PENGUJI OTOMATISASI PENILAIAN *SOURCE CODE* JAVASCRIPT DENGAN JEST DAN SUPERTEST PADA *NODE PACKAGE MANAGER* (NPM) UNTUK *PROGRAMMERS*

PROPOSAL SKRIPSI

Rahmat Sunjani

55201120030

PROGRAM STUDI TEKNIKI NFORMATIKA

FAKULTAS ILMU KOMPUTER DAN INFORMATIKA

UNIVERSITAS NURTANIO BANDUNG

2023

Kepada :

Yth. Bpk. Suharjanto Utomo, S.Si., M.T.

Ketua Program Studi Teknik Informatika (S1)

di tempat

Dengan Hormat,

Dengan ini saya yang bertanda tangan di bawah ini bermaksud mengajukan proposal Skripsi :

Nama	: Rahmat Sunjani
NPM	: 55201120030
Kelas	: IF’20 A
Telephone	: 083174506600
IPK	: -
SKS	: -

Saya menyatakan bersedia dan sanggup menyelesaikan skripsi saya sesuai dengan peraturan yang telah ditetapkan oleh pihak Universitas Nurtanio apabila proposal skripsi ini disetujui. Sebagai bahan pertimbangan, dengan ini saya lampirkan proposal skripsi saya yang berjudul “PENGUJI OTOMATISASI PENILAIAN *SOURCE CODE* JAVASCRIPT DENGAN JEST DAN SUPERTEST PADA *NODE PACKAGE MANAGER* (NPM) UNTUK *PROGRAMMERS*” Demikianlah surat pengajuan proposal skripsi ini saya buat, atas perhatian dan kebijaksanaan Bapak/Ibu, saya ucapkan terima kasih.

Bandung, 06 Februari 2024

Hormat saya

Rahmat Sunjani

55201120030

LEMBAR PERSETUJUAN

PROPOSAL SKRIPSI

Nama : Rahmat Sunjani

NPM : 55201120030

Kelas : IF’20 A

Telephone : 083174506600

Judul Tema : PENGUJI OTOMATISASI PENILAIAN *SOURCE CODE* JAVASCRIPT DENGAN JEST DAN SUPERTEST PADA *NODE PACKAGE MANAGER* (NPM) UNTUK *PROGRAMMERS*

Tanggal Persetujuan : 8 Januari 2024

Menyetujui,

Ketua Program Studi Teknik Informatika,

(Suharjanto Utomo, S.Si., MT)

ABSTRAK

Penelitian ini difokuskan pada implementasi metode *black box* pada *tools* penilaian *source code* otomatis dalam konteks bahasa pemrograman Javascript. Tujuannya adalah meningkatkan efisiensi proses penilaian *source code* dengan mengurangi waktu yang dibutuhkan. Sebagai proyek *open-source*, *tools* ini bertujuan untuk memperluas pemahaman komunitas dan memungkinkan kontribusi bersama guna meningkatkan kualitasnya. Dalam pengembangannya, *tools* ini memanfaatkan beberapa library yang disediakan oleh NPM. Metode yang digunakan untuk memajukan *tools* ini meliputi pendekatan *Black Box*. Proses desain model mengadopsi pendekatan *waterfall*, dan pengumpulan data dilakukan melalui serangkaian wawancara dengan pihak terkait. Hasil penelitian menunjukkan bahwa penerapan *tools* ini secara efektif meningkatkan kemampuan pengecekan *source code* Javascript dengan cepat dan efisien.

Kata Kunci : Penilaian Otomatis, Javascript, *Open-Source*, *Library* NPM, Metode *Black Box*, Model *waterfall*.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kode Sumber (*Source Code*) adalah komponen dasar dari program komputer yang dibuat oleh seorang *programmer*, yang sering kali ditulis dalam bentuk fungsi, deskripsi, definisi, pemanggilan, metode, dan pernyataan operasional lainnya. Kode ini dirancang agar dapat dibaca manusia dan diformat dengan cara yang dapat dimengerti oleh pengembang dan pengguna lain[1].

Penilaian (*Grading*) adalah proses penyematan atribut atau dimensi atau kuantitas (berupa angka/huruf) terhadap hasil asesmen dengan cara membandingkannya terhadap suatu instrumen standar tertentu. Hasil dari penilaian berupa atribut/dimensi/kuantitas tersebut digunakan sebagai bahan evaluasi[2].

Penilaian kode sumber adalah proses evaluasi kualitas dan kelayakan kode sumber yang telah dibuat oleh seorang *programmers*. Ini melibatkan pemeriksaan terhadap berbagai aspek seperti kejelasan, efisiensi, keamanan, dan pemeliharaan kode.

Dengan menyusun latar belakang ini, peneliti akan melakukan implementasi pembuatan *library* untuk penguji *programmers*. *Library* akan dirancang untuk efisiensi dan meminimalkan waktu yang dibutuhkan untuk mengevaluasi jawaban tes secara cepat dan efisien sehingga kedepannya tools ini akan sangat membantu dalam pengembangan tentang penilaian kode sumber.

Peneliti sangat berharap untuk peneliti selanjutnya dapat meningkatkan fitur terbaru dari *library* penilaian kode sumber baik untuk pengembangan atau perbaiki bug yang terdapat pada sistem saat ini. Project ini akan tersimpan pada github dan terdaftar di web *Node Package Manager* (NPM) dan bersifat open source untuk digunakan.

1.2 Rumusan masalah

Adapun rumusan masalah dari penelitian ini, antara lain :

1. Bagaimana perancangan *library* penilaian otomatis kode sumber untuk penguji dengan *programmers*?
2. Bagaimana cara praktis *library* pada *Node Package Manager* yang akan diterapkan untuk penilaian kode Javascript?
3. Apa manfaat yang dapat diharapkan dari implementasi penilaian kode sumber ini dalam pengembangan perangkat lunak?

1.3 Batasan masalah

Pada penelitian ini adapun batasan masalahnya, antara lain :

1. Pengujian ini akan dilakukan dengan bantuan *library* yang menggunakan bahasa pemrograman Javascript.
2. Pengujian berfokus pada mengecek kode sumber bahasa pemrograman Javascript.
3. Pembuatan library akan didukung dengan metode *Black Box*.

1.4 Tujuan penelitian

Adapun tujuan dari penelitian ini, antara lain :

1. Meningkatkan efisiensi dan konsistensi penilaian kode sumber.
2. Memudahkan penguji dan *programmers* dalam evaluasi kualitas kode sumber.
3. Meningkatkan transparansi dan objektivitas dalam penilaian kode sumber.

1.5 Rencana dan jadwal kegiatan

Tabel 1.1 Rencana dan Jadwal Kegiatan

Kegiatan	Bulan					
	1	2	3	4	5	6
Literatur Review						
Analisis & Perancangan Sistem						
Pengengembangan Pembuatan Library						
Testing Perancangan Library						
Implementasi Library						
Evaluasi & Pembarahuan System						

BAB II
TEORI DASAR

2.1 Kode Sumber

Kode Sumber (*Source Code*) adalah dasar dari rancangan suatu program yang berisi kumpulan baris teks instruksi dan kode-kode fungsi yang mengkomunikasikan suatu perintah yang harus dijalankan oleh program agar program tersebut berfungsi sesuai tujuan perancangan.[3].

2.2 Node Package Manager

Node Package Manager (NPM) adalah manajer paket yang banyak digunakan untuk JavaScript yang dapat menyederhanakan proses pengelolaan ketergantungan dalam proyek pengembangan. NPM memungkinkan pengembang untuk menginstal, memperbarui, dan mengelola pustaka dan alat pihak ketiga dengan mudah.[4].

2.3 Node.js

Node.js adalah lingkungan *runtime* JavaScript sisi *server* yang memungkinkan pengembang membangun aplikasi yang dapat diskalakan dan berkinerja tinggi. Node.js menggunakan model I/O yang digerakkan oleh peristiwa dan tidak memblokir, sehingga sangat cocok untuk menangani permintaan yang bersamaan. Node.js memiliki ekosistem modul yang luas yang tersedia melalui NPM, yang memungkinkan pengembang untuk memperluas fungsionalitasnya.[4].

2.4 Jest

Jest adalah *framework* pengujian JavaScript yang dikembangkan oleh Facebook. Ini bekerja di luar kotak dengan konfigurasi minimal dan memiliki *in-built test runner*, perpustakaan penegasan dan dukungan *mocking*[5].

2.5 Supertest

Supertest merupakan *library* untuk menguji *server* HTTP Node.js. Ini memungkinkan kami untuk mengirim permintaan HTTP secara terprogram

seperti GET, POST, PATCH, PUT, DELETE ke *server* HTTP dan mendapatkan hasil[5].

2.6 Black Box

Pengujian Black Box adalah metode di mana fungsionalitas aplikasi perangkat lunak dievaluasi tanpa melihat kode internalnya. Metode ini memeriksa apakah perangkat lunak berperilaku seperti yang diharapkan berdasarkan persyaratan, masukan, dan keluaran. Begini cara kerjanya bekerja: penguji memberikan input ke perangkat lunak dan mengamati output, membandingkannya dengan hasil yang diharapkan. Ini membantu memastikan bahwa perangkat lunak memenuhi fungsionalitas yang diinginkan tanpa perlu mengetahui bagaimana perangkat lunak tersebut dibuat secara internal[6].

BAB III

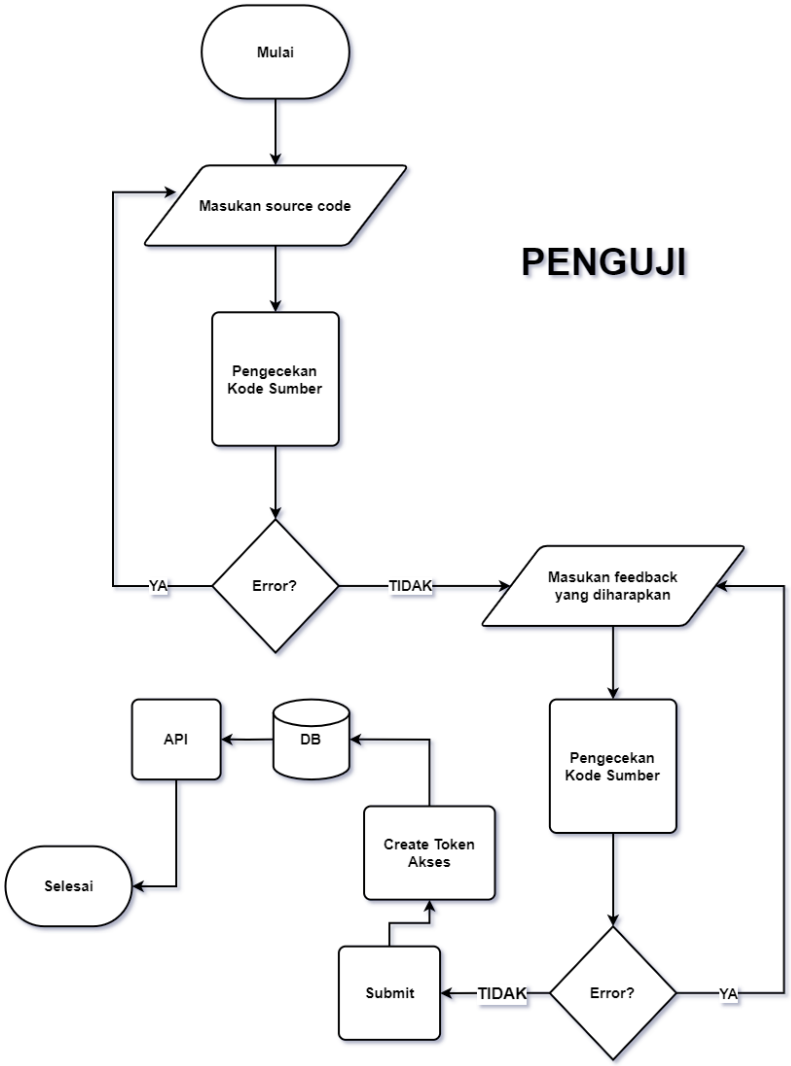
PERANCANGAN SISTEM DAN ALUR PEMODELAN

3.1 Perancangan Sistem

3.1.1 Flowchart Sistem

Flowchart Sistem adalah *flowchart* yang menampilkan tahapan atau proses kerja yang sedang berlangsung di dalam sistem secara menyeluruh. Selain itu *flowchart* sistem juga menguraikan urutan dari setiap prosedur yang ada di dalam sistem[7].

Gambar 3.1 Flowchart Sistem Penguji



Penjelasan *Flowchart* Sistem Penguji :

1. **Penguji Memasukan Kode Sumber**

   Dosen akan menyertakan kode sumber yang akan menjadi referensi penilaian untuk menilai akuratan kode sumber *programmer*.

2. **Pengecekan Kode Sumber**

   Dalam tahap ini kode sumber akan dicek menggunakan format yang umum digunkan programmer dengan bantuan *library* prettier sehingga syntax akan lebih umum digunakan, selanjutnya akan masuk proses pengecekan kode sumber secara teknis sehingga saat melakukan submit program tidak akan terkendala dengan error.

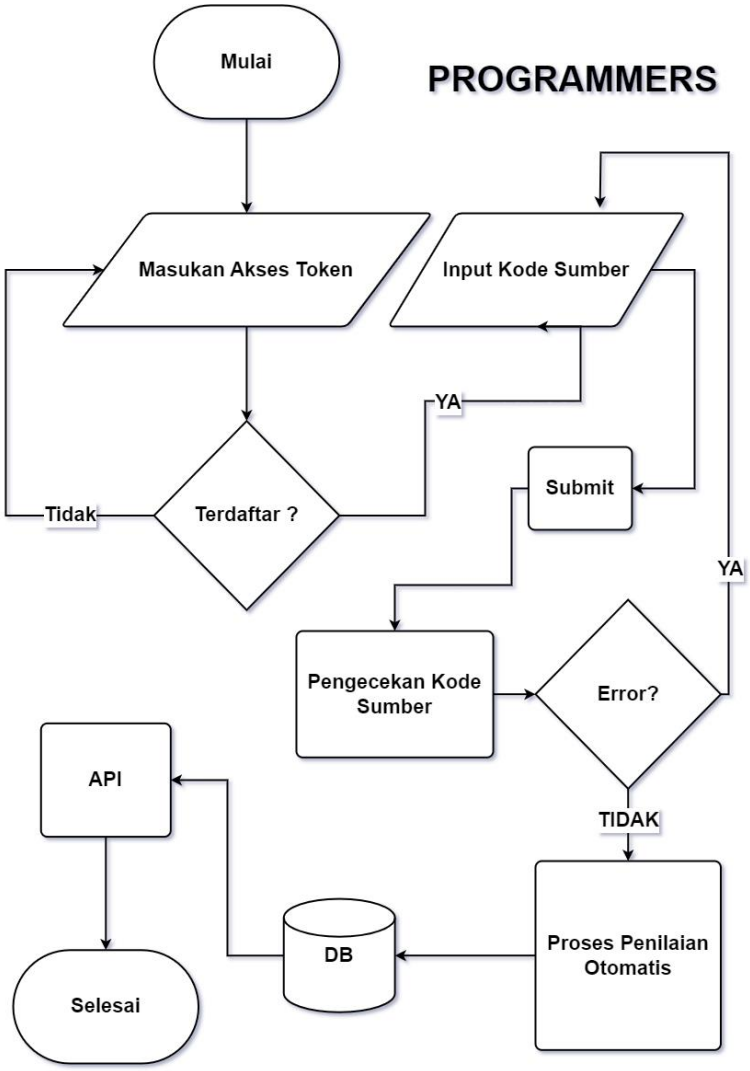
3. **Masukan Feedback Yang Diharapkan**

   Penguji dalam tahap ini akan melakukan tahap memasukan feedback yang diharapkan. Namun disini bukan feedback hasil yang benar-benar konstanta namun hasilyang sudah diatur dengan variabel tertentu pada program.

4. **Submit**

   Setelah selesai melakukan pengecekan pada kode sumber selanjutnya data akan disimpan dalam database. Sebelum dimasukkan ke database, akan otomatis menghasilkan sebuah akses token yang akan digunakan oleh *programmers* untuk mendapatkan soal. Setelah tersimpan dalam database, semua informasi akan terhubung melalui API dengan bantuan *library express* untuk mempermudah akses terhadap database.

Gambar 3.2 *Flowchart* Sistem *Programmers*



Penjelasan *Flowchart* Sistem *Programmers* :

1. **Masukan Akses Token**

   Penguji harus memiliki akses token yang telah disiapkan sebelumnya untuk memasukkan jawaban. Jika penggunaan token tidak berhasil, akan diminta untuk dimasukkan kembali; namun jika berhasil, pengguna akan diarahkan langsung ke menu *input* jawaban kode sumber yang harus diisi oleh *programmers*.

2. **Input Kode Sumber**

   Sebelum memasuki database, kode sumber akan diperiksa untuk meminimalisir kesalahan sintaksis jika terdapat error maka data tidak dapat dimasukan ke database. Kode sumber yang telah dikirimkan ke database tidak dapat diubah. Selanjutnya, para programmer akan menerima penilaian secara langsung, dan hasilnya hanya dapat dilihat oleh para programmer dengan mengambil data melalui API database.

3. **Proses Penilaian Otomatis**

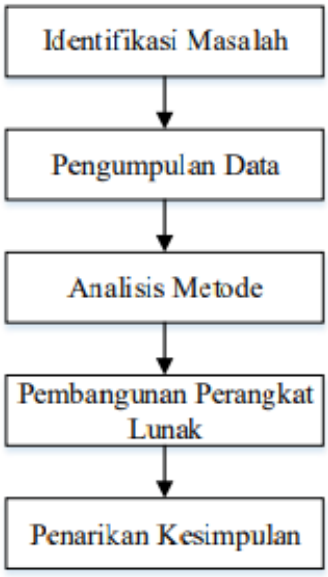
   Pada proses evaluasi source code otomatis, akan dilibatkan beberapa metode yang akan diaplikasikan beserta penerapan beberapa perpustakaan (library) dari NPM yang akan dijelaskan dalam Alur Pemodelan.

3.2 Alur Pemodelan

3.2.1 Alur Penelitian

Alur penelitian pada penelitian ini mencakup lima tahap, yaitu identifikasi masalah, pengumpulan data, analisis metode, pembangunan perangkat lunak, dan penarikan kesimpulan. Alur penelitian dapat dilihat pada Gambar 3.3.

Gambar 3.3 Alur Penelitian



Penjelasan dari alur penelitian pada Gambar 3.3 adalah sebagai berikut.

1. **Identifikasi Masalah**

   Tahap identifikasi masalah merupakan proses pengamatan terhadap penelitian yang sudah pernah dilakukan sebelumnya untuk menentukan kebutuhan dan tujuan sistem yang akan dicapai.

2. **Pengumpulan Data**

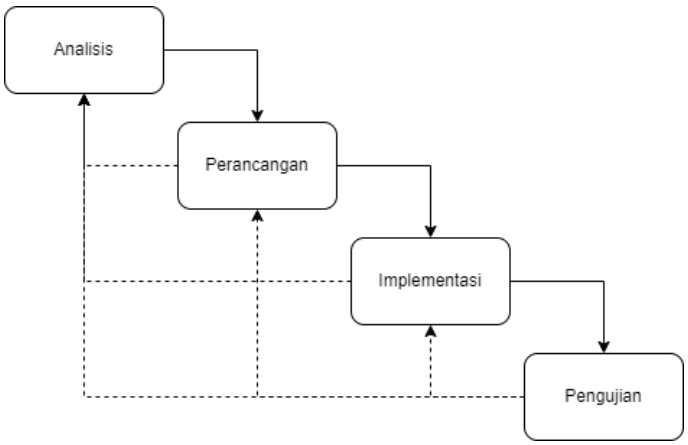
   Pada penelitian ini, pengumpulan data yang dilakukan dengan metode wawancara.

3. **Analisis Metode**

   Pada tahap analisis metode, metode yang digunakan dalam penelitian ini akan dianalisis, mulai dari tahap analisis penerapan metode dan penerapan metode serta dengan dukungan library NPM.

4. **Pembangunan Perangkat Lunak**

   Pada penelitian ini, pembangunan perangkat lunak yang digunakan adalah model *waterfall*. Model *waterfall* merupakan metode pembangunan perangkat lunak yang bersifat sekuensial dalam tiap prosesnya[8]. Alur dari model *waterfall* dapat dilihat pada Gambar 3.4.



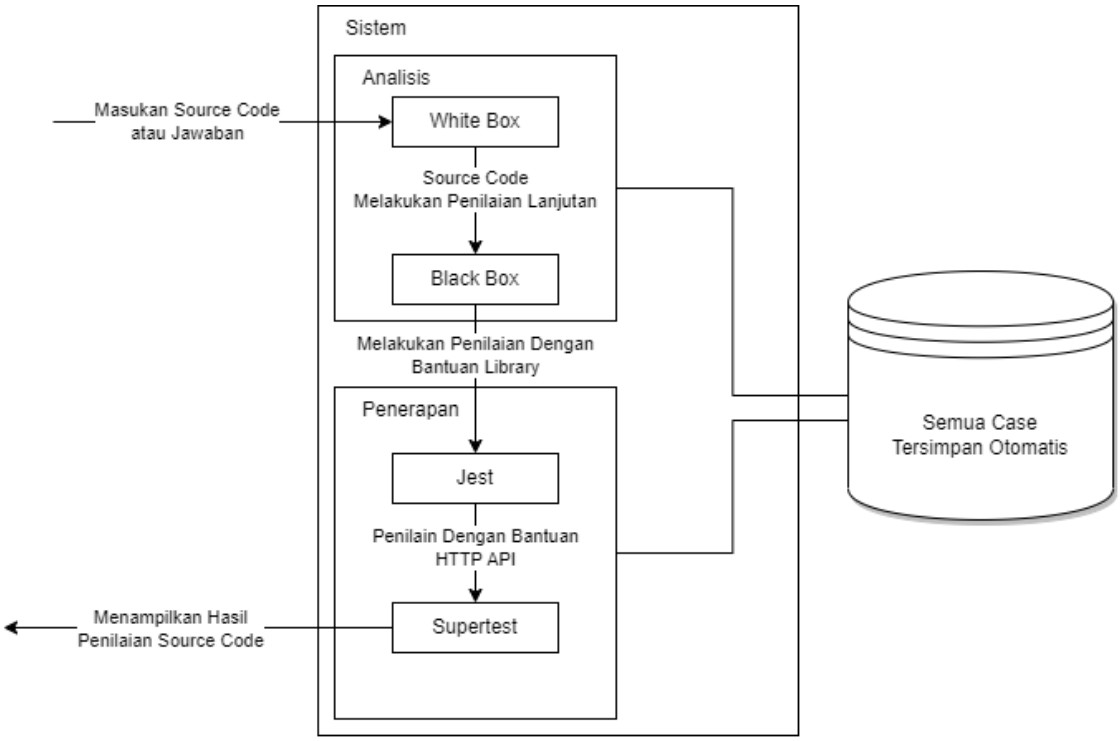
Gambar 3.4 Model *Waterfall*

5. Penarikan Kesimpulan

   Pada tahap penarikan kesimpulan merupakan penjelasan hasil dari penelitian yang sudah dilakukan.

3.2.2 Gambaran Umum Sistem

Dalam penelitian ini, sistem yang dikembangkan memiliki kemampuan untuk mengevaluasi kode sumber yang ditulis dalam bahasa JavaScript. Sistem ini terdiri dari dua tahap utama, yaitu analisis penerapan metode dan penerapan metode pada perpustakaan NPM. Proses analisis melibatkan langkah-langkah implementasi *Black Box*, sementara proses penerapan metode mencakup tahap pembuatan kode yang didukung oleh perpustakaan yang tersedia di NPM.



Gambar 3.5 Blok Diagram Sistem Keseluruhan

DAFTAR PUSTAKA

[1] Scott Wallask, “Source Code,” www.techtarget.com. Diakses: 13 Desember 2023. [Daring]. Tersedia pada: https://www.techtarget.com/searchapparchitecture/definition/source-code

[2] Universitas Islam Indonesia, “Pengantar Asesmen, Penilaian, dan Evaluasi Pembelajaran,” https://dpa.uii.ac.id. Diakses: 6 Februari 2024. [Daring]. Tersedia pada: https://dpa.uii.ac.id/pengantar-asesmen-penilaian-evaluasi/#:~:text=Penilaian%20(grading)%20adalah%20proses%20penyematan,tersebut%20digunakan%20sebagai%20bahan%20evaluasi.

[3] Dwi Arizki Verdianto, “Source Code,” teknogram.id. Diakses: 14 Desember 2023. [Daring]. Tersedia pada: https://teknogram.id/kamus/source-code/

[4] N. A. Alhazmy, Z. N. Chandra, P. Atmadiputra, dan Y. Triyana, “Building a Comprehensive Content Management System with NPM, Vue.js, Node.js, Postgresql, and Strap,” 2023.

[5] Chinedu Orie, “Testing NodeJs/Express API with Jest and Supertest,” dev.to. Diakses: 15 Desember 2023. [Daring]. Tersedia pada: https://dev.to/nedsoft/testing-nodejs-express-api-with-jest-and-supertest-1km6

[6] Prof. S.K.Totade, Trupti Tayde, dan Pranali Dhole, “Black Box Testing,” *f Innovations in Engineering and Technology (IRJIET)*, vol. Volume 7, 2023, doi: https://doi.org/10.47001/IRJIET/2023.710089.

[7] Rony Setiawan, “Flowchart Adalah: Fungsi, Jenis, Simbol, dan Contohnya,” www.dicoding.com. Diakses: 15 Desember 2023. [Daring]. Tersedia pada: https://www.dicoding.com/blog/flowchart-adalah/

[8] S. P. Roger dan R. M. Bruce, *Software engineering: a practitioner’s approach*. McGraw-Hill Education, 2015. Diakses: 15 Desember 2023. [Daring]. Tersedia pada: https://dspace.agu.edu.vn/handle/agu_library/13103