi mahasiswa melalui fitur autentikasi, pengelolaan profil, konsultasi, chat, chatbot, serta pengelolaan pengguna dan artikel. Kompleksitas fitur dan pembaruan sistem secara berkala menuntut proses pengujian yang konsisten, efisien, dan terdokumentasi. Penelitian ini bertujuan menerapkan automation testing pada website SiKosa menggunakan framework Jest dengan pendekatan Software Testing Life Cycle (STLC) untuk memastikan setiap fungsi berjalan sesuai kebutuhan sistem. Tahapan penelitian meliputi requirement analysis, test planning, test case development, test environment setup, test execution, dan test closure. Pengujian dilakukan terhadap tujuh kelompok fitur yang melibatkan peran mahasiswa, psikolog, dan administrator. Dua pendekatan digunakan, yaitu unit testing untuk menguji service layer secara terpisah menggunakan mock serta integration testing untuk menguji endpoint Application Programming Interface melalui HTTP request dengan database aktual dan Supertest. Test case disusun berdasarkan kebutuhan fungsional, mencakup jalur positif dan negatif, serta dieksekusi secara berurutan melalui Command Line Interface Jest. Hasil pengujian menunjukkan bahwa seluruh 26 test case dalam 26 test suite berhasil dijalankan dengan status lulus dan menghasilkan test pass rate sebesar 100%. Pengujian mengonfirmasi bahwa validasi masukan, pengelolaan sesi, otorisasi berbasis peran, respons layanan, dan konsistensi data berfungsi sesuai spesifikasi. Penerapan Jest dengan tahapan STLC terbukti mendukung pengujian SiKosa yang lebih cepat, terstruktur, konsisten, dapat diulang, dan terdokumentasi.

Kata kunci: *Pengujian Otomatis, Jest, Sikosa, Software Testing Life Cycle, Konseling Mahasiswa*

1. Latar Belakang

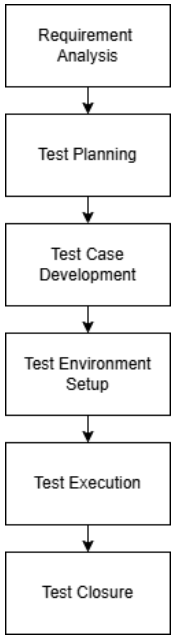
Sistem Konseling Mahasiswa (SiKosa) merupakan platform berbasis web yang dirancang untuk memfasilitasi layanan konseling mahasiswa secara daring. Melalui SiKosa, mahasiswa dapat melakukan konsultasi dengan psikolog secara online tanpa terbatas ruang dan waktu. Dalam implementasinya, sistem ini mencakup fitur autentikasi, manajemen profil, konsultasi, chat, chatbot, serta manajemen data pengguna dan artikel. Meningkatnya kompleksitas fitur berpotensi menimbulkan bug atau malfungsi[1]. Untuk itu diperlukan proses pengujian yang sistematis untuk menjamin kualitas dan stabilitas sistem[2].

Pengujian merupakan salah satu tahap penting dalam *System Development Life Cycle* (SDLC) sebagai metode sistematis dalam proses pengujian perangkat lunak[3]. Secara umum, metode pengujian terbagi menjadi manual dan otomatis[4] . Pengujian manual tergolong mudah diterapkan namun memiliki risiko kesalahan tinggi dan kurang efisien, terutama pada aplikasi web seperti SiKosa yang mengalami pembaruan secara berkala[5]. Sebaliknya, pengujian otomatis memanfaatkan alat bantu untuk mengeksekusi skenario uji secara sistematis, sehingga lebih cepat, konsisten, dan memiliki cakupan yang lebih luas[6]. Dalam penerapan pengujian otomatis, diperlukan testing framework yang ringan, mudah dikonfigurasi, serta sesuai dengan bahasa pemrograman aplikasi. Jest dipilih karena mudah diintegrasikan dengan proyek berbasis JavaScript dan memiliki dukungan komunitas yang luas[7]. Jest mendukung unit test, integration test, dan snapshot test, serta dilengkapi fitur automatic mocking, coverage reporting, dan parallel execution yang memungkinkan pengujian berjalan secara efisien[8]. Karena SiKosa dibangun dengan teknologi JavaScript, Jest menjadi pilihan ideal untuk pengujian website ini.

Berdasarkan celah tersebut, penelitian ini bertujuan untuk menerapkan pengujian otomatis pada website SiKosa menggunakan framework Jest dengan pendekatan Software Testing Life Cycle (STLC). Pengujian mencakup unit testing dan integration testing pada seluruh modul sistem. Hasil pengujian dievaluasi melalui metrik test pass rate untuk memastikan sistem berjalan sesuai kebutuhan secara efisien dan konsisten.

2. Metode Penelitian

Penelitian ini menggunakan pendekatan kuantitatif deskriptif dengan metode *Software Testing Life Cycle* (STLC). STLC dipilih karena berfokus pada aktivitas pengujian secara sistematis, berbeda dengan SDLC yang mencakup seluruh siklus pengembangan[9]. Tahapan STLC yang diterapkan meliputi *requirement analysis*, *test planning*, *test case development*, *test environment setup*, *test execution*, dan *test closure*. Tahap *Requirement Analysis* dilakukan dengan menganalisis kebutuhan sistem SiKosa untuk memahami fitur, alur kerja, dan spesifikasi yang akan diuji[10]. Selanjutnya, pada tahap *Test Planning* disusun strategi pengujian dengan menetapkan *Jest* sebagai *testing framework* serta menentukan ruang lingkup[11].Tahap *Test Case Development* berfokus pada penyusunan *test case* sebagai acuan pelaksanaan pengujian [12]. Setelah itu, tahap *Test Environment Setup* dilakukan dengan menyiapkan lingkungan pengujian melalui instalasi *tools*, konfigurasi berkas pengujian, penyusunan struktur direktori, dan penyediaan data uji[13]. Pada tahap *Test Execution*, seluruh *test case* dieksekusi menggunakan *Jest* melalui *Command Line Interface* (CLI), kemudian hasil pengujian dicatat secara otomatis dalam bentuk status *pass/fail* beserta *error log* apabila ditemukan kegagalan[14]. Tahap terakhir, yaitu *Test Closure*, dilakukan dengan mengevaluasi hasil pengujian dan menyusun laporan untuk mendokumentasikan hasil uji serta temuan bug.



Gambar 1. Alur tahapan Software Testing Life Cycle (STLC)

Objek penelitian adalah *website* Sistem Konseling Mahasiswa (SiKosa). Pengujian mencakup tujuh kelompok fitur: autentikasi, manajemen profil mahasiswa, manajemen profil psikolog, konsultasi, *chat*, *chatbot*, serta manajemen data *user* dan artikel. Tiga peran pengguna diuji: mahasiswa, psikolog, dan *admin*. Dua pendekatan pengujian diterapkan. Unit testing menguji service layer secara terpisah dan *integration testing* menguji *API endpoint* dengan database asli melalui *HTTP request* menggunakan Supertest[15]. *Framework* utama adalah Jest, dengan jest-html-reporters dan jest-stare untuk dokumentasi hasil.

Test case dikembangkan berdasarkan *main map* setiap fitur, diturunkan menjadi *test scenario*, lalu dijabarkan ke *test case* terperinci yang mencakup *Scenario ID*, *Test Case ID*, *Scenario Description Test Steps*, *Expected Result[16]*. Data uji menggunakan data simulasi (*dummy*) yang sudah disiapkan.

Evaluasi dilakukan melalui metrik *test pass rate*, *test pass rate* dihitung sebagai persentase *test case* lulus terhadap total eksekusi.

$$\text{Test Pass Rate} = \frac{\text{Jumlah Test Case Lulus}}{\text{Total Test Case}} \times 100\% \qquad \textbf{(1)}$$

Rumus (1) digunakan untuk menghitung *Test Pass Rate*, yaitu persentase *test case* yang berhasil dijalankan sesuai dengan hasil yang diharapkan terhadap seluruh *test case* yang diuji. Nilai *Test Pass Rate* diperoleh dengan membagi jumlah *test case* yang berstatus lulus (*pass*) dengan total *test case* yang dieksekusi, kemudian hasilnya dikalikan 100% sehingga diperoleh nilai dalam bentuk persentase. Semakin tinggi nilai *Test Pass Rate*, semakin banyak skenario pengujian yang berhasil dipenuhi oleh sistem, yang menunjukkan bahwa fungsionalitas sistem telah berjalan sesuai dengan spesifikasi yang ditetapkan.

DOI https://doi.org/10.31004/riggs.v5i2.11863

3. Hasil dan Diskusi

Pelaksanaan pengujian otomatis pada SiKosa menggunakan Jest mengikuti enam tahapan STLC. Berikut diuraikan hasil dan pembahasan setiap tahapan tersebut secara terintegrasi untuk seluruh peran pengguna.

3.1 *Requirement Analysis*

Tahap analisis kebutuhan mengidentifikasi tujuh kelompok fitur yang menjadi fokus pengujian, yaitu autentikasi, manajemen profil mahasiswa, manajemen profil psikolog, konsultasi, chat, chatbot, serta manajemen data user dan artikel. Dua pendekatan pengujian ditetapkan pada tahap ini, yaitu unit testing untuk menguji service layer dengan mock dan integration testing untuk menguji API endpoint melalui HTTP request.

Tabel 1. Distribusi kelompok fitur dan pendekatan pengujian

Kelompok Fitur	Peran Terkait	Unit Test	Integration Test
Autentikasi (registrasi, login, logout)	Mahasiswa, Psikolog, Admin	Ya	Ya
Manajemen profil mahasiswa	Mahasiswa	Ya	Ya
Manajemen profil psikolog	Psikolog	Ya	Ya
Konsultasi	Mahasiswa, Psikolog	Ya	Ya
Chat	Mahasiswa, Psikolog	Ya	Ya
Chatbot	Mahasiswa	Ya	Ya
Manajemen data user dan artikel	Admin, Psikolog	Ya	Ya

3.2 *Test Planning*

Strategi pengujian ditetapkan dengan Jest sebagai framework utama, Supertest untuk pengujian endpoint API, dan Mongoose sebagai penghubung database. Dokumentasi hasil menggunakan jest-html-reporters dan jest-stare. Pelacakan kebutuhan dilakukan melalui Requirement Traceability Matrix (RTM) untuk memastikan seluruh kebutuhan fungsional terverifikasi oleh setidaknya satu test case.

3.3 *Test Case Development*

Test case dikembangkan secara bertahap dimulai dari *main map* setiap fitur, diturunkan menjadi test scenario, lalu dijabarkan ke test case terperinci yang mencakup *Scenario id*, *test case id scenario description test steps expected result*. Seluruh kebutuhan fungsional pada modul mahasiswa, psikolog, dan admin telah dipetakan ke test case yang sesuai.

Tabel 2. Daftar test scenario pengujian SiKosa

Scenario ID	Peran	Skenario Pengujian
TS-MHS-01	Mahasiswa	Autentikasi: Proses pendaftaran akun baru, masuk ke sistem, dan keluar dari sistem
TS-MHS-02	Mahasiswa	Kelola Profil: Mengubah data diri seperti nama, NIM, dan foto profil
TS-MHS-03	Mahasiswa	Konsultasi: Mengajukan konsultasi ke psikolog, riwayat dan detail konsultasi
TS-MHS-04	Mahasiswa	Chat: Berkomunikasi melalui pesan teks dengan psikolog
TS-MHS-05	Mahasiswa	Chatbot AI: Berinteraksi dengan asisten virtual berbasis AI untuk mendapatkan informasi atau bantuan awal
TS-PSI-01	Psikolog	Autentikasi: Proses masuk ke sistem dan keluar dari sistem
TS-PSI-02	Psikolog	Kelola Profil: Mengubah data diri seperti nama, deskripsi, spesialisasi, latar belakang pendidikan, dan foto profil
TS-PSI-03	Psikolog	Konsultasi: Melihat notifikasi konsultasi masuk serta menerima atau menolak permintaan konsultasi dari mahasiswa
TS-PSI-04	Psikolog	Chat: Berkomunikasi melalui pesan teks dengan mahasiswa yang konsultasinya telah diterima
TS-PSI-05	Psikolog	Kelola Artikel: Membuat, mengubah, melihat, dan menghapus artikel yang ditulis
TS-ADM-01	Admin	Autentikasi: Proses masuk ke sistem dan keluar dari sistem
TS-ADM-02	Admin	Kelola User: Melihat, menambah, mengubah, dan menghapus data pengguna (mahasiswa, psikolog, admin)
TS-ADM-03	Admin	Kelola Artikel: Melihat, menambah, mengubah, dan menghapus artikel

Untuk masing-masing scenario ditampilkan dua *test case*, yaitu satu jalur positif (*happy path*) dan satu jalur negatif (*negative path*), yang diambil dari *unit testing* maupun *integration testing*.

DOI https://doi.org/10.31004/riggs.v5i2.11863

Tabel 3. Contoh test case scenario TS-MHS-01 – Autentikasi Mahasiswa

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-MHS-01	TC-LOGIN-10	Login sebagai mahasiswa berhasil. Kondisi awal: Akun mahasiswa dengan email "mahasiswa@mail.com" dan kata sandi "Valid123!" sudah terdaftar.	1. Siapkan data login dengan mengisi kolom email dengan "mahasiswa@mail.com" dan mengisi kolom kata sandi dengan "Valid123!". 2. Kirim data login ke sistem.	Permintaan berhasil (200). Sistem mengembalikan data user dengan peran mahasiswa, beserta kode akses dan kode penyegar.
TS-MHS-01	TC-INT-LOGIN-003	Body login kosong. Kondisi awal: Tidak ada data khusus.	1. Kosongkan seluruh body login (tanpa mengisi kolom email dan kata sandi). 2. Kirim data login ke sistem.	Permintaan ditolak (400). Sistem mengembalikan daftar kesalahan.

Tabel 4. Contoh test case scenario TS-MHS-02 – Kelola Profil Mahasiswa

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-MHS-02	TC-INT-UP-017	Ubah profil - tanpa perubahan tetap berhasil.	1. Login menggunakan akun mahasiswa (email "mahasiswa@mail.com", kata sandi "Valid123!").	Permintaan berhasil (200).
TS-MHS-02	TC-MHS-UP-06	Ubah tanpa kode akses. Kondisi awal: User mahasiswa terdaftar.	1. Kirim data perubahan (nama "New Name") tanpa token akses.	Permintaan ditolak (401).

Tabel 5. Contoh test case scenario TS-MHS-03 – Konsultasi Mahasiswa

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-MHS-03	CONS-MHS-09	Ajukan konsultasi berhasil. Kondisi awal: Belum ada konsultasi, ID psikolog valid.	1. Masuk sebagai mahasiswa (email "mahasiswa@mail.com", kata sandi "Valid123!"). 2. Kirim psychologistId valid dan message "Saya ingin konsultasi". 3. Periksa basis data.	Permintaan berhasil (201). Konsultasi baru status "menunggu".
TS-MHS-03	TC-INT-CONS-MHS-012	Ajukan - sesi kadaluwarsa. Kondisi awal: Sesi	1. Tunggu hingga token akses kadaluwarsa. 2. Kirim konsultasi dengan psychologistId valid dan	Permintaan ditolak (401).

Tabel 6. Contoh test case scenario TS-MHS-04 – Chat Mahasiswa

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-MHS-04	TC-INT-CHAT-MHS-006	Kirim pesan dengan emoji. Kondisi awal: Ruang aktif.	1. Login menggunakan akun mahasiswa (email "mahasiswa@mail.com", kata sandi "Valid123!"). 2. Kirim pesan dengan	Permintaan berhasil (201).
TS-MHS-04	TC-MHS-01	Daftar ruang obrolan tanpa kode akses. Kondisi awal: Belum ada sesi.	1. Kirim permintaan daftar ruang obrolan tanpa token akses.	Permintaan ditolak (401).

Tabel 7. Contoh test case scenario TS-MHS-05 – Chatbot Mahasiswa

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-MHS-05	TC-CHATBOT-SVC-02	Kirim chat dengan 1 pesan user. Kondisi awal: Layanan Groq diatur respons "Test reply".	1. Siapkan data chat dengan messages [{ "role": "user", "content": "Halo" }]. 2. Kirim chat ke sistem.	Permintaan berhasil (200). Respons peran "assistant", isi "Test reply".
TS-MHS-05	TC-CHATBOT-INT-02	Layanan Groq tidak tersedia. Kondisi awal: Layanan Groq mati.	1. Kirim chat ke sistem dengan messages [{ "role": "user", "content": "Halo" }].	Permintaan gagal (503).

Tabel 8. Contoh test case scenario TS-PSI-01 – Autentikasi Psikolog

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-PSI-01	TC-INT-LOGIN-001	Login - session tercatat. Kondisi awal: Akun terdaftar, belum ada sesi.	1. Login dengan email "psikolog@mail.com" dan kata sandi "Valid123!". 2. Periksa data di basis data sesi.	Sesi baru tercatat.
TS-PSI-01	TC-LOGIN-01	Login dengan email kosong. Kondisi awal: Belum ada sesi.	1. Siapkan data login dengan mengosongkan kolom email dan mengisi kolom kata sandi dengan "Valid123!". 2. Kirim data login ke sistem.	Permintaan ditolak (400). Email wajib diisi.

DOI https://doi.org/10.31004/riggs.v5i2.11863

Tabel 9. Contoh test case scenario TS-PSI-02 – Kelola Profil Psikolog

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-PSI-02	TC-PSI-UP-13	Ubah nama berhasil. Kondisi awal: Psikolog dengan nama "Old Name", kode valid.	1. Login dengan email "psikolog@mail.com" dan kata sandi "Valid123!". 2. Kirim nama "New Name". 3. Periksa data di basis data.	Permintaan berhasil (200). Nama berubah.
TS-PSI-02	TC-INT-PSI-UP-006	Ubah - file bukan gambar ditolak. Kondisi awal: Psikolog login.	1. Login dengan email "psikolog@mail.com" dan kata sandi "Valid123!". 2. Unggah file "test.txt" (bukan gambar).	Permintaan ditolak (400).

Tabel 10. Contoh test case scenario TS-PSI-03 – Konsultasi Psikolog

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-PSI-03	TC-INT-CONS-PSI-014	Accept konsultasi berhasil. Kondisi awal: Konsultasi pending, room inactive.	1. Login dengan email "psikolog@mail.com" dan kata sandi "Valid123!". 2. Kirim update status konsultasi dengan status "accepted". 3. Periksa basis data.	Permintaan berhasil (200). Status jadi "accepted", room jadi "active".
TS-PSI-03	CONS-PSI-01	Lihat notifikasi tanpa kode akses. Kondisi awal: Belum ada sesi.	1. Kirim permintaan notifikasi tanpa token akses (Authorization header).	Permintaan ditolak (401).

Tabel 11. Contoh test case scenario TS-PSI-04 – Chat Psikolog

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-PSI-04	TC-PSI-02	Daftar ruang berhasil. Kondisi awal: Room milik psikolog ada.	1. Login dengan email "psikolog@mail.com" dan kata sandi "Valid123!". 2. Kirim permintaan daftar chat room.	Permintaan berhasil (200). Daftar ruang.
TS-PSI-04	TC-INT-CHAT-PSI-001	Daftar ruang tanpa kode akses. Kondisi awal: Belum ada sesi.	1. Kirim permintaan daftar ruang chat tanpa token akses.	Permintaan ditolak (401).

Tabel 12. Contoh test case scenario TS-PSI-05 – Kelola Artikel Psikolog

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-PSI-05	TC-INT-PSI-14	Update artikel milik sendiri berhasil. Kondisi awal: Artikel milik psikolog ini.	1. Login dengan email "psikolog@mail.com" dan kata sandi "Valid123!". 2. Kirim permintaan update artikel dengan title baru.	Permintaan berhasil (200).
TS-PSI-05	TC-PSI-ART-01	Buat artikel tanpa kode akses. Kondisi awal: Belum ada sesi.	1. Kirim data artikel baru (judul "Test", konten "Test content") tanpa token akses (Authorization header).	Permintaan ditolak (401).

Tabel 13. Contoh test case scenario TS-ADM-01 – Autentikasi Admin

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-ADM-01	TC-LOGIN-08	Login admin berhasil. Kondisi awal: Akun admin dengan email "admin@mail.com", sandi "Valid123!" terdaftar.	1. Siapkan data login dengan mengisi kolom email dengan "admin@mail.com" dan mengisi kolom kata sandi dengan "Valid123!". 2. Kirim data login ke sistem. 3. Periksa data.	Permintaan berhasil (200). Data dengan peran admin.
TS-ADM-01	TC-INT-LOGIN-003	Body kosong. Kondisi awal: Tidak ada data.	1. Kosongkan seluruh body (tanpa email dan kata sandi). 2. Kirim data ke sistem.	Permintaan ditolak (400).

Tabel 14. Contoh test case scenario TS-ADM-02 – Kelola Data User (Admin)

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-ADM-02	TC-INT-004	Lihat semua user berhasil (admin). Kondisi awal: Admin login.	1. Login dengan email "admin@mail.com" dan kata sandi "Valid123!". 2. Kirim permintaan daftar seluruh user.	Permintaan berhasil (200). data adalah array.
TS-ADM-02	TC-UM-001	Lihat semua user tanpa kode akses. Kondisi awal: Admin di basis data.	1. Kirim permintaan daftar user tanpa token akses (Authorization header).	Permintaan ditolak (401).

DOI https://doi.org/10.31004/riggs.v5i2.11863

Tabel 15. Contoh test case scenario TS-ADM-03 – Kelola Artikel (Admin)

Scenario ID	Test Case ID	Scenario Description	Test Steps	Expected Result
TS-ADM-03	TC-ADM-ART-07	Lihat artikel by ID valid. Kondisi awal: Artikel ada.	1. Login dengan email "admin@mail.com" dan kata sandi "Valid123!". 2. Kirim permintaan dengan articleId valid.	Permintaan berhasil (200).
TS-ADM-03	TC-INT-ADM-05	Buat artikel tanpa kode akses. Kondisi awal: Belum ada sesi.	1. Kirim permintaan buat artikel tanpa token akses.	Permintaan ditolak (401).

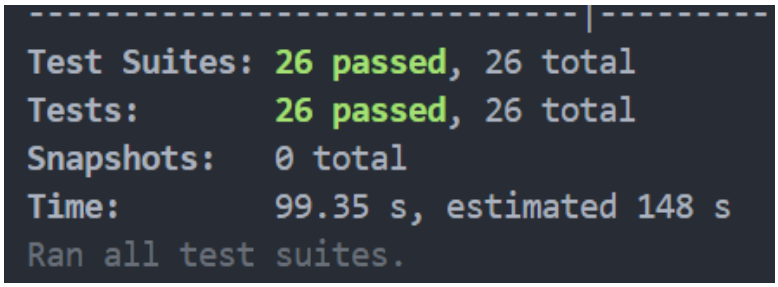
Berdasarkan contoh test case pada Tabel 3 hingga Tabel 15, tampak bahwa setiap scenario diuji dari dua sisi. Jalur positif memverifikasi sistem memberikan respons berhasil (kode 200/201) ketika masukan valid, sedangkan jalur negatif memastikan sistem menolak masukan tidak valid dengan kode kesalahan yang tepat (400 untuk data tidak valid, 401 untuk akses tanpa otorisasi, 403 untuk pelanggaran hak akses, 404 untuk data tidak ditemukan, serta 503 saat layanan eksternal tidak tersedia).

3.4 *Test Environtment Setup*

Lingkungan pengujian dikonfigurasi melalui instalasi *Jest* dan *dependensi* pendukung sebagai *devDependencies* menggunakan npm. Konfigurasi pengujian ditetapkan dalam berkas ***jest.config.js*** yang mencakup pengaturan lingkungan, cakupan, dan reporter. Struktur direktori pengujian disusun secara modular untuk memisahkan unit test dan integration test, dilengkapi folder setup, helpers, dan reporters.

3.5 *Test Execution*

Seluruh *test case* dieksekusi menggunakan *Command Line Interface* (CLI) Jest secara **sequential** untuk menghindari potensi konflik akses terhadap database selama proses *integration testing*. Pendekatan ini dipilih agar setiap skenario pengujian dijalankan secara berurutan, sehingga konsistensi data tetap terjaga.



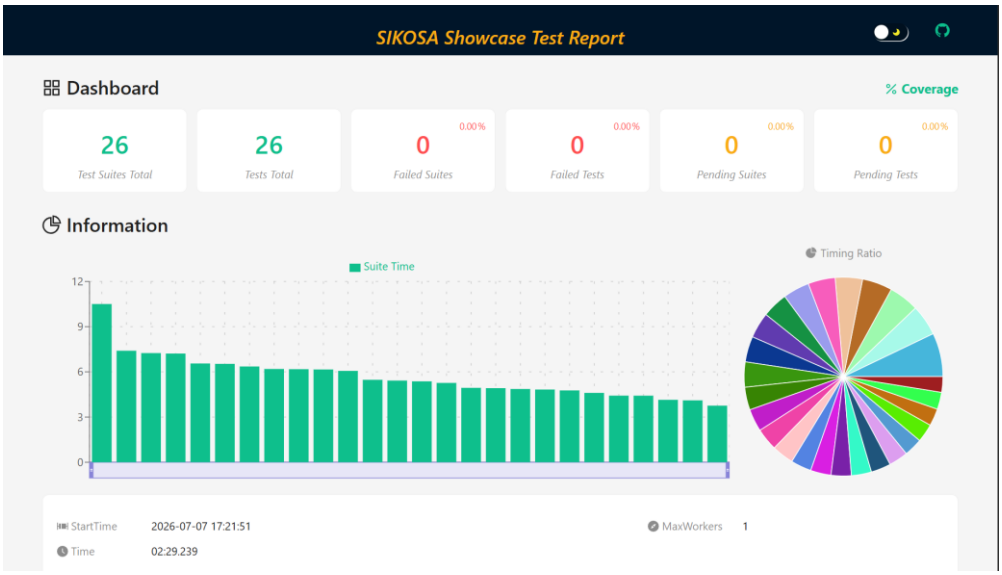
Gambar 2. Hasil eksekusi Jest

Berdasarkan hasil eksekusi, seluruh **26 test suite** yang berisi **26 test case** berhasil dijalankan dengan status **pass**, tanpa ditemukan adanya kegagalan (*failed test*) maupun pengujian yang dilewati (*skipped test*). Dengan demikian, tingkat keberhasilan pengujian (*test pass rate*) mencapai **100%**.

3.5 *Test Closre*

Berdasarkan hasil eksekusi seluruh skenario pengujian, diperoleh test pass rate sebesar 100%, di mana seluruh 26 test case yang tergabung dalam 26 test suite berhasil dieksekusi dengan status pass. Hasil tersebut menunjukkan bahwa seluruh fungsi yang menjadi objek pengujian telah berjalan sesuai dengan spesifikasi dan kebutuhan fungsional yang telah ditetapkan. Selain itu, tidak ditemukan kegagalan (*failed test*) maupun pengujian yang terlewat (*skipped test*), sehingga dapat disimpulkan bahwa implementasi sistem telah memenuhi tujuan pengujian serta memiliki tingkat keandalan yang baik berdasarkan cakupan pengujian yang dilakukan.

DOI https://doi.org/10.31004/riggs.v5i2.11863



Gambar 3. Dashboard laporan hasil pengujian

4. Kesimpulan

Penelitian ini berhasil menerapkan pengujian otomatis pada website Sistem Konseling Mahasiswa (SiKosa) menggunakan framework Jest dengan pendekatan *Software Testing Life Cycle* (STLC). Seluruh 26 test case pada 26 test suite berhasil dieksekusi dengan test pass rate 100%. Hasil pengujian mengonfirmasi bahwa validasi input, manajemen sesi, otorisasi berbasis peran, dan konsistensi data pada SiKosa telah berfungsi sesuai kebutuhan. Dengan demikian, penerapan pengujian otomatis berbasis Jest terbukti meningkatkan efisiensi serta konsistensi pengujian dan memenuhi standar cakupan pengujian yang memadai. Penelitian ini berhasil mengimplementasikan pengujian otomatis (*automated testing*) pada website Sistem Konseling Mahasiswa (SiKosa) menggunakan framework *Jest* dengan mengacu pada tahapan *Software Testing Life Cycle* (STLC). Proses pengujian dilaksanakan secara sistematis mulai dari tahap perencanaan, perancangan *test case*, implementasi skenario pengujian, pelaksanaan pengujian, hingga tahap evaluasi (*test closure*). Pendekatan tersebut memastikan bahwa setiap kebutuhan fungsional yang telah ditetapkan dapat divalidasi melalui serangkaian pengujian yang terstruktur dan terdokumentasi. Dengan demikian, penerapan pengujian otomatis berbasis *Jest* terbukti memberikan manfaat dalam meningkatkan efisiensi proses pengujian dibandingkan pengujian manual, karena setiap skenario dapat dieksekusi secara konsisten, berulang, dan terdokumentasi dengan baik. Selain mempercepat proses verifikasi setelah perubahan kode, pendekatan ini juga membantu menjaga kualitas perangkat lunak melalui deteksi dini terhadap potensi kesalahan (*regression*).

Referensi

[1] A. Sulma and A. Zacky W, "Pengaruh Kualitas Sistem Dan Layanan Terhadap Kepuasan Pengguna Aplikasi SIAPP di Kabupaten Sidenreng Rappang," *JURNAL TERAPAN PEMERINTAHAN MINANGKABAU*, vol. 3, no. 2, pp. 155–175, Nov. 2023, doi: 10.33701/jtpm.v3i2.3673.

[2] S. P. Ramadhani,) Farsya, A. Saputra, F. Dwiansyah, I. Veritawati, and R. Artikel, "Pengujian Sistem Informasi Akademik (NeoSiak) Berbasis Website Menggunakan Equivalence Partitioning dan Metode Black Box INFO ARTIKEL ABSTRAK," vol. 3, no. 1, p. 18, 2024, doi: 10.55123.

[3] F. Widianto, "Implementasi Model SDLC Dalam Perancangan Sistem Informasi Manajemen Perpustakaan Berbasis Web," 2024, doi: 10.69533.

[4] J. Sains, T. dan Kesehatan, R. Amaliyah Kamisr, and R. Artikel, "Studi Komparatif Pengujian Manual Dan Pengujian Otomatis Dengan Cypress Pada Website," vol. 2, no. 2, pp. 354–364, 2025, doi: 10.62335.

[5] J. Lian Min, A. Istiqomah, A. Rahmani, P. Negeri Bandung, and P. P. Tester Padepokan Tujuh Sembilan-Bandung, "Evaluasi Penggunaan Manual Dan Automated Software Testing Pada Pelaksanaan End-To-End Testing," *Jurnal Teknologi Terapan)* |, vol. 6, no. 1, 2020.

[6] F. Dhimas Wicaksono, "Rancang Bangun Automation Test Journey Pada E-Commerce," 2022.

[7] G. A. Prasetyo and N. Setiani, "Perbandingan Kinerja Framework Automation UI Testing Menggunakan The Distance to The Ideal Alternative," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 10, no. 1, pp. 224–237, Jan. 2025, doi: 10.29100/jipi.v10i1.5760.

[8] N. Hashemi, A. Tahir, S. Rasheed, A. Shi, and R. Blagojevic, "JS-TOD: Detecting Order-Dependent Flaky Tests in Jest," Aug. 2025, [Online]. Available: http://arxiv.org/abs/2509.00466

[9] Ruliansyah, Tukino, Baenil Huda, and April Lia Hananto, "Penerapan Software Testing Life Cycle Pada Pengujian Otomatisasi Platform Dzikra," *CSRID (Computer Science Research and Its Development Journal)*, vol. 15, no. 1, pp. 01–11, May 2023, doi: 10.22303/csrid.15.1.2023.01-11.

DOI https://doi.org/10.31004/riggs.v5i2.11863

[10] A. Arfan, "Penerapan STLC dalam Pengujian Automation Aplikasi Mobile (Studi kasus: LMS Amikom Center PT.GIT Solution)," 2022.

[11] A. Perbandingan Tools Pengujian Otomatis pada GUI Aplikasi Berbasis WEB Septi Melia, F. Profesio Putra, C. Author, and S. Melia, "Analisis Perbandingan Tools Pengujian Otomatis pada GUI Aplikasi Berbasis WEB," 2023. [Online]. Available: https://journal.irpi.or.id/index.php/sentimas

[12] A. N. Hasibuan and T. Dirgahayu, "Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna," 2021.

[13] F. W. Azhari and D. F. Suyatno, "Pengujian Otomatis GUI dengan Katalon Studio pada Situs Lowongan Kerja Jobstreet dan Glints," 2024.

[14] I. Gusti, N. Putu, D. Dicky Diastama, I. Made Sukarsa, N. Kadek, and A. Wirdiani, "Pengembangan Test Script untuk Load Testing Web dengan metode Software Testing Life Cycle," 2021.

[15] D. Widhyaestoeti *et al*., "Black Box Testing Equivalence Partitions Untuk Pengujian Front-End Pada Sistem Akademik Sitoda," 2021.

[16] J. Shadiq, A. Safei, R. Wahyudin Ratu Loly, C. sitasi, L. Rwr, and P. Aplikasi Peminjaman Kendaraan Operasional Kantor Menggunakan BlackBox Testing, "Pengujian Aplikasi Peminjaman Kendaraan Operasional Kantor Menggunakan BlackBox Testing," *Information Management for Educators and Professionals*, vol. 5, no. 2, pp. 97–110, 2021.

DOI https://doi.org/10.31004/riggs.v5i2.11863