SKRIPSI

APLIKASI DETEKSI SIMILARITAS JAVASCRIPT CODE BERBASIS MERN MENGGUNAKAN ALGORITMA RABIN KARP

Disusun dan diajukan oleh:

MUHAMMAD NUR FAISI S
D121 18 1503

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2023

LEMBAR PENGESAHAN SKRIPSI

APLIKASI DETEKSI SIMILARITAS JAVASCRIPT CODE BERBASIS MERN MENGGUNAKAN ALGORITMA RABIN KARP

Disusun dan diajukan oleh

MUHAMMAD NUR FAISI S
D121181503

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian
Studi Program Sarjana Program Studi Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 17 Februari 2023
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Eng. Ir. Muhammad Niswar., S.T., M.IT.	Iqra Aswad S.T., M.IT.
NIP. 19730922 199903 1 001	NIP. 19901128 201904 3 001

Ketua Program Studi,

Prof. Dr. Indrabayu, ST., MT., M.Bus.Sys., IPM., ASEAN Eng.
NIP 19750716 200212 1 004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini ;

Nama	:	Muhammad Nur Faisi S
NIM	:	D121181503
Program Studi	:	Teknik Informatika
Jenjang	:	S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

Aplikasi Deteksi Similaritas Javascript Code Berbasis MERN Menggunakan Algoritma Rabin Karp

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 27 Februari 2023

Yang Menyatakan

Muhammad Nur Faisi S

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT atas rahmat dan berkat-Nya, sehingga dapat menyelesaikan tugas akhir skripsi yang berjudul "**Aplikasi Deteksi Similaritas Javascript Code Berbasis MERN Menggunakan Algoritma Rabin Karp"** sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 di Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin.

Penulis menyadari banyak kesulitan dan kendala yang dihadapi saat penyusunan tugas akhir ini. Dalam prosesnya, penulis memperoleh banyak bantuan, dukungan, dan bimbingan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan terima kasih kepada:

1. Allah SWT melalui berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Kedua orang tua penulis, Ayah Syarifuddin Ganti dan Ibu Suarna Jaelani yang selalu menyertai penulis dalam doanya serta mendukung, membantu, memberi semangat serta kasih sayang dalam perjalanan penulis menyelesaikan tugas akhir ini.
3. Bapak Dr. Eng. Ir. Muhammad Niswar., S.T M.IT. selaku pembimbing I dan Bapak Iqra Aswad S.T M.T. selaku pembimbing II, yang senantiasa menyediakan waktu, tenaga, pikiran, dan perhatian yang luar biasa dalam mengarahkan penulis untuk menyelesaikan tugas akhir.
4. Segenap Dosen dan Staff Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah banyak membantu penulis selama masa perkuliahan.
5. Teman-teman Teknik Informatika Angkatan 2018 selaku rekan yang telah memberi bantuan, dukungan dan semangat selama masa perkuliahan dan penyusunan tugas akhir ini.
6. Serta berbagai pihak atas segala dukungan dan bantuannya yang tidak dapat penulis tuliskan satu persatu.

Penulis berharap semoga Allah SWT berkenan membalas segala kebaikan yang telah diterima oleh penulis dari berbagai pihak yang telah membantu mempermudah penulis dalam mengerjakan tugas akhir ini. Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna, oleh karena itu penulis mengharapkan segala bentuk saran serta masukan yang membangun dari berbagai pihak. Semoga tugas akhir ini dapat memberikan pengetahuan dan manfaat bagi penulis dan pembaca.

Makassar, Oktober 2022

Penulis,

Muhammad Nur Faisi S

ABSTRAK

Muhammad Nur Faisi S. *Aplikasi Deteksi Similaritas Javascript Code Berbasis MERN Menggunakan Algoritma Rabin Karp* (dibimbing oleh Dr. Eng. Ir. Muhammad Niswar., S.T M.IT dan Iqra Aswad S.T M.T)

Tugas pemrograman merupakan standarisasi yang dibuat untuk mengukur keberhasilan dan tingkat pemahaman mahasiswa dari sebuah kompetensi dalam proses belajar. Hasil dari tugas yang dikerjakan dapat menjadi penilaian terhadap pemahaman mahasiswa akan suatu materi. Saat ini pengumpulan tugas mahasiswa dapat dilakukan secara *online* melalui LMS kampus. Namun, hal ini menjadi peluang bagi mahasiswa untuk melakukan tindak kecurangan yaitu menontek karena pengumpulan tugas yang dilakukan secara *online*. Saat ini belum ada pengecekan similaritas kode pada tugas sehingga banyak tugas yang memiliki kemiripan tetapi tidak terdeteksi oleh LMS kampus. Jika dilakukan secara *manual* juga akan sangat memakan waktu bagi dosen untuk memeriksa satu per satu tugas mahasiswa. Belum lagi jika terdapat banyak mahasiswa dalam satu kelas. Dengan adanya sistem pendeteksi similaritas, hal tersebut dapat dicegah karena sistem dapat mendeteksi similaritas kode antar mahasiswa baik itu dalam satu kelas maupun di kelas yang beda terhadap tugas pemrograman yang dikerjakan oleh mahasiswa. Dengan menerapkan algoritma *Rabin-Karp*, maka dapat dibuat sistem yang dapat mendeteksi similaritas kode pada tugas – tugas yang diberikan . Sistem ini juga dilengkapi dengan *text editor* sehingga mahasiswa tidak perlu mengerjakan tugasnya di aplikasi *third-party* untuk mengeksekusi kode dan juga terdapat *unit testing* yang dapat diterapkan oleh dosen supaya kode yang dikumpulkan bisa dicek terlebih dahulu sebelum mahasiswa dapat mengumpulkannya. Adapun data yang digunakan adalah tugas mahasiswa yang telah menyelesaikan tugasnya pada sistem dan disimpan di dalam database. Proses pengecekan tugas mahasiswa menggunakan *Esprima* yang berfungsi untuk melakukan *tokenizing*. Hasil dari proses tokenizing kemudian akan dicek kesamaannya menggunakan algoritma *Rabin Karp.* Sistem berhasil melakukan perhitungan similaritas kode terhadap tugas mahasiswa dan menghasilkan persentase kesamaan tugas untuk tiap – tiap tugas yang telah dikumpulkan.

Kata kunci: tugas pemrograman, similaritas kode, *esprima*, *Rabin-Karp*

ABSTRACT

Muhammad Nur Faisi S. *A MERN-Based Javascript Code Similarity Detection Application Using the Rabin-Karp Algorithm. (supervised by Dr. Eng. Ir. Muhammad Niswar., S.T M.IT and Iqra Aswad S.T M.T)*

Programming assignments are standardized to measure students' success and level of understanding of a competence in the learning process. The results of the assignments can be used to evaluate students' understanding of a subject. Currently, students can submit their assignments online through the university's LMS. However, this becomes an opportunity for students to cheat by copying because the assignments are submitted online. Currently, there is no code similarity check on assignments, so many assignments are similar but not detected by the university's LMS. If done manually, it will also take a lot of time for the instructor to check each student's assignment one by one. Not to mention if there are many students in one class. With the availability of a similarity detection system, this can be prevented because the system can detect code similarities between students, both within one class and in different classes, on programming assignments. By applying the Rabin-Karp algorithm, a system can be created that can detect code similarities on the given assignments. The system is also equipped with a text editor, so students do not have to work on their assignments in a third-party application to execute the code, and there is a unit testing that can be applied by the instructor so that the code can be checked first before students can submit it. The data used is the students' assignments that have completed the assignments on the system and are stored in the database. The process of checking student assignments uses Esprima, which functions to perform tokenizing. The result of the tokenizing process will then be checked for similarity using the Rabin Karp algorithm. The system successfully calculates the code similarity to the students' assignments and generates the percentage of assignment similarity for each submitted assignment.

Keywords: programming assignments, code similarity, esprima, Rabin-Karp

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemrograman komputer menjadi hal yang dasar dan harus diketahui oleh setiap mahasiswa yang sedang menjalani studi di Teknik Informatika. Mata kuliah yang ada di Teknik Informatika juga Sebagian besar berhubungan dengan Pemrograman Komputer sehingga tugas yang akan dikerjakan berhubungan dengan membuat sebuah kode yang digunakan untuk menyelesaikan sebuah masalah. Tugas yang dikumpul berupa *source code* yang menjadi tolak ukur penilaian bagi dosen.

Sebagai mahasiswa Teknik Informatika terutama untuk mahasiswa di semester awal, tugas untuk melakukan pemrograman akan sangat terasa berat karena sangat asing bagi mahasiswa sehingga mahasiswa bisa saja melakukan tindak plagiarisme karena *output* yang diinginkan dosen untuk seluruh mahasiswa pastinya akan sama dan tidak ada perbedaan antara mahasiswa dengan mahasiswa lain. Hal ini menjadi kelemahan bagi dosen yang memberikan tugas karena akan sulit untuk mendeteksi similaritas kode yang dilakukan oleh mahasiswa jika harus memeriksa tugas setiap mahasiswa satu per satu secara *manual* jika ingin mencari kesamaan tugas mahasiswa dengan mahasiswa lainnya sehingga jika terdapat banyak mahasiswa di kelas itu maka cara ini sangat tidak efisien.

Tindak plagiarisme masih sangat banyak terjadi di sekitar bahakn di lingkup kampus sendiri. Menurut penelitian yang dilakukan oleh Anies Makriftin (2022) berjudul Hubungan Antara Ketakutan Akan Kegagalan Dengan Perilaku Plagiarisme Mahasiswa Psikologi Universitas Mercu Buana Yogyakarta mengatakan bahwa seorang mahasiswa yang kurang memiliki keterampilan dalam mengerjakan tugas yang diberikan akan merasa tertekan dengan konsekuensi negatif seperti mendapat nilai yang rendah sehingga akan tergoda untuk melakukan tindak plagiarisme. Pada penelitian yang dilakukan oleh (Fadhillah, 2021) yang berjudul Pendeteksian *Source Code Plagiarism* Dengan Algoritma Rabin Karp p

Pada *Online Judge*, Fadhillah berhasil membuat sebuah sistem yang dapat mendeteksi plagiarisme terhadap *source code* dengan menerapkan algoritma *string matching*. Fadhillah dalam penelitian ini membandingkan beberapa algoritma dan memilih *Rabin Karp* berdasarkan hasil dari rata – rata kompleksitas dan ruang yang baik.

Algoritma Rabin Karp memang sangat sering digunakan dalam proses pendeteksian plagiarsme dalam beberapa kasus bukan hanya untuk plagiarisme pada *source code*. Pada penelitian yang dilakukan oleh Bahrul Khoir, dkk (2019) yang berjudul *Implementation of Rabin-Karp algorithm to determine the similarity of synoptic gospels,* Bahrul, dkk menggunakan algoritma Rabin Karp untuk menentukan Injil Sinoptik yang terdiri atas tiga Injil dalam Perjanjian Baru yaitu Matthew, Mark dan Luke dan berhasil memperkuat salah satu hipotesis yaitu fenomena prioritas Markus.

Dengan adanya system yang dapat digunakan oleh tenaga pengajar dalam mendeteksi plagiarisme, tenaga pengajar dapat menyeleksi mahasiswa – mahasiswa yang melakukan Tindakan plagiarisme. Output system yang nantinya berupa angka persentase dari kemiripan tugas antar mahasiswa dapat menjadi bahan acuan bagi pengajar seberapa paham mahasiswa tersebut dengan tugas yang diberikan.

Berdasarkan permasalahan tersebut, maka pada penelitian ini akan dibangun sebuah system yagn dapat digunakan oleh dosen untuk mengecek plagiarisme dari *source code* yang telah dikerjakan oleh mahasiswa dan juga system ini dapat digunakan oleh mahasiswa untuk mengerjakan tugas yang diberikan oleh dosen karena nantinya akan memiliki *code editor*sehingga mahasiswa langsunng mengerjakan pada sistem dan akan dikirim langsung ke dosen

1.2 Rumusan Masalah

1. Bagaimana membuat sebuah system yang dapat mendeteksi kesamaan program kode yang dibuat oleh antar mahasiswa?

2. Bagaimana hasil pengujian yang dilakukan menggunakan system yang telah dibuat?

1.3 Tujuan Penelitian

1. Menciptakan system yang bisa mendeteksi kesamaan program kode yang dibuat oleh mahasiswa
2. Menunjukkan bahwa system dapat menghasilkan pengujian yang dilakukan menggunakan system yang telah dibuat
3. Menciptakan kode yang *readable* dengan menerapkan *clean code* agar bisa diteruskan oleh developer lain.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat membantu:

1. Sistem yang dihasilkan membantu dosen menjadi lebih efisien dalam memeriksa tugas mahasiswa
2. Mahasiswa diharapkan menjadi lebih mudah mengerjakan tugas yang diberikan oleh dosen dalam pemrograman karena memiliki system sendiri

1.5 Ruang Lingkup

Sistem deteksi plagiarisme dilakukan pada tugas mahasiswa yang menggunakan Bahasa Pemrograman Javascript.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN berisi latar belakang, tujuan penelitian, manfaat, penelitian, batasan masalah dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA berisi landasan teori yang digunakan untuk menganalisis masalah yang akan diteliti dan digunakan untuk menyusun tugas akhir ini.

BAB III METODOLOGI PENELITIAN berisi tentang tahapan pada penelitian, teknik pengolahan data, metode penelitian, algoritma penelitian, teknik evaluasi hasil, dan implementasinya.

BAB IV HASIL DAN PEMBAHASAN berisi hasil dan pembahasan dari sistem yang telah dibangun.

BAB V PENUTUP berisi kesimpulan berdasarkan hasil penelitian dan saran untuk pengembangan penelitian selanjutnya

BAB II
TINJAUAN PUSTAKA

2.1 Plagiarisme

Plagiarisme adalah praktik penyalahgunaan hak kekayaan intelektual milik orang lain dan karya tersebut diakui tidak sah sebagai hasil karya pribadi. Menurut Sastroasmoro, klasifikasi plagiarisme berdasarkan proporsi atau presentasi kata, kalimat, atau paragraph yang dibajak, dibagi menjadi tiga, yaitu plagiarisme ringan: < 30%, plagiarisme sedang: 30% - 70%, dan plagiarisme berat: > 70% (Yulianto & Nurhasanah, 2021).

Banyak faktor yang menyebabkan terjadinya tindakan plagiarisme. Faktor penyebab plagiarisme sendiri adalah kemalasan mereka sendiri, karena mereka merasa stres, memiliki keyakinan bahwa perilaku tidak akan diketahui, dan perilaku tersebut tidak salah untuk dilakukan atau berbahaya (Yulianto & Nurhasanah, 2021).

2.2 String Matching

String Matching adalah proses pencarian semua kemunculan query yang selanjutnya disebut pattern ke dalam string yang lebih panjang (teks). Algoritma *string matching*adalah suatu metode yang digunakan untuk menemukan suatu keakuratan atau hasil dari satu atau beberapa pola teks yang diberikan. *String matching* merupakan pokok bahasan yang penting dalam ilmu komputer karena teks merupakan bentuk utama dari pertukaran informasi antar manusia, misalnya pada literatur, karya ilmiah, halaman web, dan sebagainya (Ginting & Utomo, 2019).

Algoritma *string matching* dapat diklasifikasikan menjadi tiga jenis sesuai dengan arah pencarian, yaitu (Charras & Lecroq, 2004)

a. Dari kiri ke kanan

   Dari arah yang paling natural yaitu kiri ke kanan yang merupakan arah membaca. Algoritma yang termasuk dalam kategori ini adalah algoritma Brute Force, algoritma Knuth Morris dan Pratt.

b. Dari kanan ke kiri

   Dari arah kanan ke kiri, arah yang biasanya menghasilkan hasil terbaik secara praktis. Algoritma yang termasuk dalam kategori ini adalah algoritma Boyer – Moore.

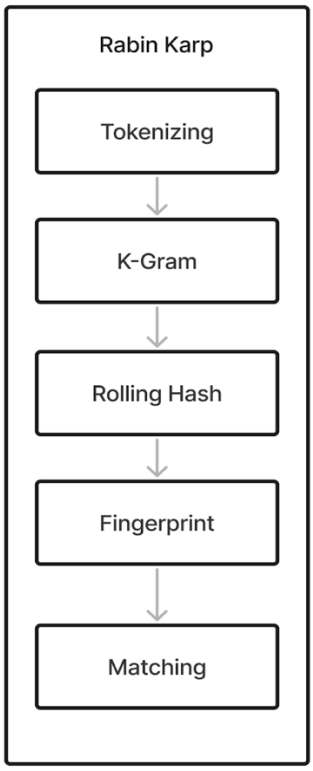
c. Dalam urutan tertentu

   Dari arah tertentu yang ditentukan oleh algoritma, arah ini secara teoritis menghasilkan hasil terbaik. Salah satu contoh algoritma dalam kategori ini adalah Colossi Crochemore Perrin.

2.3 Algoritma Rabin Karp

Algoritma Rabin-Karp adalah algoritma pencarian string yang paling sederhana. Algoritma ini menggunakan fungsi hash untuk menemukan pola potensial dalam teks input (Hidayat dkk., 2022). Algoritma Rabin-Karp adalah salah satu algoritma yang dapat digunakan untuk mencari dimana sebuah string (dalam kasus ini dinamakan sebagai pola) apakah ditemukan di dalam kumpulan string lain dengan ukuran yang lebih besar. Rabin-Karp menggunakan metode *hash* untuk mempercepat pencarian pola dalam kalimat. Metode ini mengubah setiap *string* menjadi angka yang merupakan nilai *hash.* Apabila 2 *string* adalah sama persis, maka nilai hashnya juga akan sama sehingga pencarian string dapat diturunkan dengan cara menghitung nilai *hash* dari pola dan kemudian melakukan pencarian pola dengan nilai *hash* yang sama pada data (Steveson dkk., 2018).

Adapun tahapan dari algoritma Rabin Karp adalah sebagai berikut :



Gambar 2.3.1 Algoritma Rabin Karp

Pada gambar 2.3.1 merupakan tahapan dari algoritma Rabin Karp. Algoritma Rabin Karp memiliki beberapa karakteristik yaitu menggunakan Hashing dan K-gram. Tahapan algoritma Rabin Karp dimulai dengan melakukan proses *tokenizing*. Proses *tokenizing* yang dilakukan dibantu dengan library *esprima* pada Javascript yang akan mendefinisikan kategori – kategori dari kode yang akan dijadikan sebagai data nantinya. Setelah itu, dilakukan proses K – gram. K-gram adalah rangkaian token yang panjangnya adalah *k*. Metode ini mengambil potongan – potongan karakter huruf sejumlah nilai *k* dari sebuah teks yang secara kontiunitas dibaca dari awal teks hingga akhir teks sumber (Karisma Wibowo & Hastuti, 2016).

Setelah itu, dilakukan proses hashing untuk mengubah karakter string menjadi integer yang disebut nilai hash. Proses pengubahan menjadi nilai hash menggunakan fungsi *rolling hash* (Karisma Wibowo & Hastuti, 2016). Selanjutnya, setelah didapatkan hashing maka dilakukan proses *fingerprint* dan

matching untuk nilai hash yang sama dan akan dihitung total *nilai hash* yang *match* antara dua dokumen. Total dari nilai *match* yang sama akan digunakan untuk menghitung persentase *similarity* dengan menggunakan *dice coefficient similarity. Dice coefficient similarity* adalah metode pengukuran yang paling umum digunakan untuk menghitung nilai kesamaan dengan pendekatan K-gram (Filcha & Hayaty, 2019).

2.4 ReactJS

Menurut dokumentasi dari situs resmi reactjs.org, ReactJS merupakan *library* bahasa pemrograman javascript yang diciptakan oleh facebook untuk membangun *user interface* sebuah *web application* yang bersifat *open source,*dimana kita dapat membuat beberapa komponen user interface pada aplikasi yang kita kembangkan secara kompleks. ReactJS memiliki beberapa keunggulan seperti kecepatan, *simplicity*, dan *scalability*. ReactJS memungkinkan pengembang dapat membangun sebuah komponen UI yang lebih interaktif, *stateful*, dan *reusable* (Panjaitan & Pakpahan, 2021). Peran ReactJS dalam kaidah MVC adalah sebagai *view* saja sehingga jika ingin menerapkan model MVC sangat disarankan menggunakan ReactJS untuk handle *view*.

React memiliki beberapa fitur utama yang menjadikannya lebih unggul dibandingkan dengan library Javascript lainnya seperti :

- JSX

  JSX adalah syntax Javascript yang digunakan dalam pembuatan elemen React. Developer menggunakannya untuk menyematkan (embed) kode HTML pada object sehingga membantu mempersingkat kode yang kompleks. (Maratkar & Adkar, 2021)
- DOM Virtual

  Document Object Model (DOM) berfungsi untuk menyajikan halaman web dalam tampilan struktur data *(tree*/pohon). ReactJS menyimpan struktur data DOM Virtual ini dalam memorinya sehingga jika terdapat perubahan pada bagian tertentu dalam struktur data tersebut, aplikasi yang dibangun tidak perlu untuk merender ulang kembali semuanya. Setiap kali ada

perubahan data, ReactJS akan membuat satu struktur data DOM Virtual baru dan membandingkannya dengan yang sebelumnya, lalu mencari cara tercepat untuk menerapkan perubahan tersebut pada DOM yang asli. Proses ini disebut dengan *diffing* sehungga proses render perubahn pun bisa lebih cepat dan hemat resource.

- Komponen dan Properti React

  ReactJS memisahkan antara *user interface* menjadi beberapa potongan kode tersendiri yang nantinya bisa digunakan lagi yang disebut sebagai komponen. Cara kerja komponen react mirip dengan fungsi Javascript, yaitu sama – sama menerima input arbitrer yang disebut property atau props.

- Manajemen State

  State adalah *object* Javascript yang mewakili satu bagian dari sebuah komponen. Setiap kali *user* berinteraksi dengan aplikasi, *state* juga akan berubah, dengan merender UI baru guna menampilkan perubahan dari interaksi tersebut. Fitur manajemen *state* ini mengacu upada prosedur yang dilakukan untuk mengelola keadaan sebuah aplikasi React seperti menyimpan data di library manajemen state pihak ketiga, dan memicu proses *rendering* ulang setiap kali data diubah.

2.5 ExpressJS

ExpressJS adalah salah satu *framework* yang berjalan pada *runtime* NodeJS dan dapat digunakan untuk mengembangkan aplikasi besar ataupun kecil, terutama layanan web ataupun *RESFful API.* (Jonsson dkk., 2022). ExpressJS juga bisa digunakan untuk membuat *web framework* yang kompleks seperti MERN (*MongoDB*, *ExpressJS, ReactJS, NodeJS*). Peran dari ExpressJS sendiri dalam arsitektur model MVC adalah sebagai *controller* yang mengatur *logic business* yang dimiliki oleh aplikasi yang telah dibuat. Adapun beberapa keunggulan yang dimiliki oleh *ExpressJS* antara lain :

- Dukungan pembuatan *middleware*.

- Dukungan terhadap berbagai HTTP *verb* seperti POST, GET, PUT, DELETE, OPTION, HEAD, dan *method* lainnya.
- Mudah untuk dimodifikasi.
- Dukungan komunitas yang cukup besar

2.6 Mocha

Mocha adalah framework unit testing berbasis Javascript yang berjalan di NodeJS dan juga browser. Terdapat beberapa *function* yang digunakan pada Mocha yaitu :

- *Function describe()*

  Function describe() adalah cara sederhana untuk mengelompokkan pengujian di Mocha. Fungsi ini memungkinkan untuk membuat serangkaian tes. Secara umum, fungsi describe() membutuhkan 2 argumen. Pertama adalah nama atau deskripsi grup pengujian dan yang kedua adalah *callback function*, yaitu fungsi yang perlu dijalankan setelah fungsi lain selesai dijalankan.
- *Function it()*

  Fungsi it() adlaah cara untuk mendeksripsikan kasus uji individu. Tes ini harus bersarang di dalam blok description (). Fungsi it() harus dijelaskan dengan cara yang masuk akal untuk kasus uji yang diberikan.

2.7 Chai

Chai merupakan library yang digunakan untuk memudahkan Mocha dalam melakukan *assertion* pada setiap API yang dipanggil. Chai memiliki beberapa *interfaces* yang memungkinkan pengembang memilih gaya yang paling nyaman. Gaya BDD berkemampuan *chain-capable* yang memberikan Bahasa ekspresif dan gaya yang dapat dibaca, sedangkan gaya penegasan TDD memberikan nuansa yang lebih klasik.

2.8 RESTful API

API adalah singkatan dari *application programming interface* yaitu sebuah software yang memungkinkan para developer untuk mengintegrasikan dan mengizinkan dua aplikasi yang berbeda secara bersamaan untuk saling terhubung satu sama lain. Tujuan penggunaan dari API adalah untuk saling berbagi data antar aplikasi yang berbeda tersebut sehingga mempercepat proses pengembangan aplikasi dengan cara menyediakan sebuah function yang terpisah sehingga para developer tidak perlu membuat fitur serupa (Edy dkk. 2017).

Web service merupakan sebuah perangkat lunak yang tidak terpengaruh oleh *platform*, arsitektur maupun bahasa pemrograman yang menyediakan layanan atau *method – method* untuk pertukaran data yang dapat diakses oleh *network*. Salah satu implementasi dari *web service* adalah REST atau RESTful (*Representational State Transfer*). REST sendiri memungkinkan *system request* dapat mengakses dan memanpulasi teks yang direpresentasikan dari sebuah *web service. Web service* API menggunakan REST disebut dengan RESTful API (Penidas & Magist, 2016).

REST menentukan sekumpulan prinsip arsitektur yang dapat digunakan untuk merancang *web service* yang berfokus pada sumber daya sistem, termasuk bagaimana sumber daya sistem, termasuk bagaimana sumber daya yang dialamatkan dan dtransfer melalui HTTP oleh berbagai klien yang ditulis dalam bahasa pemrograman yang berbeda (Penidas & Magist, 2016.).

Resource	Method			
	GET	POST	PUT	DELETE
/api/users	*Get a list of all users*	Create a new list of user	Update a list of user	Delete all users
/api/users/1	*Get a user by user's id*	Treat as a collection. Create a new user it.	If student exists, update the user. If user does not	Delete the user by user's id

			exist. Create a new user.	

Tabel 2.8.8.1 Contoh RESTful API

Adapun beberapa metode yang digunakan pada REST antara lain yaitu

- **GET** digunakan untuk mendapatkan data
- **POST** digunakan untuk membuat data baru
- **PUT** digunakan untuk memperbarui data berdasarkan data contohnya id data
- **DELETE** digunakan untuk menghapus data atau sekumpulan data

2.9 JSON Web Token (JWT)

JWT merupakan sebuah token berbentuk string yang terdiri dari tiga bagian yaitu : *header, payload* dan *signature* yang digunakan untuk proses otentikasi dan pertukaran informasi. Token terdiri dari dua jenis : token pembawa dan token pemegang kunci. Sedangkan berdasarkan tujuan terdapat dua skema : token identitas dan token akses. Cara kerja JWT sama seperti password, Ketika pengguna berhasil login maka server akan memberikan token yang disimpan di *local storage* atau *cookies browser*. Token digunakan untuk mengakses halaman tertentu, pengguna akan mengirim balik token tersebut sebagai bukti bahwa pengguna sudah berhasil login.

```
Header : Algoritma dan Tipe Token
  {
  "alg" : "HS256",
  "typ" : "JWT"
  }
Payload : Data
  {
  "iss" : "#appbackend",
  "user" : "#username",
  "pass" : "password"
  }
Signature : hasil dari Hash
  {
  "Base64-encoded(Header.Payload)" + "key" +
"Algorithm"
  }
```

Gambar 2.12.1 Struktur JWT

Gambar 2.12.1 menampilkan tiga bagian utama (*header, payload* dan *signature*) yang dapat digunakan dalam penggabungan untuk menghasilkan JWT. Adapun token yang dihasilkan dari JWT dibangun dengan formula seperti pada gambar 2.12.2

$$\boldsymbol{Token} = f(Base64Encode) \sum_{n=\alpha,\beta}^{\infty} (header.payload.signature)$$

Gambar 2.12.2 Formula JWT

Gambar 2.12.2 menampilkan formula JWT yang merupakan fungsi dari *Base64Encode* dengan parameter *header, payload* dan *signature*. Token yang diperoleh dari server akan disisipkan pada *header Hypert Text Transfer Protocol (HTTP)*, seperti pada gambar 2.12.3

Gambar 2.12.3 Contoh Token pada *Header HTTP.*

2.10 Esprima

Esprima merupakan sebuah *library* pada Javascript yang digunakan untuk melakukan *syntactic analysis* maupun *lexical analysis*. Fungsi esprima adalah melakukan penguraian (*parsing*) terhadap kode program Javascript. Hal ini dikenal sebagai *lexical analysis* (Umniyah, 2020). *Lexical Analysis* adalah proses konversi urutan karakter (seperti kode program) ke dalam urutan token. Jadi, nantinya hasil dari proses konversi ini akan diidentifikasi sesuai dengan yang telah ditetapkan sehingga akan berbentuk berupa token.

Lexical token atau *simply token* adalah string dengan makna yang ditetapkan dan telah diidentifikasi. Token ini disusun sebagai pasangan yang teridiri dari nama token dan nilai token opesional. Nama token adalah kategori unit leksikal. Adapun beberapa contoh nama token yang umum yaitu :

Token Name	**Sample token values**
Identifer	x color, UP
Keyword	if, while, return
Separator	}, (, ;
Operator	+ , <, =
Literal	True, 6.02e23, “music”
Comment	/* *Retreves user data */. // must be negative*

Tabel 2.10.10.1. Tabel Contoh Nilai Token

Pada gambar 2.6.1 adalah contoh token yang umum digunakan . Nama token nantinya akan dibuat menjadi nilai alfabet sebagai penanda . Contohnya, token *Identifier* yang akan diubah menjadi huruf **'A'**.

2.11 Node.JS

Node.JS adalah *runtime environment* untuk Javscript yang bersifat open-source dan cross-paltform. Dengan Node.js kita dapat menjalankan kode Javascript di mana pun , tidak hanya terbatas pada lingkungan browser. Node.js menjalankan V8 Javascript engine (yang juga merupakan inti dari Google Chrome) di luar browser sehingga memungkin Node.js memiliki performa yang tinggi. Node.js menyediakan banyak library/module Javascript yang membantu menyederhanakan pengembangan aplikasi web. Berikut adalah beberapa fitur penting dari Node.js

- ***Asynchronous & Event-driven***
  Semua API dari Node.js bersifat asynchronous, artinya tidak memblokir proses lain sembari menunggu satu proses selesai. Server Node.js akan melanjutkan ke pemanggilan API berikutnya lalu memanfaatkan mekanisme event notification untuk mendapatkan respon dari panggilan API sebelumnya.
- ***Very Fast***
  Eksekusi kode dengan Node.js sangat cepat karena berjalan pada V8 Javascript engine dari Google Chrome.
- ***Single Threaded but Highly Scalable***
  Node.js menggunakan model single thread dengan event looping. Mekanisme ini membantu server untuk merespon secara asynchronous dan menjadikan server lebih scalable dibandingkan server tradisional

Node.js dirancang untuk aplikasi dengan proses I/O yang intensif seperti network server atau backend API. Pemrograman dengan multithreading relative lebih berat dan sulit untuk dilakukan. Jika kita ingin membuat web server yang bisa menangani ratusan request bersamaan, menggunakan ratusan thread akan mebutuhkan memori yang bessar. Oleh karena itu, karakteristik Node yang

asynchronous dan single thread dirancang untuk memungkinkan implementasi server yang dapat menangani banyak request pada waktu sama.

2.12 Node Package Manager

Node Package Manager (NPM) merupakan pengelola package untuk JavaScript yang dapat memudahkan pengguna dalam mengelola package yang tersedia pada https://www.npmjs.com/. NPM ini menjadi standard package manager yang disediakan oleh Node.js dan otomatis terpasang ketika memasang Node.JS pada komputer kita. Menurut dokumentasi dari situs resmi NPM , bahwa dengan adanya NPM memudahkan *developer* dalam berkolaborasi karena *package* yang ada di NPM bisa digunakan secara gratis jika package tersebut bersifat public dan dikenakan biaya jika NPM tersebut bersifat pribadi biasaya merupakan kebutuhan suatu organisasi .

2.13 MongoDB

MongoDB adalah basis data NoSQL yang bersifat *document based,* artinya hanya tersusun atas koleksi dan dokumen. MongoDB bersifat *document based* artinya hanya memiliki koleksi dan dokumen. Data yang disimpan dalam basis data *MongoDB* berupa file JSON yang disebut dengan istilah *BSON* (Binary JSON). Sistem basis data *MongoDB* menggunakan pasangan key-value, artinya setiap dokumen dalam *MongoDB* dipastikan memiliki key (Cahyo Santoso dkk., 2020.).

Adapun kelebihan yang dimiliki oleh MongoDB adalah memiliki format yang cepat karena mampu mencached dan juga model datanya berbasis dokumen, berbentuk seperti file JSON yang disebut BSON. Model datanya yang berbasis dokumen membuat penggunanya tidak perlu merancang struktur table seperti pada SQL. MongoDB bisa membuat struktur tabelnya sendiri secara otomatis pada saat melakukan Insert karena skemanya yang lebih fleksibel. MongoDB juga memiliki penyimpanan data yang lebih besar serta *low-cost* sebagai basis data NoSQL (Cahyo Santoso dkk., 2020).

2.14 Mongoose

Mongoose adalah sebuah Object Document Mapper (ODM). Artinya, mongoose berfungsi utuk mendefinisikan obyek dengan skema yang benar – benar diketik yang dipetakan ke sebuah dokumen MongoDB. Mongoose menyediakan fungsionalitas yang luar biasa dalam pembuatan dan pengerjaan skema. Mongoose saat ini memiliki delapan tipe skema dimana propertinya disimpan seperti saat berada di MongoDB. Diantaranya:

- String
- Number
- Date
- Buffer
- Boolean
- Mixed
- ObjectId
- Array

Setiap tipe data menmungkinkan untuk menentukan :

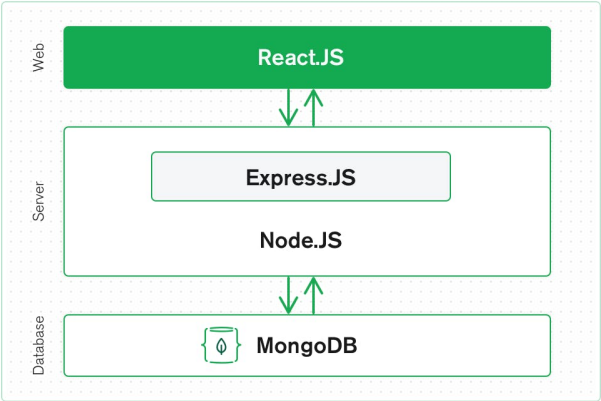
- Sebuah nilai default
- Sebuah fungsi validasi custom
- Menujukkan field yang dibutuhkan
- Fungsi get yang memungkinkan untuk memanipulasi data sebelum dikembalikan sebagai object
- Sebuah set fungsi yang memungkinkan untuk memanipulasi data sebelum disimpan ke dalam databse
- Membuat indeks yang memungkin data agar ditarik secara lebih cepat.

2.15 MERN Stack

MERN adalah sebuah istilah yang digunakan untuk menggambarkan sekumpulan teknologi berbasis Javascript yang digunakan dalam proses pengembangan aplikasi web. MERN memungkinkan *developer* untuk membangun *3-tier architecture* (frontend, backend, database) seluruhnya menggunakan satu

Bahasa pemrograman yaitu Javascript. MERN merupakan singkatan dari (Mongo, Express, React dan Node) yang digabung menjadi sebuah *stack*.

- **MongoDB** document database
- **ExpressJS** NodeJS Framework
- **ReactJS** client side framework
- **NodeJS** premier javascript web server



Gambar 2.15.1 MERN Stack