BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Penelitian dan pengembangan yang dilakukan oleh peneliti saat ini berdasarkan pada pendekatan Design and Development (D&D) yang dikembangkan oleh Ellis dan Levy (2010). Design and Development (D&D) merupakan sebuah studi sistematis yang didalamnya terdapat proses desain, pengembangan, dan evaluasi yang bertujuan untuk menetapkan hasil dasar empiris dalam menciptakan produk dan alat bantu intruksional ataupun non instruksional serta model baru atau hasil penyempurnaan dari pengembangan terdahulu. Seperti definisi yang dikemukakan Ellis dan Levy yaitu D&D diartikan sebagai "the systematic study of design, development and evaluation processes with the aim of establishing an empiricalbasis for the creation of instructional and non-instructional products and tools and new or enhanced models that govern their development" (Ellis dan Levy, 2010). Metode yang diimplementasikan pada penelitian ini sesuai dengan tujuan untuk mengembangkan dan mengevaluasi sebuah sistem berupa sistem deteksi SQL Injection berbasis aplikasi web dengan menggunakan algoritma Deep Learning salah satunya Text Convolutional Neural Network (TextCNN). Dalam hal ini Richey dan Klein ini terdiri dari enam tahapan utama yang beurutan guna memastikan pengembangan sistem yang terstruktur dan teruji. Peffers dkk. (dalam Ellis dan Levy, 2010) mengemukakan tahapan dalam penelitian D&D yang terdiri dari 6 fase seperti pada bagan berikut ini :

Gambar 3. 1 Prosedur Penelitian ((Ellis dan Levy, 2010)

Pada bagan diatas, tahapan dalam penelitian D&D terdiri dari identifikasi masalah penelitian, mendeskripsikan tujuan penelitian,

merancang dan mengembangkan produk, melakukan uji coba, melakukan evaluasi sistem dan menyampaikan hasil uji coba yang sudah dievaluasi.

3.2 Identifikasi Masalah

Tahap identifikasi masalah pada metodologi penelitian Design and Development (D&D) merupakan pondasi awal yang sangat penting sebelum memasuki proses perancangan. Ancaman serangan *SQL Injection* menjadi salah satu serangan populer dikalangan kejahatan siber.Serangan ini memanfaatkan celah input validasi yang dapat menyebabkan data sensitif bocor dan gangguan sistem jaringan.

Implementasi sistem keamanan telah banyak dilakukan oleh berbagai penelitian, namun kerentanan SSQL Injection tetap sulit diatasi. Kesulitan ini disebabkan banyaknya variasi baru yang terus berkembang dari SQL Injection yang lebih bervariasi, sehingga keterbatasan metode deteksi terdahulu dapat dikembangkan celah keamanannya. Sebagian besar peneliti terdahulu menghadapi keterbasan. Akan tetapi, sebagian besar peneliti masih menghadapi keterbatasan seperti akurasi, yang menurun saat dihadapkan pada data yang berjumlah besar, tingginya tingkat false positive, atau kurangnya kemampuan model dalam mengeksrasi pola semantic

Model TextCNN dirancang untuk memproses data berbasis teks dengan menggunakan teknik ekstraksi fitur lokal melalui lapisan konvolusi, sehingga model mampu menangkap pola penting dari urutan kata atau token dalam kueri SQL. TextCNN mampu untuk melakukan ektraksi fitur secara otomatis, berbeda dengan model machine learning yang bergantung pada rekayasa fitur manual. Keunggulan model TextCNN untuk menjadikannya algoritma sistem deteksi SQL Injection lebih efisien serta adaptif terhadap variasi serangan. Penerapan TextCNN secara aplikasi hanya berbatas aplikasi web

3.3 Deskripsi Tujuan

Berdasarkan permasalahan tersebut, tujuan penelitian ini adalah:

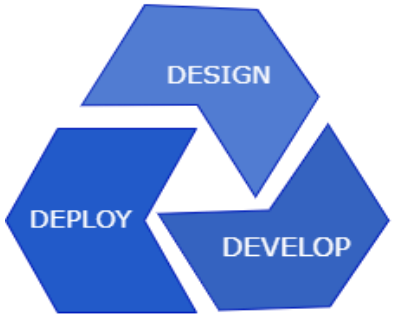
1. Penelitian yang diusulkan akan diarahkan pada perancangan arsitektur sistem deteksi SQL Injection yang mampu mengintegrasikan representasi teks dengan model klasifikasi TextCNN melalui tokenisasi.
2. Untuk penerapan sistem deteksi serangan SQL Injection dalam aplikasi berbasis web yang dapat langsung menganalisis kueri yang diberikan oleh pengguna.
3. Untuk menguji kinerja sistem menggunakan dataset kueri SQL normal dan berbahaya.
4. Penilaian hasil pengujian terhadap kinerja dan efektivitas sistem dalam mendeteksi berbagai jenis serangan SQL Injection.

3.4 Desain dan Pengembangan

Pada tahap ini, fokus penelitian terletak pada proses desain dan pengembangan yang akan diadopsi dalam penelitian. Desain dilakukan agar setiap langkah dalam proses pengembangan terstruktur dengan baik. Desain yang baik membantu mengurangi ketergantungan yang ada antara komponen sistem sedemikian rupa sehingga jika terjadi kesalahan pada salah satu modul, hal itu tidak akan memengaruhi sistem secara keseluruhan.

3.4.1 Metode Pengembangan Model

Proses desain dan pengembangan akan dianggap sebagai fase utama dari penelitian ini. Temuan dari tahap-tahap sebelumnya kemudian akan diwujudkan selama fase penelitian ini. Fase ini akan mengadopsi model pengembangan Siklus Hidup Kecerdasan Buatan, seperti yang digambarkan oleh De Silva, D., & Alahakoon, D., 2022, dalam fase desain dan pengembangan sistem AI mereka, seperti yang ditunjukkan pada Gambar 3.2 di bawah ini:



Gambar 3. 2 Artificial Intelligence Life Cycle (De Silva, D., & Alahakoon, D.

2022)

Pada gambar 3.2 menampilkan metode AILC yang merupakan bagian penting dari model yang dirancang dan dikembangkan berdasarkan Kecerdasan Buatan karena kontribusinya yang signifikan dalam membangun sistem deteksi serangan SQL Injection berdasarkan Kecerdasan Buatan dan TextCNN. *Artificial Intelligence Life Cycle* memiliki tiga tahap utama yaitu desain, pengembangan, dan implementasi. *Artificial Intelligence Life Cycle* membantu mengembangkan sistem deteksi serangan *SQL Injection* berbasis web menggunakan TextCNN karena TextCNN adalah jenis kecerdasan buatan dan telah digunakan untuk membangun sistem deteksi serangan *SQL Injection* berbasis Kecerdasan Buatan.

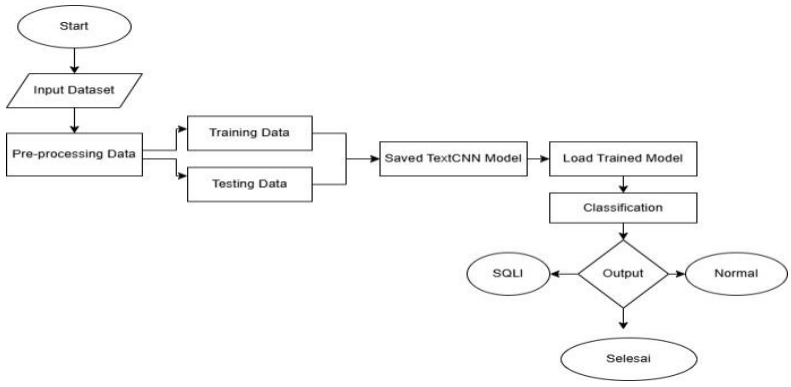
Pada tahap pengembangan, TextCNN digunakan sebagai model dalam menganalisis fitur lokal dan aspek sintaksis yang terdapat dalam kueri SQL. Hal ini dilakukan melalui pelatihan dan pengujian model, dengan tujuan untuk mendapatkan model yang optimal. Alasan pemilihan TextCNN adalah efisiensinya dalam ekstraksi fitur lokal dari teks secara cepat, sehingga memungkinkan identifikasi pola serangan *SQL Injection* tanpa melibatkan aturan yang telah ditentukan sebelumnya. Proses evaluasi model menggunakan data uji untuk mengukur akurasi, presisi,

dan kemampuan generalisasi dalam mengidentifikasi serangan *SQL Injection*, termasuk pola yang tidak dikenal.

Model TextCNN diimplementasikan dalam sistem aplikasi web sebagai bagian dari fase penyebaran untuk bertindak sebagai mekanisme deteksi serangan *SQL Injection* secara *real-time*. Peran model ini adalah untuk mengevaluasi kueri SQL yang dikirimkan pengguna untuk indikasi potensi serangan sebelum pemrosesan basis data yang sebenarnya. Akurasi model juga akan dipantau secara cermat untuk memprediksi potensi perkembangan pola serangan dan pergeseran dalam data yang dapat memengaruhi akurasi deteksi. Seperti yang dipahami oleh *Artificial Intellgence Life Cycle* (AILC), penelitian ini mencakup tidak hanya deteksi yang akurat tetapi juga solusi aplikasi web untuk mengatasi ancaman dinamis yang selalu berubah ini yang disebut SQL Injection.

3.4.1.1 Perancangan Model

Desain arsitektur model deteksi SQL Injection berbasis TEXTCNN dilakukan sebelum implementasi sistem untuk memastikan pendekatan implementasi yang sistematis dan terstruktur. Arsitektur model yang dirancang oleh penelitian ini dapat direpresentasikan oleh diagram alir pada Gambar 3.3.



Gambar 3. 3 Flowchart Alur Pengembangan Model

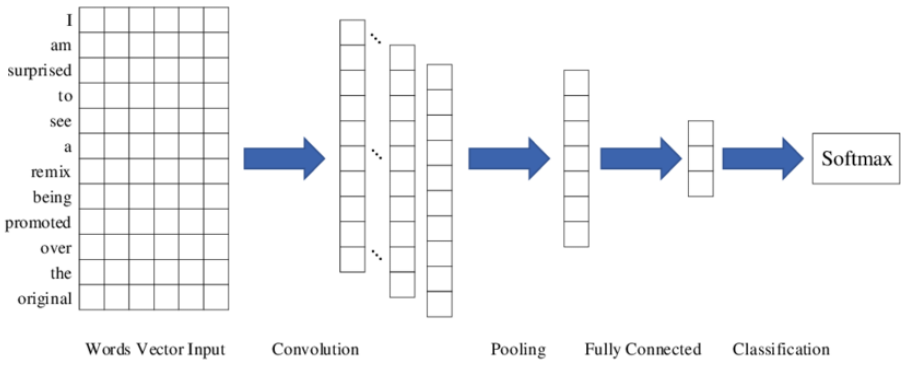
Diagram alir perancangan model deteksi *SQL Injection* berdasarkan tipe *SQL Injection* menggunakan TextCNN dirancang secara sistematis berdasarkan diagram alir, dengan memberikan perhatian khusus pada pemanfaatan TextCNN, yang merupakan *Convolutional Neural Network* yang dirancang untuk teks. Proses dimulai pada input dataset, di mana terdapat kumpulan kueri SQL berlabel, baik tipe normal maupun tipe *SQL Injection*. Proses berlanjut pada preprocessing data, di mana keseragaman data dipastikan melalui berbagai proses pembersihan: data kosong dihapus, semua teks diubah menjadi huruf kapital, dan spasi yang berlebihan dihapus. Dataset kemudian di tokenisasi melalui penggunaan Keras Tokenizer dengan batas optimal 5.000 kata dan penggunaan <OOV> untuk merepresentasikan kata-kata yang berada di luar kosakata yang dipelajari dari data selama pelatihan. Proses *padding* dilakukan untuk memastikan panjang yang sama yaitu 43 kata untuk dimasukkan ke dalam input model. Ekstraksi fitur dilakukan secara implisit melalui lapisan embedding TextCNN untuk memperoleh Data Pelatihan dan Pengujian.

Lapisan embedding dengan dimensi 128 berfungsi sebagai titik awal untuk struktur model TextCNN, yang diikuti oleh tiga lapisan konvolusi untuk model ini. Lapisan konvolusi ini menerapkan kernel dengan ukuran berbeda – ukuran 3, ukuran 4, dan ukuran 5, masing-masing berisi 128 filter untuk mengekstrak fitur lokal dari kueri, termasuk urutan kata kunci SQLi. Hasil dari setiap lapisan konvolusi dimasukkan melalui lapisan *Global Max Pooling* dan digabungkan. Kemudian diikuti oleh lapisan dropout dengan tingkat 0,5 untuk mencegah *overfitting*. Selanjutnya diikuti oleh lapisan Dense dengan fungsi aktivasi ReLU dan total 64 neuron. Akhirnya dihubungkan ke lapisan akhir dengan fungsi aktivasi Sigmoid untuk klasifikasi biner. Model ini telah dilatih dengan fungsi entropi silang biner sebagai fungsi biaya bersama dengan pengoptimal Adam untuk mendapatkan model yang tersimpan (*Saved TextCNN Model*).

Model dimuat kembali untuk mengklasifikasikan kueri SQL baru. Pada langkah Klasifikasi ini, kueri SQL baru diproses dengan cara yang

sama, termasuk tokenisasi, padding, dan klasifikasi akhir oleh model. Model memeriksa pola dalam kata, simbol, dan struktur dalam kueri untuk klasifikasi sebagai serangan aman atau serangan yang mungkin terjadi dalam *SQL Injection*. Hasilnya adalah klasifikasi biner, "Normal" untuk kueri aman dan "SQLi" untuk penyerang.

Evaluasi dilakukan pada data uji, dan akurasi, presisi, recall, dan skor F1 diukur untuk memastikan deteksi serangan *SQL Injection* yang akurat dan *false positive* yang rendah. Model ini bertujuan untuk mendeteksi berbagai jenis serangan. Deteksi berbasis TextCNN untuk mendeteksi serangan baru yang tidak ada dalam data pelatihan secara *real-time*.



Gambar 3. 4 Arsitektur *CNN for Text Classification* (Tan, J., Miao, D., & Tan, Q,. 2019).

Gambar 3.4 menggambarkan arsitektur yang digunakan untuk mengidentifikasi fitur-fitur penting dari urutan kata atau token dalam sebuah kueri SQL, diikuti oleh klasifikasi untuk mengklasifikasikan apakah kueri SQL tersebut normal atau merupakan serangan *SQL Injection*. Sebagai input ke jaringan, terdapat urutan token dengan batasan panjang maksimum tetap (maxlen=43 token). Dengan memanfaatkan lapisan *embedding*, setiap token dalam kueri SQL dikonversi menjadi vektor 128 dimensi untuk mempelajari representasi setiap token dalam kueri SQL.

Fitur *embedding* ini diikuti oleh alur tiga lapisan konvolusi satu dimensi (Conv1D), masing-masing dengan ukuran kernel 3, 4, dan 5, dengan 128 filter untuk setiap lapisan. Penggunaan lapisan konvolusi ini dimaksudkan untuk mengekstrak pola lokal atau fokus pada n-gram dengan ukuran yang bervariasi seperti kata kunci, operator logika, dan pola kueri yang lazim dalam serangan *SQL injection*.

Hasil dari setiap lapisan konvolusi kemudian diteruskan ke tahap *Global Max Pooling*, di mana nilai-nilai fitur terkuat diekstrak berdasarkan filter. Hasil dari ketiga lapisan tersebut kemudian digabungkan, menghasilkan total 384 fitur utama, yang kemudian dihubungkan ke lapisan dense dengan 64 neuron.

Pada tahap akhir, melibatkan penggunaan lapisan sigmoid dengan satu neuron untuk output, yang menghasilkan ukuran probabilitas berkisar dari 0 hingga 1. Berdasarkan ukuran probabilitas yang dihasilkan, jika nilainya di atas 0,5, kueri tersebut dikatakan sebagai serangan *SQL Injection* (SQLi). Sebaliknya, itu adalah kueri normal jika ukurannya 0,5 atau lebih rendah.

Secara umum, pelatihan model mencakup pembentukan parameter yang dapat dilatih berupa bobot pada lapisan embedding, kernel konvolusi, dan koneksi antar neuron pada lapisan dense. Desain dan struktur jaringan ini digunakan karena kemampuan mekanisme konvolusi dalam mengidentifikasi karakteristik serangan SQL Injection pada data. Hal ini dapat dikaitkan dengan karakteristik data berupa frasa kunci yang telah ditentukan sebelumnya dengan karakter khusus.

3.4.1.2 Pengembangan Model

Pengembangan model dilakukan berdasarkan *AI Life Cycle* mencakup desain arsitektur serta proses pelatihan dan evaluasi.

1) Data Acquisition

Penelitian ini menggunakan dataset publik yang telah tersedia sebelumnya pada *kaggle* dan diperoleh dari repositori *GitHub* milik

ajinmathew. Dataset diunduh secara langsung dari sumber daring menggunakan protokol HTTP dan disimpan dalam format CSV. Data diambil dari dataset yang bersumber dari yang berisi kumpulan kueri SQL normal serta kueri yang mengandung serangan SQL Injection. Dataset ini selanjutnya digunakan sebagai data latih dan data uji dalam proses pelatihan serta pengujian model deteksi SQL Injection.

SQL Injection yang berbeda memiliki pola dan cara kerja yang berbeda. Berikut adalah beberapa contoh jenis serangan SQL Injection yang mungkin menjadi fokus deteksi penelitian ini:

a) Classic SQL Injection

Tipe dasar dari *SQL Injection* di mana penyerang menyisipkan perintah SQL berbahaya ke dalam input pengguna, yang kemudian dieksekusi oleh server database. Biasanya, serangan ini memanfaatkan celah dalam aplikasi yang tidak melakukan validasi atau pemfilteran terhadap input pengguna.

contohnya dengan pengguna dapat memasukkan perintah SQL seperti ' OR 1=1 --, yang sering digunakan untuk mendapatkan akses ke basis data atau melewati kontrol login. Dalam hal ini, model TextCNN akan dilatih untuk mengenali pola karakteristik yang ada dalam kueri-kueri SQL yang berbahaya seperti penggunaan operator logika seperti OR, AND, dan tanda komentar --.

b) Blind SQL Injection

Blind SQL Injection terjadi ketika aplikasi tidak langsung mengembalikan data yang diminta oleh penyerang, tetapi penyerang dapat mengamati perbedaan respons aplikasi untuk membocorkan informasi tentang struktur basis data atau data yang disimpan. Penyerang mungkin memasukkan kueri seperti ' AND 1=1, dan aplikasi tidak akan menampilkan pesan kesalahan, namun akan menunjukkan perubahan dalam tampilan halaman. Sistem harus dapat mendeteksi pola-pola perbedaan kueri yang mengandung logika perbandingan atau percabangan (seperti AND, OR, =, atau perbandingan boolean lainnya).

c) Union Based SQL Injection

Dalam jenis serangan ini, penyerang menggunakan perintah UNION untuk menggabungkan hasil dari beberapa kueri SQL untuk mengekstrak data dari tabel yang tidak ada hubungannya dengan kueri asli. Penyerang mungkin menggunakan kueri seperti UNION SELECT username, password FROM users. Model TextCNN dapat dilatih untuk mengenali pola penggunaan kata kunci seperti UNION, yang dapat menunjukkan potensi serangan dengan menggabungkan hasil dari beberapa kueri.

d) Error Based SQL Injection

Serangan ini terjadi ketika penyerang menyebabkan kesalahan SQL di aplikasi untuk memanipulasi sistem dan mengambil informasi mengenai struktur database atau data yang ada. Penyerang dapat mengirimkan input yang mengarah pada kesalahan SQL, seperti 1' AND 1=1; --, yang memaksa database untuk mengembalikan pesan kesalahan yang dapat digunakan oleh penyerang untuk memperoleh informasi lebih lanjut. Sistem deteksi harus mampu mengenali input yang berpotensi menyebabkan kesalahan dalam kueri dan memverifikasi apakah pesan yang dihasilkan menunjukkan pola-pola kesalahan.

e) Time Based Blind SQL Injection

Pada serangan ini, penyerang mengirimkan perintah yang akan menyebabkan delay ada respon server jika kondisi tertentu terpenuhi, seperti SLEEP() atau WAITFOR DELAY. Dengan memanfaatkan waktu respon server, penyerang dapat mengidentifikasi data tertentu atau struktur kueri. Penyerang mengirimkan input seperti ' OR SLEEP(5) --, yang menyebabkan server menunggu selama beberapa detik sebelum merespons. Sistem deteksi akan dilatih untuk mengenali pola yang menunjukkan adanya perintah yang berhubungan dengan penundaan waktu, seperti penggunaan kata kunci SLEEP, WAITFOR DELAY, atau parameter waktu lainnya.

f) Out-of-Band SQL Injection

Jenis SQL Injection ini terjadi ketika penyerang menggunakan teknik yang menyebabkan server atau aplikasi web membuat permintaan ke server atau sistem lain untuk mengambil informasi yang diinginkan. Teknik ini dapat digunakan ketika aplikasi web tidak memberikan umpan balik langsung kepada penyerang, tetapi mengeksekusi instruksi untuk mengirimkan data melalui protokol lain. Penyerang bisa memasukkan kueri yang mengarahkan aplikasi untuk membuat permintaan HTTP atau DNS ke server tertentu, seperti ' OR 1=1; SELECT * FROM users INTO OUTFILE '/var/www/html/file.txt'. Deteksi jenis ini memerlukan pemantauan terhadap komunikasi aplikasi yang keluar ke sistem lain dan mengidentifikasi kueri yang menyebabkan aplikasi mengirimkan data ke luar jaringan.

Untuk memberikan gambaran mengenai karakteristik data yang digunakan dalam penelitian ini, Tabel 3.1 menyajikan contoh kueri SQL yang terdapat dalam dataset yang berisi kueri normal maupun kueri yang mengandung SQL Injection. Setiap kueri telah diberi label, dimana label 1 menunjukkan kueri berbahaya (SQL Injection) dan label 0 menunjukkan kueri normal.

Tabel 3.1 Dataset

Sentence	Label
a',"	1
""a' or 1 = 1; --"","	1
@',"	1
""x' or full_name like '%bob%"","	1
23 or 1 = 1; --',"	1
select * from users where id = 1 or (	1
) or pg_sleep (__TIME__) --,1	1
SELECT * FROM users WHERE id = 1;	0
SELECT username, password FROM users WHERE username = 'admin';	0
SELECT * FROM products WHERE category = 'electronics';	0
SELECT email FROM customers WHERE customer_id = 25;	0

Sentence	Label
administrative budget , " said Rothmund "	0
select * from users where id = 1 or ""	0

2) Data *Preprocessing*

Sebelum masuk proses pemodelan, terdapat tahap preprocessing data. Preprocessing dilakukan untuk memastikan bahwa kualitas dataset telah terpenuhi dan siap memasuki tahap selanjutnya. Proses preprocessing data diawali dengan pengambilan dataset dan diunduh secara langsung dari reposiotri github melalui protokol HTTP, lalu disimpan dalam format CSV. Selanjutnya dataset dibaca oleh lib pandas dengan pengaturan encoding UTF-16 dengan mengabaikan baris yang tidak valid melalui parameter on_bad_lines='skip'. Hasilnya apabila struktur kolom pada dataset ditemukan tidak sesai, penyesuaikan akan dilakukan dnegan mengatur ulang header kolom menjadi sentence dan label

Tahap selanjutnya masuk kedalam pembersihan data atau data cleaning bertujuan meningkatkan kualitas dataset dengan dilakukannya penghapusan entri yang tidak valid. Pada tahap ini kueri yang kosong atau bernilai null atau hanya berisi teks tidak bermakna seperti “nan” dan “none” akan dihapus dari dataset. Permbersihan teks (text cleaning) dilakukan pada setiap kueri SQL yang didalamnya terdapat kolom sentence. Proses ini termasuk pengubahan seluruh karakter menjadi huruf kecil (lowercasing), penghapusan spasi ganda, penghapusan spasi ganda pada bagian awal dan akhir teks. Langkah-langkah dilakukan untuk meminimalkan variasi teks yan tidak relevan tanpa menghilangan karakter penting yang merepresentasikan struktur sintaks SQL.

Setelah data dibersihkan, pertama kali saya lakukan analisis distribusi label untuk mengetahui proporsi dari query normal dan query dengan SQL injection. Dari 33.757 query yang ada di dataset, terdapat 22.304 query yang dilabeli 0 dan 11.453 query dilabeli 1 dikategorikan sebagai SQL injection. Kemudian proses tokenisasi dilakukan dengan

merubah query SQL menjadi representasi angka dengan menggunakan Tokenizer yang ada di Keras. Tokenisasi ini dilakukan dengan batas maksimal 5.000 kata yang berbeda dan dengan penambahan token khusus OOV (out-of-vocabulary) untuk mengakomodasi kata yang tidak ada di vocab. Setiap kata yang ada dikonversi menjadi indeks angka sesuai frekuensi kemunculannya dalam dataset. Token yang dihasilkan dalam bentuk array yang berisi 0 semua kemudian array tersebut dihapus untuk memastikan semua samples data sudah memiliki representasi angka yang valid.

Pada tahap terakhir preprocessing, dataset yang telah diproses dibagi menjadi dataset pelatihan dan pengujian dengan rasio 80:20 menggunakan metode train-test split. Pembagian ini dilakukan untuk mengevaluasi kinerja model secara objektif menggunakan data yang tidak akan digunakan selama pelatihan. Melalui proses ini, dataset yang bersih, terstruktur, dan siap digunakan dalam tahap pemodelan dicapai. Data yang telah dibersihkan, dianalisis untuk distribusi label, tokenized, dan dipadatkan menjadi dasar bagi model TextCNN untuk mengenali pola yang menunjukkan serangan SQL Injection.

3) Modelling

Arsitektur TextCNN dibangun untuk memenuhi kebutuhan tahap tertentu dalam pengembangan model, yang berfokus pada klasifikasi kueri SQL menjadi kueri normal dan kueri serangan SQL injection. Arsitektur ini dirancang untuk model neural konvolusional untuk mengekstrak pola lokal dalam data teks berurutan menggunakan pendekatan Conv1D. Selama fase awal, beberapa parameter kunci awal ditetapkan, termasuk vocab_size sebesar 5.000, embedding_dim sebesar 128, dan maxlen sebesar 43 token. Model ini dirancang untuk klasifikasi biner, sehingga terdapat satu neuron di lapisan keluaran (num_classes = 1). Untuk mengatasi masalah ketidakseimbangan dalam jumlah sampel untuk kelas normal dan kelas serangan, bobot kelas diterapkan menggunakan metode

seimbang. Bobot kelas dihitung berdasarkan distribusi label untuk mengurangi bias terhadap kelas mayoritas.

Urutan token dengan panjang tetap berfungsi sebagai input model. Setiap token dipetakan ke lapisan embedding dengan dimensi 128. Ini membuat representasi semantik kata-kata dipelajari secara otomatis selama pelatihan. Untuk menangkap pola lokal dari kueri SQL, beberapa lapisan Conv1D diterapkan secara paralel dengan ukuran kernel 3, 4, dan 5. Alasan untuk memvariasikan ukuran kernel adalah untuk memungkinkan ekstraksi n-gram dari panjang yang berbeda untuk lebih baik mengenali pola serangan SQL Injection yang berbeda.

Proses pooling dilanjutkan dengan penggabungan seluruh fitur menggunakan lapisan concatenate. Dominasi fungsi pooling sebelumnya diekstrak dengan lapisan Global Max Pooling untuk setiap lapisan. Representasi fitur yang lebih kaya dihasilkan dari penggabungan seluruh bagian lapisan pooling. Di sini nilai Dropout 0.5 digunakan untuk memotong pengurangan risiko dari lapisan tersebut. Representasi fitur ini lalu diproses oleh lapisan Dense dengan 64 neuron, fungsi aktivasi dari setiap neuron di dalamnya menggunakan ReLU, untuk mempelajari pola di tingkat yang lebih tinggi sebelum klasifikasi terakhir. Fungsi aktivasi sigmoid digunakan dalam neuron tunggal klasifikasi output untuk mendapatkan probabilitas dari klasifikasi yang bersifat biner. Model di-compile menggunakan binary cross-entropy untuk loss function-nya, Adam untuk optimizer, dan akurasi sebagai metric evaluasi. Selama 10 epoch dengan batch size 128, 20% dari data dipakai sebagai data validasi. Selama periode ini model mempelajari perbedaan antara query normal dan query attack.

Setiap model yang telah selesai dilatih, akan digunakan untuk memprediksi pada data testing. Model akan memprediksi target variabel dalam bentuk probabilitas pertama, selanjutnya probabilitas tersebut dikategorikan ke dalam kelas biner dengan cutoff 0,5. Evaluasi model akan

menggunakan laporan klasifikasi yang berisi metrik presisi, recall, F1-score, serta akurasi.

4) Train Model

Setelah melewati tahap pemrosesan data, data latih tersebut digunakan sebagai input untuk keperluan pengujian model Text Convolutional Neural Network (TextCNN) dalam pelatihan. Optimizer Adam digunakan dalam pelatihan dengan binary cross-entropy sebagai fungsi kerugian, epoch 10, dan ukuran batch 128. 20% dari total data latih digunakan sebagai data validasi untuk mengevaluasi kemampuan model. Selain itu, weight class yang digunakan untuk menyeimbangkan hubungan antar kelas pada dataset, agar pola normal query dan query serangan SQL Injection dipelajari lebih proporsional. Dalam pelatihan model yang sedang berlangsung, nilai akurasi dan loss dari data latih dan validasi digunakan sebagai tolak ukur untuk melaporkan konvergensi dan untuk mengetahui jika terjadi overfitting. Pengaturan pelatihan ini dilakukan untuk meningkatkan kemampuan model pada generalisasi agar dapat melakukan identifikasi serangan SQL Injection dengan lebih akurat dan dengan stabil.

5) Model Deployment

Tahap AI Model Deployment merupakan proses akhir dalam metodologi penelitian yang bertujuan untuk mengintegrasikan model TextCNN yang telah melalui proses pelatihan dan evaluasi ke dalam sistem aplikasi web yang dapat diakses oleh pengguna. Pada tahap ini, model yang telah mencapai performa optimal disimpan dalam format Keras (.h5) yang mencakup arsitektur jaringan saraf dan bobot hasil pelatihan. Penyimpanan model dalam format ini memungkinkan penggunaan kembali secara efisien tanpa memerlukan proses pelatihan ulang. Selanjutnya, model diimplementasikan ke dalam lingkungan aplikasi web sebagai modul deteksi SQL Injection, sehingga sistem mampu melakukan proses inferensi terhadap kueri SQL yang dikirimkan oleh pengguna. Melalui proses ini, model dapat mendeteksi potensi

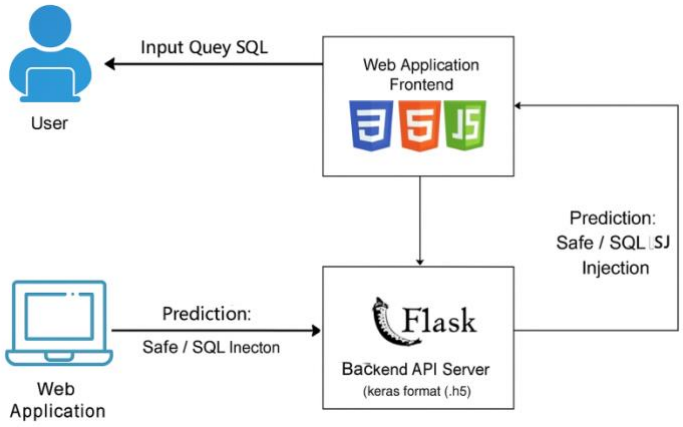
serangan SQL Injection secara otomatis dan mendukung mekanisme pengamanan aplikasi web secara real-time. Tahap deployment ini berfungsi sebagai penghubung antara hasil pengembangan model berbasis deep learning dan penerapannya dalam skenario penggunaan nyata.

3.4.2 Metode Pengembangan Aplikasi

Perancangan dan pengembangan aplikasi yang dilakukan dalam penelitian sistem deteksi SQL Injection emnggunakan TextCNN yang berbabasis web bertujuan untuk memastikan seluruh komponen dalam websitebekerja sesuai kebutuhan sistem. Proses ini mencakup pembuatan desain sistem menggunakan diagram arsitektur, Use Case Diagram, Flowchart Aplikasi, dan Desain sistem antarmuka pengguna.

3.4.2.1 Pengembangan Aplikasi

Perancangan diagram arsitektur bertujuan sebagai alat visual guna memetakan struktur komponen dan hubungan antar bagian dalam suatu sistem. Diagram ini dapat memudahkan dalam penelitian untuk memproses perencanaan, komunikasi, pengambilan keputusan, serta dokumentasi dalam proses pengembangan dan pemeliharaan sistem secara efisien. Gambar 3.5 menunjukan rancangan arsitektur aplikasi deteksi gambar asli berbasis website yang dikembangkan dalam peneltian ini.



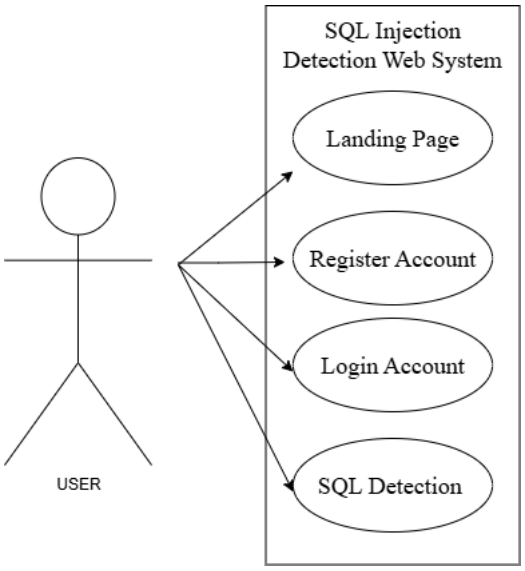
Gambar 3.5 Rancangan Arsitektur Sistem

Deteksi SQL Injection berbasis web yang terdiri dari empat komponen utama, yaitu front-end, back-end, model TextCNN (.h5), dan API deteksi. Pada bagian front-end, pengguna berinteraksi melalui halaman web yang dirancang menggunakan HTML, CSS, dan JavaScript. Melalui antarmuka ini, pengguna dapat memasukkan kueri SQL yang akan diperiksa keamanannya. Selanjutnya, bagian back-end dibangun menggunakan framework Flask (Python) yang berfungsi sebagai penghubung antara antarmuka pengguna dan model deteksi. Flask bertanggung jawab untuk menerima input kueri dari pengguna, mengirimkannya ke model TextCNN, serta menampilkan hasil prediksi kepada pengguna.

Model TextCNN (.h5) yang telah dilatih sebelumnya berperan sebagai inti sistem yang melakukan analisis terhadap pola teks kueri untuk mendeteksi adanya indikasi serangan SQL Injection. Model ini diintegrasikan melalui API deteksi yang memungkinkan komunikasi antara aplikasi web dan model pembelajaran mesin secara real-time. Dengan arsitektur ini, sistem dapat memproses input kueri pengguna, menjalankan prediksi, dan menampilkan hasil klasifikasi apakah kueri tersebut aman atau mengandung ancaman SQL Injection, semuanya melalui tampilan web yang interaktif dan responsif.

3.4.2.2 Use Case Diagram Aplikasi

Use case diagram merupakan salah satu jenis diagram dalam Unified Modeling Language (UML) yang digunakan untuk menggambarkan hubungan antara pengguna (user) dengan sistem dalam mencapai tujuan tertentu. Diagram ini menyoroti fungsi-fungsi utama yang dapat diakses oleh pengguna dari sudut pandang interaksi nyata dengan sistem.



Gambar 3. 3 Use Case Diagarm sistem

Berikut penjelasan untuk masing-masing use case pada sistem deteksi SQL Injectiom berbasis web :

1. Landing Page

Halaman awal yang akan ditampilkan kepada pengguna saat pertama kali mengakses sistem. Pada halaman ini terdapat informasi singkat mengenai fungsi dan tujuan sistem deteksi SQL Injection, dua tombol utama Login dan Register akan mengarahkan pengguna untuk masuk atau membuat akun baru sebelum menggunakan fitur utama deteksi.

2. Register Account

Fitur ini memungkinkan pengguna baru untuk membuat akun dengan mengisi data seperti username dan password. Data pengguna kemudian disimpan secara aman di dalam database (misalnya MySQL). Tujuan use case ini adalah untuk mengontrol akses sistem dan melacak aktivitas pengguna secara individual.

3. Login Account

Setelah terdaftar, pengguna dapat mengakses sistem dengan memasukkan username dan password yang valid. Sistem akan memverifikasi kredensial tersebut terhadap data yang tersimpan di basis data. Jika valid, pengguna diarahkan menuju Dashboard Deteksi SQL

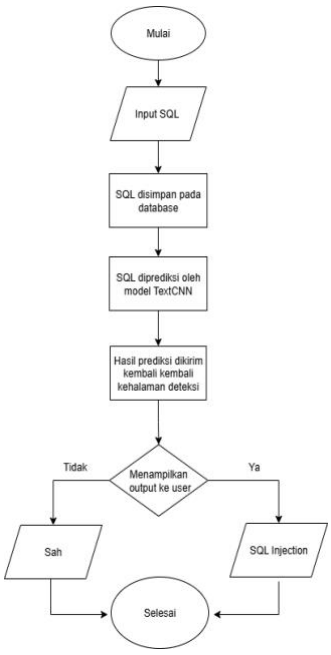
Injection. Proses ini menjamin keamanan akses dan mencegah penyalahgunaan sistem.

4. Halaman Deteksi SQL Injection

Merupakan inti dari sistem, di mana pengguna dapat memasukkan perintah atau kueri SQL ke dalam kolom input yang tersedia. Setelah pengguna menekan tombol Scan, sistem akan mengirim kueri tersebut ke model TextCNN melalui API Flask untuk dianalisis. Hasil prediksi akan ditampilkan di layar, menunjukkan apakah kueri tersebut tergolong aman atau mengandung potensi serangan SQL Injection, beserta nilai probabilitas kepercayaannya.

3.4.2.3 Flowchart Aplikasi

Flowchart adalah bentuk visualisasi dari alur proses atau lagoritma yang ditunjukkan dengan simbol-simbol untuk merepresentasikan dari tahapan awal, proses, hinggal pengambilan keputusan dan akhir alur. Dengan adanya flowchart membantu memperjelas dan menyusun setiap langkah secara runtut dan sistematis, sehingga alur keseluruhan proses menjadi lebih mudah dipahami.

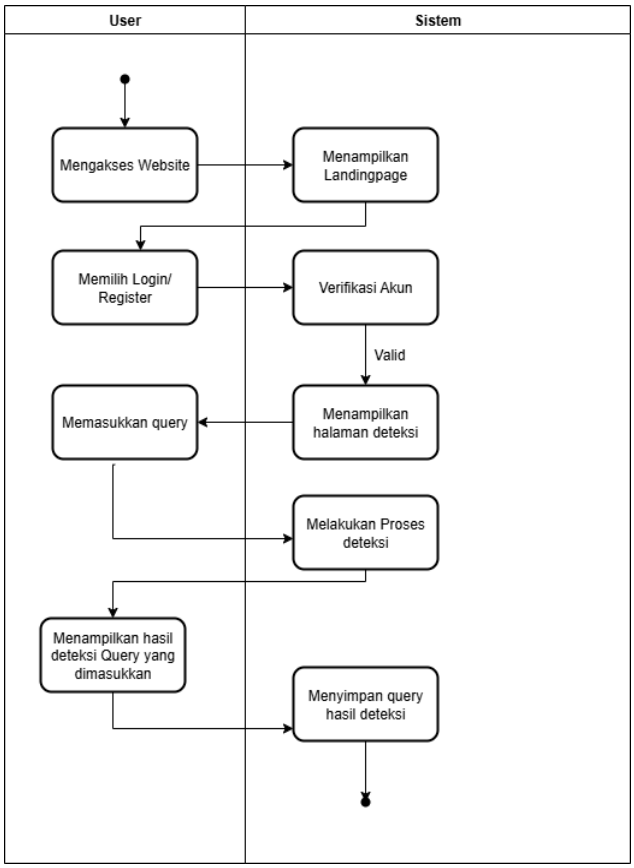


Gambar 3. 4 Flowchart Sistem

3.4.2.4 Activity Diagram Aplikasi

Activity Diagram adalah salah satu jenis diagram dalam Unified Modeling Language (UML) yang berfungsi untuk menggambarkan alur aktivitas atau proses yang terjadi dalam suatu sistem. Pada penelitian ini, activity diagram digunakan untuk memvisualisasikan urutan proses yang terjadi dalam sistem deteksi serangan SQL Injection berbasis TextCNN

melalui platform web. Diagram ini menjelaskan secara sistematis bagaimana interaksi antara pengguna dan sistem berlangsung, mulai dari pengguna mengakses halaman utama hingga sistem menampilkan hasil deteksi potensi serangan SQL Injection



Gambar 3. 5 Activity Diagram Sistem

Proses dimulai ketika pengguna membuka website utama (landing page), dimana pengguna dapat memilih untuk login atau mendaftar akun

baru (register). Jika pengguna belum memiliki akun, sistem akan mengarahkan ke halaman registrasi untuk mengisi data seperti username dan password, kemudian data tersebut disimpan secara aman ke dalam basis data MySQL. Setelah berhasil mendaftar atau login, pengguna akan diarahkan ke halaman dashboard deteksi SQLi.

Di halaman deteksi ini, pengguna memasukkan perintah SQL (kueri) ke dalam kolom input yang telah disediakan. Sistem kemudian melakukan validasi format input, untuk memastikan bahwa perintah SQL sesuai dengan sintaks yang dapat dianalisis oleh model. Setelah pengguna menekan tombol Scan, sistem akan mengirimkan kueri tersebut ke model deteksi berbasis TextCNN melalui API Flask. Model akan memproses input dan menghasilkan prediksi berupa label "Aman" atau "SQLi" beserta nilai probabilitas keyakinan model. Jika hasil prediksi menunjukkan adanya potensi SQL Injection, sistem menampilkan pesan peringatan serta tingkat probabilitas ancaman. Sebaliknya, jika kueri dinilai aman, sistem akan menampilkan hasil bahwa kueri tersebut tidak mengandung indikasi serangan SQLi. Proses ini kemudian berakhir dengan pengguna yang dapat melakukan deteksi ulang terhadap kueri lain atau keluar dari sistem (logout) untuk mengakhiri sesi.

Dengan adanya activity diagram ini, alur kerja sistem deteksi SQL Injection menjadi lebih mudah dipahami karena setiap langkah digambarkan secara terstruktur, memperlihatkan hubungan antar aktivitas dari sisi pengguna hingga proses analisis internal oleh model berbasis TextCNN.

3.4.2.5 Desain Antarmuka Pengguna

Desain antarmuka pengguna (User Interface) pada sistem deteksi SQL Injection berbasis web ini dirancang dengan fokus pada kesederhanaan, kemudahan interaksi, dan kejelasan tampilan informasi. Pendekatan ini bertujuan agar pengguna, baik dari kalangan teknis seperti pengembang web maupun pengguna umum, dapat dengan mudah

memahami cara penggunaan sistem serta memperoleh hasil deteksi secara cepat dan akurat. Antarmuka dibuat agar intuitif, dengan struktur navigasi yang jelas mulai dari dashboard deteksi, hingga tampilan hasil analisis.

Gambar 3. 6 Dashboard

Pada gambar 3.9 Desain antarmuka dashboard deteksi SQL Injection dibuat sederhana dan mudah digunakan agar pengguna dapat langsung memahami fungsinya. Pada halaman ini terdapat judul “Masukkan Kueri yang Ingin Dideteksi”, diikuti dengan kolom input untuk mengetik atau menempelkan perintah SQL. Setelah itu, pengguna dapat menekan tombol “Deteksi Sekarang” untuk memulai analisis menggunakan model TextCNN.

3.5. Pengujian Sistem

Tahap pengujian dilakukan setelah seluruh proses perancangan dan pengembangan sistem diselesaikan. Pada tahap ini, evaluasi terhadap kinerja dan efektivitas sistem secara menyeluruh dilakukan untuk memastikan bahwa hasil implementasi telah sesuai dengan tujuan serta spesifikasi penelitian yang ditetapkan. Peran penting pengujian ditujukan untuk menilai sejauh mana sistem dapat dioperasikan dengan baik serta mampu menghasilkan keluaran sesuai dengan yang diharapkan. Proses pengujian diterapkan pada beberapa komponen utama, yang mencakup pengujian model untuk mengukur kinerja algoritma pendeteksian serangan SQL Injection, serta pengujian fungsionalitas

website guna memastikan seluruh fitur dan alur interaksi pengguna dapat dijalankan secara stabil, akurat, dan sesuai dengan rancangan sistem.

3.5.1 Evaluasi Model

Tahapan evaluasi dalam penelitian ini dilakukan berdasarkan metode pengembangan model AI Life Cycle pada bagian deployment. Pada tahap ini, evaluasi model diterapkan untuk menilai kemampuan model TextCNN dalam mengklasifikasikan kueri SQL normal dan kueri yang mengandung serangan SQL Injection (SQLi) berdasarkan data uji yang telah disiapkan. Confusion matrix digunakan sebagai tabel yang menggambarkan perbandingan antara label hasil prediksi model dan label sebenarnya. Melalui tabel tersebut, jumlah prediksi yang benar maupun yang salah pada masing-masing kelas dapat diketahui, sehingga analisis yang lebih mendalam terhadap kesalahan klasifikasi dapat dilakukan (Krstinić dkk., 2020). Berdasarkan informasi yang diperoleh dari confusion matrix, berbagai metrik evaluasi selanjutnya dapat dihitung untuk menilai efektivitas model secara keseluruhan...

1) *Accuracy* digunakan untuk menunjukkan persentase prediksi yang dilakukan secara benar terhadap keseluruhan data uji. Dalam metrik ini, jumlah prediksi positif dan negatif yang sesuai dengan label sebenarnya diperhitungkan. Persamaan yang digunakan untuk menghitung nilai accuracy ditunjukkan sebagai berikut:

Persamaan Menghitung Accuracy (1)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Keterangan :

A. *True Positive* (TP): Jumlah prediksi benar untuk kelas positif.
B. *True Negative* (TN): Jumlah prediksi benar untuk kelas negatif .
C. *False Positive* (FP): Jumlah prediksi salah untuk kelas positif.
D. *False Negative* (FN): Jumlah prediksi salah untuk kelas negative.

True Positive (TP) didefinisikan sebagai jumlah kueri serangan SQL Injection yang berhasil dideteksi dengan benar oleh sistem, sedangkan *True*

Negative (TN) merujuk pada jumlah kueri normal yang diklasifikasikan secara tepat. Kondisi *False Positive* (FP) terjadi ketika kueri normal diklasifikasikan sebagai serangan, yang berpotensi menimbulkan gangguan pada layanan aplikasi web. Sementara itu, *False Negative* (FN) menggambarkan keadaan ketika serangan *SQL Injection* tidak berhasil terdeteksi oleh sistem, sehingga risiko keamanan dapat muncul. Kinerja sistem deteksi *SQL Injection* yang diusulkan dinilai baik apabila nilai TP dan TN diperoleh dalam jumlah tinggi, serta nilai FP dan FN dapat ditekan serendah mungkin.

2) Precision

Precision digunakan untuk mengukur tingkat ketepatan model dalam memprediksi kueri berbahaya atau SQL Injection. Nilai precision yang tinggi menunjukkan bahwa kesalahan klasifikasi kueri normal sebagai serangan jarang terjadi. Persamaan untuk menghitung precision ditunjukkan sebagai berikut:

Persamaan Menghitung Precision (2)

$$Precision = \frac{TP}{TP + FP}$$

3) Recall

Recall digunakan untuk menunjukkan sejauh mana seluruh kueri yang benar-benar berbahaya dapat dideteksi oleh model. Semakin tinggi nilai recall yang diperoleh, semakin sedikit serangan yang tidak terdeteksi. Persamaan recall dinyatakan sebagai berikut:

Persamaan Menghitung Recall (3)

$$Recall = \frac{TP}{TP + FN}$$

4) F1-*score*

F1-*Score* didefinisikan sebagai rata-rata harmonis antara precision dan recall. Metrik ini digunakan untuk menilai keseimbangan antara kemampuan model dalam mendeteksi serangan (recall) dan kemampuan menghindari kesalahan deteksi (precision), khususnya pada kondisi dataset yang tidak

seimbang antara kueri normal dan kueri SQL Injection. Persamaan F1-Score ditunjukkan sebagai berikut:

Persamaan Menghitung F1-Score (4)

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Proses pengujian dilakukan dengan menggunakan data uji yang dipisahkan dari data latih, di mana kumpulan kueri SQL diklasifikasikan ke dalam dua kelas, yaitu Normal dan *SQL Injection*. Setelah seluruh kueri diproses oleh model, hasil prediksi yang dihasilkan dibandingkan dengan label sebenarnya untuk membentuk sebuah confusion matrix. Berdasarkan tabel tersebut, nilai *accuracy, precision, recall*, dan F1-*score* kemudian dihitung sebagai indikator utama dalam menilai kinerja model.

Selain itu, hasil evaluasi dimanfaatkan untuk dilakukan analisis terhadap pola kesalahan klasifikasi yang muncul, seperti kecenderungan model dalam mengklasifikasikan kueri normal sebagai *SQL Injection* (*false positive*) maupun kegagalan dalam mendeteksi serangan (*false negative*). Analisis ini dijadikan dasar untuk dilakukan penyesuaian hyperparameter, penambahan variasi data, atau modifikasi arsitektur model guna meningkatkan performa deteksi.

Apabila hasil evaluasi menunjukkan bahwa deteksi serangan *SQL Injection* dapat dilakukan oleh model dengan nilai presisi dan recall yang tinggi, maka model tersebut dinyatakan layak untuk diintegrasikan ke dalam sistem pendeteksian serangan pada aplikasi web secara *real-time.*

3.5.2 Pengujian Fungsionalitas Website

Pengujian perangkat lunak dilakukan untuk mengidentifikasi kesalahan yang terdapat pada sistem sebelum kesalahan tersebut ditemukan oleh pengguna. Pada penelitian ini, tahap pengujian fungsionalitas sistem deteksi SQL Injection berbasis web dilakukan dengan menerapkan metode black-box

testing. Pengujian ini difokuskan pada pemeriksaan fungsi-fungsi utama aplikasi dari sisi antarmuka pengguna, tanpa melibatkan analisis terhadap struktur internal maupun kode program yang digunakan. Pendekatan tersebut diterapkan untuk menilai apakah setiap fitur sistem dapat dijalankan sesuai dengan kebutuhan serta spesifikasi yang telah ditetapkan. Tujuan utama dari pengujian ini adalah untuk memastikan bahwa respons sistem terhadap input kueri SQL telah diberikan dengan benar, serta keluaran deteksi yang dihasilkan sesuai dengan yang diharapkan, baik berupa status aman maupun peringatan adanya serangan.

Proses pengujian dilakukan dengan merancang sejumlah skenario penggunaan yang merepresentasikan kondisi nyata, seperti pengujian menggunakan kueri normal yang valid maupun kueri yang mengandung payload SQL Injection dengan tingkat kompleksitas yang beragam. Setiap skenario pengujian disusun dalam bentuk uji kasus yang mencakup langkah-langkah yang dilakukan oleh pengguna, keluaran yang diharapkan, serta hasil aktual yang ditampilkan oleh sistem. Fitur utama yang diuji meliputi proses input kueri oleh pengguna, mekanisme klasifikasi menggunakan model TextCNN, serta penyajian hasil deteksi pada antarmuka web. Seluruh hasil pengujian, baik yang berhasil maupun yang tidak berhasil, dicatat dan dievaluasi untuk menilai kesesuaian fungsi sistem dengan rancangan serta tujuan penelitian. Selanjutnya, format instrumen pengujian black-box yang digunakan disajikan sebagai acuan evaluasi

Tabel 3. 2 Instrumen Pengujian Fungsionalitas dengan metode blackbox

No	**Uji Kasus**	**Skenario**	**Hasil Yang Diharapkan**
1.	Menampilkan Login/Register	Pengguna memilih tombol login/register	Sistem mengarahkan kehalaman login/register

No	Uji Kasus	Skenario	Hasil Yang Diharapkan
2	Menekan tombol register	Pengguna menekan tombol register	sistem menampilkan kolom isi username dan password yang akan didaftarkan
3.	Menekan tombol login	Pengguna menekan tombol login	Sistem menampilkan kolom isi username dan password yang sudah terdaftar dan sistem mengarahkan kehalaman deteksi
4	Menekan tombol “deteksi kueri”	pengguna menekan tombol “deteksi kueri”	sistem menampilkan hasil deteksi kueri yang dideteksi

3.6 Evaluasi Sistem

Evaluasi sistem bertujuan untuk menilai kinerja akurasi dan efektifitas aplikasi web deteksi SQL Injection yang dikembangkan. Evalusasi difokuskan pada performa model deteksi berbasis TextCNN yang dengan metrik akurasinya. sedangkan performa aplikasi web deteksi SQL Injection mencakup aspek fungsionalitas dan kemudahan pengguna dari sistem yang diimplementasikan melalui framework Flask. Pengujian diterapkan dengan memanfaatkan dataset uji yang dipisahkan dari data pelatihan untuk menilai kemampuan generalisasi model dalam mendeteksi kueri berbahaya

Confussion matrix digunakan sebagai tolak ukur ketepatan performa model dalam mengklasifikasikan kueri kedalam dua kategori yaitu, aman atau mengandung potensi serangan. Pada pengujian aplikasi web, metode blackbox diterapkan sebagai pendekatan evaluasi sistem.

3.7 Penyampaian Hasil Pengujian

Setelah menyelesaikan semua tahapan dari mulai perancangan hingga evaluasi sistem, berikutnya adalah penyajian hasil penelitian. Pada tahap ini, Penelitian ini disusun sebagai skripsi yang mencakup seluruh proses dan temuan penelitian, kemudian direpresentasikan disidang skrisi didepan dosen penguji. Dengan melalui tahapan tersebut, hasil penelitian dinilai secara ilmiah dan sistematis, sehingga kontribusi pengetahuan dapat diberikan terhadap perkembangan ilmu keamanan siber serta penerapan deep learning dalam mendeteksi serangan SQL Injection.